FROM SEOUL TO SONY:

THE HISTORY OF THE DARKSEOUL GROUP AND THE SONY INTRUSION MALWARE DESTOVER

By Snorre Fagerland, Blue Coat Systems Inc.

February 2016

EXECUTIVE SUMMARY

The attack on Sony Pictures Entertainment in November 2014 was not a single incident. Through technical indicators, we connect the attack to several destructive events going back to at least 2009.

The identity of the perpetrators is unknown, but several of these previous events have been attributed by others to North Korean threat actors. In this report, we show how we have connected these events to the threat actors known as **DarkSeoul** or **Silent Chollima**.

Whoever they are, this group is still active, mainly going after South Korean targets in several sectors. Malware belonging to this threat complex has apparently been produced as late as January 2016.

We detail the evolution of some of the most common tools used by these attackers and present indicators of compromise and mitigation information where we can.

In parallel with this report, the security company Novetta is publishing its own independent research covering the same threat complex. This report is available from http://operationblockbuster.com.

INTRODUCTION

Much has been written about the Sony hack. However, hard data has not been as plentiful. In an attempt to provide additional insight, we detail some facts about the malware reportedly used in the attack, and attempt to draw lines to other malware and incidents, beyond the mere speculative.

In order to expand the case, we will look at a variety of evidence. In most cases, we will not settle for one single factor as the basis for assessments, but instead correlate information of different kinds. Factors that we will include are for example:

- Obfuscation methods
- Code structure
- Text strings, such as encryption keys
- Known localization
- Digital code signing certificates

Details about the different indicators are included in the appendixes.

Acknowledgements

A big thank you goes out to all who helped with this paper – notably Waylon Grange, always an invaluable source of insight and information, and the good folks over at Farsight Security who gracefully provided passive DNS data.

MALWARE KNOWN TO BE CONNECTED WITH THE SONY CASE

To start at the beginning: The official statements from the FBI (1) and US-CERT (2) mention the md5 hashes of the following set of malware files:

d1c27ee7ce18675974edf42d4eea25c6 (dropper)
760c35a80d758f032d02cf4db12d3e55 (wiper)
e1864a55d5ccb76af4bf7a0ae16279ba (web server)
e904bf93403c0fb08b9683a9e858c73e (backdoor)

In the weeks following the attack, a number of other malware instances came to light that were obviously connected; such as

2618dd3e5c59ca851f03df12c0cab3b8 (SMB worm)
b80aa583591eaf758fd95ab4ea7afe39 (wiper)
6467c6df4ba4526c7f7a7bc950bd47eb (backdoor)

Most vendors now use the name Destover for a group of malware that was part of the Sony intrusion. Though many pieces of malware are somewhat different, we'll use that name as well to avoid confusion.

The US-CERT advisory also mentions the import hashes of a number of other malware. These are non-unique indicators, but can help in locating related samples.

A NOTE ABOUT THE HANGUL WORD PROCESSOR (*.HWP, HWPX) FORMAT

The Hangul Word Processor is software developed by the Korean company Hancom. It is similar in usage area to Microsoft Word, but is specifically adapted to the Korean written language Hangul.

The file format used by this software is also somewhat similar to Microsoft Word, with the use of OLE2-based documents for previous versions of HWP, and ZIP archive-based documents for newer versions.

A number of vulnerabilities have existed for these formats. These have been used maliciously by several different threat actors over time, also by the threat actors mentioned in this paper.

MALWARE ARCHEOLOGY

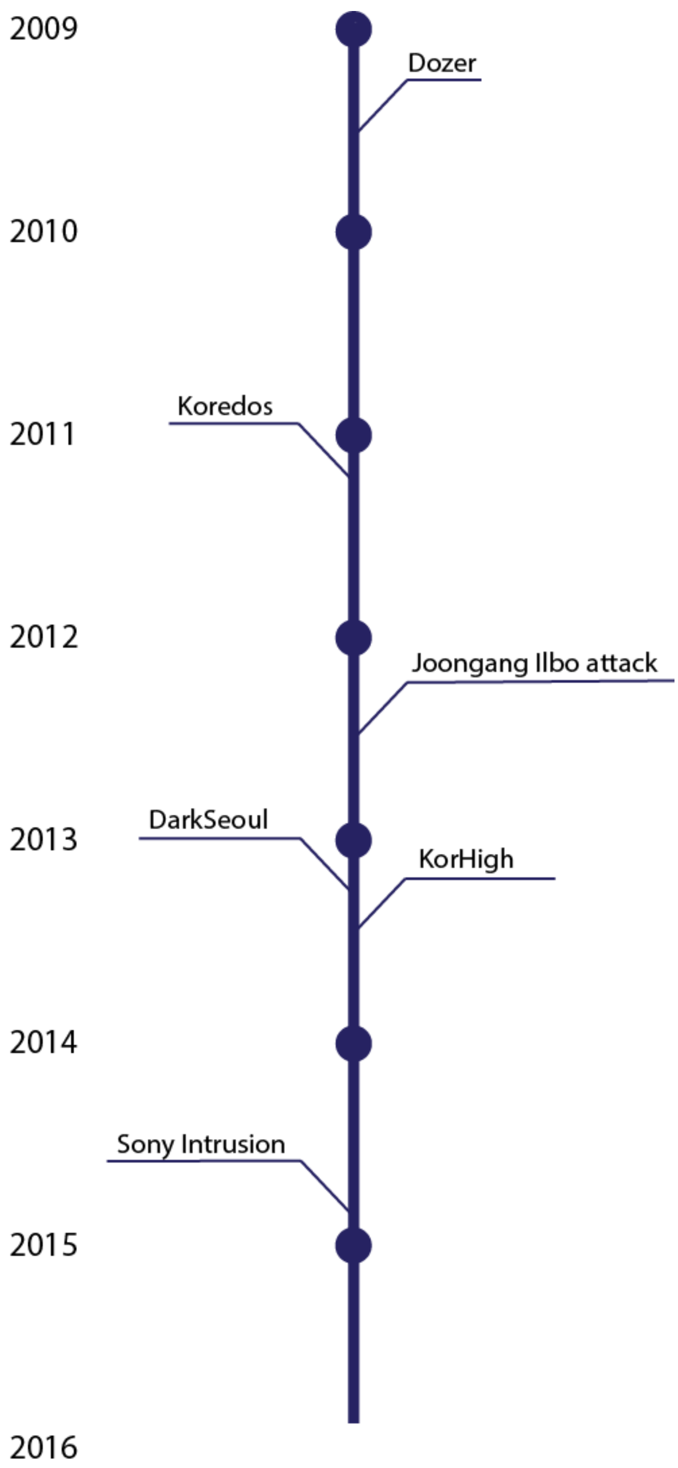
As research into this case progressed, it became obvious that we were tracing malware relationships back in time. In fact, the earliest indicators we've found go all the way back to at least 2009.

Around this time a malware development project started that would become the backbone of intrusions and destructive attacks against mainly South Korean targets for years to come. In fact, modern-day malware from the same threat actor still contains traces of this first eo-malware. The initial starting points were likely publicly available source codes for Rbot and Mydoom, found on Chinese code sharing sites like Programmers United Develop Net (PUDN).

There is no universally adopted naming for the early generations of this family in the AV industry. Usually they are detected as Dllbot or Npkon, but these names can also cover other families, thus our use of a different name in this paper - **KorDllbot**.

We will cover the evolution of KorDllbots and related malware, and how these came to be involved in various intrusion cases.

TIMELINE OF LIKELY DARKSEOUL-RELATED ATTACKS



A timeline of destructive intrusions in or related to the Korean peninsula.

THE KORDLLBOT BACKDOOR FAMILY

KorDllbot is a family of small/medium size trojans that usually are configured to be installed as services.

Samples can vary a great deal in functionality - from just listening on a port and accepting commands, to harvesting data, to actively spreading over SMB. This functionality seems almost modular, using different encryption and encoding methods and different C&C command words. Build environment for the early generations was typically Visual Studio 6.

```
int __cdecl KorDllbot__DeleteFile(int a1, LPCSTR lpFileName)
{
  int result; // eax@2

  if ( DeleteFileA(lpFileName) )
    result = KorDllbot__SendStatus(a1, 0x1E27, 0);
  else
    result = KorDllbot__SendStatus(a1, 0x1E28, 0);
  return result;
}
```

```
int __cdecl KorDllbot2__DeleteFile(const CHAR *a1)
{
  int result; // eax@2

  if ( DeleteFileA(a1) )
    result = KorDllbot2__SendStatus(s_socket, 0x20110512, 0);
  else
    result = KorDllbot2__SendStatus(s_socket, 0x20110513, 0);
  return result;
}
```

```
int __cdecl KorDllbot3__DeleteFile(const CHAR *a1)
{
  int result; // eax@2

  if ( DeleteFileA(a1) )
    result = KorDllbot3__SendStatus(s_socket, 0x4273461, 0);
  else
    result = KorDllbot3__SendStatus(s_socket, 0x4273462, 0);
  return result;
}
```

```
int __cdecl KorDllbot4__DeleteFile(const CHAR *a1)
{
  int result; // eax@2

  if ( DeleteFileA(a1) )
    result = KorDllbot4__SendStatus(s_socket, 0x34567811, 0);
  else
    result = KorDllbot4__SendStatus(s_socket, 0x34567812, 0);
  return result;
}
```

KorDllbots use C&C commands starting at different integer offsets depending on version. Here, versions 1.1/1.2/1.5, 1.03, 1.04.2 and 1.05.2 sending success or error status back to remote control client after file deletion.

Common capability seen in the KorDllbot family is:

- Get bot status
- List logical drives
- List directory
- Change directory
- Get process list
- Kill process
- Execute file
- Delete file
- Change file time
- Execute shell command
- Download file
- Upload file
- Get volume serial number
- Get file attributes

Most of these trojans use encrypted or encoded C&C communication, but the algorithms vary between versions.

A very common trait in these bots is for API's to be dynamically declared through the use of LoadLibrary and GetProcAddress, where the API names are obfuscated, encoded or encrypted in some way, and decoded before they are declared. This is not unique to KorDllbots, but is a fairly static common behavior for this family.

Another trait which is peculiar enough to be *an identifier in itself* is the way this malware creates command line statements. The construction of the command line is deliberately obfuscated by concatenating string segments. Typically, this looks something like this:

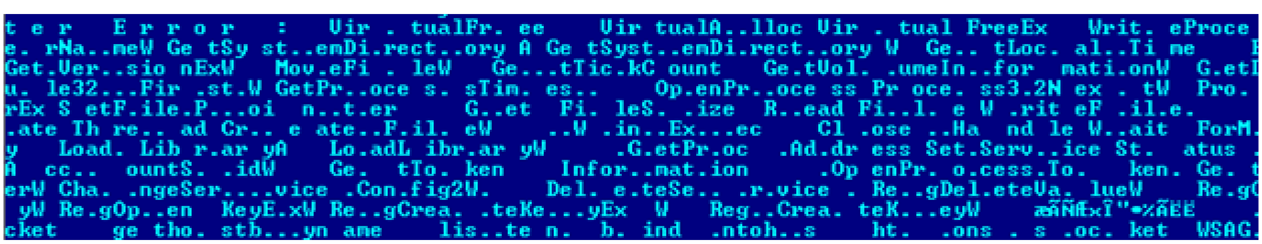
```
sprintf(commandline, "%sd.e%sc %s >%s 2>&1", "cm", "xe /", command, logfile_name);
//command and tempfile_name are arbitrary strings inserted by the malware.
```

This translates to "cmd.exe /c *command>logfile_name* 2>&1", i.e execute command and direct output to a log file. This particular construct, with very little deviation, is used in almost all KorDllbots and its successors. We'll reference this by the name "CMXE" string obfuscation later on in the paper.

The earliest KorDllbot we have has a compile timestamp of July 1st. 2007. This date is however possible to falsify. The earliest *verified* time KorDllbots were observed was mid-2011, with the executable with the sha256 hash of 87bae4517ff40d9a8800ba4d2fa8d2f9df3c2e224e97c4b3c162688f2b0d832e. This sample listens for connections on port 179 and allows remote access through an encoded proprietary protocol.

Already here we can note a connection to the Sony case. Current antivirus detection of this file includes the names Destover and Escad, names introduced by AV vendors in connection with the Sony attack. It has a compile date (May 17th 2011) and import hash that matches data from the US-CERT advisory (2)**.**

This malware contains a very noticeable API string obfuscation algorithm where API strings have been broken up into segments of varying size using either spaces or dots as filler. This is presumably done to avoid detection by anti-malware solutions or YARA rules. We have called this technique **Chopstring**, just to have a reference later on. ChopString is used by many KorDllbots, and also shows up elsewhere in the Sony intrusion case.



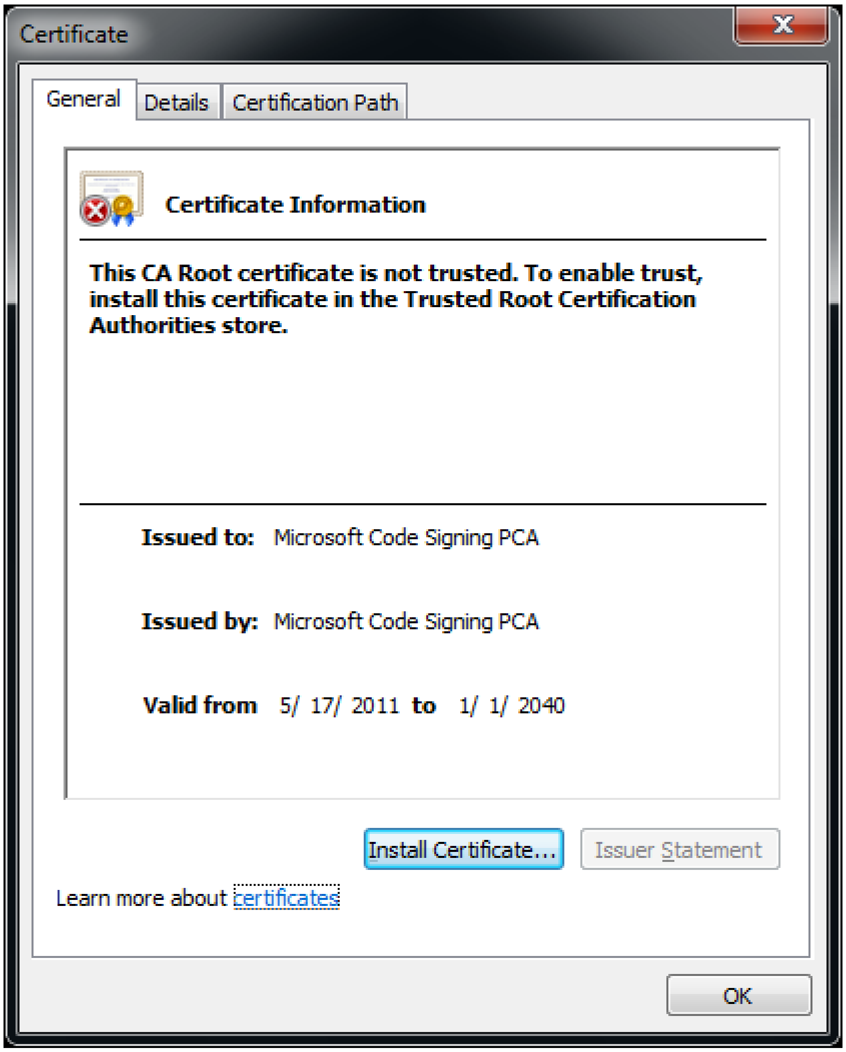
Chopstring'ed strings inside malware.

As far as we know, this exact method is not in widespread use in the underground or shared between threat actors. These APIs are reconstructed before use by calling special string-deobfuscation functions early in the execution of the program. For details about this and other algorithms, see the Appendix.

However, there is another interesting trait of this particular sample, and that is its *digital signature.*

THE MicrosoftCodeSigningPCA CERTIFICATE CLUSTER

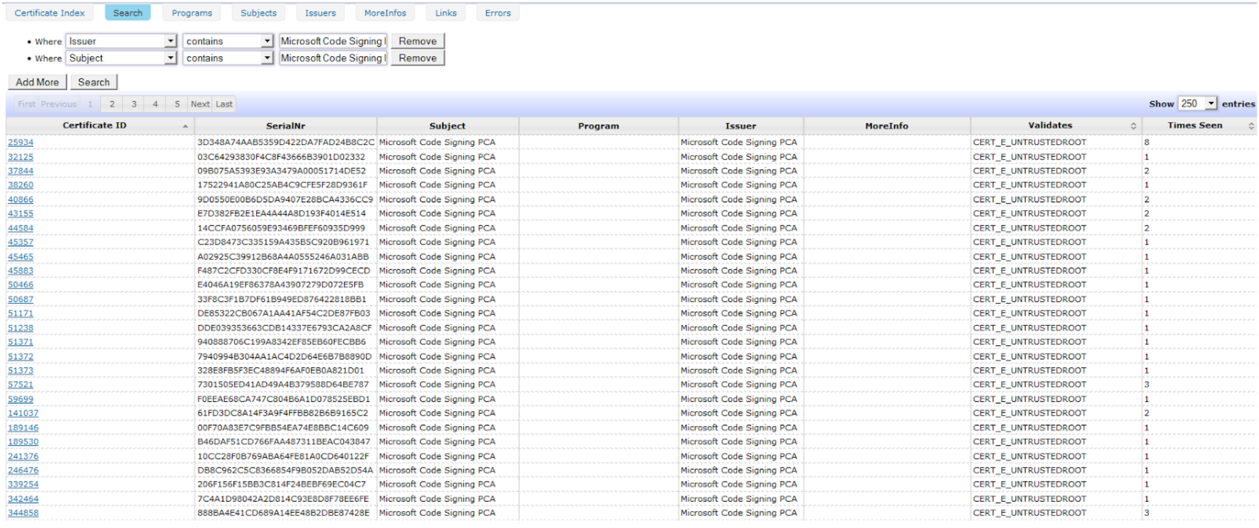
The KorDllbot sample 87bae4517ff40d9a8800ba4d2fa8d2f9df3c2e224e97c4b3c162688f2b0d832e is digitally signed using a non-original (and thus non-validating) Microsoft certificate. The file is in reality self-signed.



This signature doesn't say much about who made it. However, the way the certificate is constructed is peculiar. The faked issuer in this case is *Microsoft Code Signing PCA*. The real Microsoft Code Signing PCA is one of the certificate authorities used by Microsoft to sign their software.

The Subject - i.e. the entity the certificate is supposed to have been issued to - is **also** Microsoft Code Signing PCA. This is a construct never seen in legitimate certifications, and it is rare enough in faked certificates that it's worthwhile checking other malware signed in this way.

Blue Coat maintains a database of code signing certificates which we can mine for this type of information.



Certificate ID	SerialNr	Subject	Program	Issuer	MoreInfo	Validates	Times Seen
25934	3D348A74AAB5359D422DA7FAD24B8C2C	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	8
32125	03C64293830F4C8F43666B3901D02332	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
37844	09B075A5393E93A3479A00051714DE52	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	2
38260	17522941A80C25AB4C9CFE5F28D9361F	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
40866	9D0550E00B6D5DA9407E28BCA4336CC9	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	2
43155	E7D382FB2E1EA4A44A8D193F4014E514	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	2
44584	14CCFA0756059E93469BFEF60935D999	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	2
45357	C23D8473C335159A435B5C920B961971	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
45465	A02925C39912B68A4A0555246A031ABB	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
45883	F487C2CFD330CF8E4F9171672D99CECD	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
50466	E4046A19EF86378A43907279D072E5FB	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
50687	33F8C3F1B7DF61B949ED876422818BB1	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
51171	DE85322CB067A1AA41AF54C2DE87FB03	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
51238	DDE039353663CDB14337E6793CA2A8CF	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
51371	940888706C199A8342EF85EB60FECBB6	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
51372	7940994B304AA1AC4D2D64E6B7B8890D	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
51373	328E8FB5F3EC48894F6AF0EB0A821D01	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
57521	7301505ED41AD49A4B379588D64BE787	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	3
59699	F0EEAE68CA747C804B6A1D078525EBD1	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
141037	61FD3DC8A14F3A9F4FFBB82B6B9165C2	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	2
189146	00F70A83E7C9FBB54EA74E8BBC14C609	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
189530	B46DAF51CD766FAA487311BEAC043847	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
241376	10CC28F0B769ABA64FE81A0CD640122F	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
246476	DB8C962C5C8366854F9B052DAB52D54A	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
339254	206F156F15BB3C814F24BEBF69EC04C7	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
342464	7C4A1D98042A2D814C93E8D8F78EE6FE	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	1
344858	888BA4E41CD689A14EE48B2DBE87428E	Microsoft Code Signing PCA		Microsoft Code Signing PCA		CERT_E_UNTRUSTEDROOT	3

We found several certificate serial numbers matching this pattern. Each serial number identifies a certificate used to sign a small number of malware samples – typically on the range of one to four samples, with one outlier at eight samples.

The malware can be clustered into a few main buckets. Some malwares of different families are signed by the same certificate, which creates a high-confidence link between them.

This collection of signed malware is dominated by KorDllbots. These are not all identical, there is considerable variation between generations in functionality, encoding and encryption methods, but the similarities in overall structure; string usage etc. is quite unmistakable. (See appendix for a full list of executables with this type of signature.)

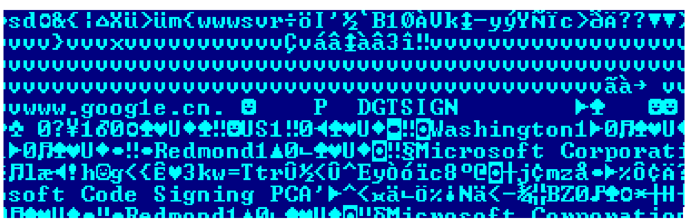
Other samples include keyloggers, SMB worms, Yahoo Messenger-communicating backdoor trojans and the legitimate ProxyMini lightweight proxy server.

KORDLLBOT-RELATED SMB WORMS

The malware samples **163571bd56001963c4dcb0650bb17fa23ba23a5237c21f2401f4e894dfe4f50d** and **e0cd4eb8108dab716f3c2e94e6c0079051bfe9c7c2ed4fcbfdd16b4dd1c18d4d** in the cluster of signed malware do not look like KorDllbots at first glance.

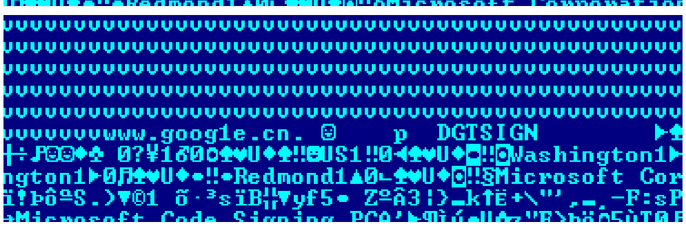
The usual service DLL dropper is here replaced with a worm component. After installation and reboot, this worm generates random IP addresses and attempts to connect to the *admin$* share on remote machines using the hard coded usernames “administrator” and “db2admin”. The malware contains a list of common passwords and it will also construct passwords based on the username. If successful, the worm copies itself to the remote machine’s system directory and installs it as a service there.

In addition to spreading, these samples drop a backdoor component which is somewhat different in structure to the “standard” KorDllbots. The dropper code logic used in these worms is however used in other KorDllbot dropper samples and is unmistakable - the strings “DGTSIGN” and “www.goog1e.cn” are markers which the malware uses to locate its embedded content.



9bc8fe605a4ad852894801271efd771da688d707b9fbe208106917a0796bbfdc

This is a KorDllbot dropper



e0cd4eb8108dab716f3c2e94e6c0079051bfe9c7c2ed4fcbfdd16b4dd1c18d4d

This is an SMB worm

THE JOANAP/BRAMBUL WORM FAMILY

Speaking of SMB worms, a group of malware signed using the MicrosoftCodeSigningPCA pattern were a series of SMB worms that had not appeared on our radar before. The variant we found first was named “Joanap” by several antivirus vendors; presumably because of name appearing in the TO: field of callback emails from the malware – “Joana.”

The malware comes as a dropper which installs three sub-components – one SMB spreading DLL (wmmvsvc.dll), one backdoor DLL (scardprv.dll) and one configuration file (mssscardprv.ax).

The spreader component generates random IP addresses and attempts to copy the dropper and the config file to these over SMB. If successful, the worm sends an email back to its creator via Google’s SMTP server. The backdoor component is essentially a KorDllbot. Not only is there code overlap with this family, but it also creates its API decryption AES key based on the same string (“**Bb102@jH4$t3hg%6&G1s*2J3gCNwVr*UeI!Dr3hytg^CHGf%ion**”) as previously mentioned KorDllbots, eg. sha256 a795964bc2be442f142f5aea9886ddfd297ec898815541be37f18ffeae02d32f.

Recently, Symantec published information (3) that links these worms to the Duuzer malware family. As we shall see later on, this is just another connection to our threat actors.

We were able to locate several variants of Joanap-like malware using different email addresses and containing different functionality. The earliest of these were apparently compiled as early as January 2009, with verified occurrences of a newer variant late same year. See appendix for more details.

The latest versions of Joanap we found appear to be the type of SMB worm observed in connection with the Sony attack, something also PriceWaterhouseCoopers has mentioned in a blog post (4).

THE DOZER (AKA 7.7 DDOS) ATTACK

The Dozer attack in July 2009 was one of the first attacks on South Korean targets that received international attention. DDOS bots were distributed with lists of sites to attack – notably various Korean websites covering government and bank functions, but also a great deal of US .gov, .mil and .com sites – including *whitehouse.gov*. This also involved wiping of hard disks of the infected computers.

There is a known set of malware (7) connected with this incident.

Some of these samples appear to have been written specifically for the Dozer attack. However, the sample with the sha256 hash 7dee2bd4e317d12c9a2923d0531526822cfd37eabfd7aecc74258bb4f2d3a643 shares code with KorDllbots, as can be seen in the function below, which does network receipt with xor decoding.

```
mov     [esp+1001Ch+var_10008], 1
call    ds:GetTickCount
mov     esi, eax

                        ; CODE XREF: KorDllbot__RecvAndDecode+B7↓j
                        ; KorDllbot__RecvAndDecode+E7↓j
call    ds:GetTickCount
mov     ecx, [esp+1001Ch+arg_8]
sub     eax, esi
cmp     eax, ecx
ja      loc_10001F37
mov     ecx, [esp+1001Ch+arg_0]
push    0               ; _DWORD
lea     eax, [esp+10020h+var_10004]
push    ebp             ; _DWORD
push    eax             ; _DWORD
push    ecx             ; _DWORD
call    recv
mov     ebx, eax
test    ebx, ebx
jz      loc_10001F37
cmp     ebx, 0FFFFFFFFh
jz      short loc_10001ED0
lea     edx, [esp+1001Ch+var_10004]
push    ebx
push    edx
call    KorDllbot__XorDecode
mov     edi, [esp+10024h+var_1000C]
mov     eax, [esp+10024h+arg_4]
sub     edi, ebp
mov     ecx, ebx
add     edi, eax
mov     eax, ecx
lea     esi, [esp+10024h+var_10004]
add     esp, 8
shr     ecx, 2
rep movsd
mov     ecx, eax
sub     ebp, ebx
and     ecx, 3
rep movsb
call    ds:GetTickCount
mov     esi, eax
jmp     short loc_10001EE5
---------------------------------------------------------------

                        ; CODE XREF: KorDllbot__RecvAndDecode+61↑j
call    WSAGetLastError
cmp     eax, 2733h
jnz     short loc_10001F37
push    1Eh             ; dwMilliseconds
call    ds:Sleep
```

```
mov     [esp+101Ch+var_1008], 1
call    ds:GetTickCount
mov     esi, eax

                        ; CODE XREF: KorDllbot__RecvAndDecode+B7↓j
                        ; KorDllbot__RecvAndDecode+E7↓j
call    ds:GetTickCount
mov     ecx, [esp+101Ch+arg_8]
sub     eax, esi
cmp     eax, ecx
ja      loc_402837
mov     ecx, [esp+101Ch+s]
push    0               ; flags
lea     eax, [esp+1020h+buf]
push    ebp             ; len
push    eax             ; buf
push    ecx             ; s
call    recv
mov     ebx, eax
test    ebx, ebx
jz      loc_402837
cmp     ebx, 0FFFFFFFFh
jz      short loc_4027D0
lea     edx, [esp+101Ch+buf]
push    ebx
push    edx
call    KorDllbot__XorDecode
mov     edi, [esp+1024h+var_100C]
mov     eax, [esp+1024h+arg_4]
sub     edi, ebp
mov     ecx, ebx
add     edi, eax
mov     eax, ecx
lea     esi, [esp+1024h+buf]
add     esp, 8
shr     ecx, 2
rep movsd
mov     ecx, eax
sub     ebp, ebx
and     ecx, 3
rep movsb
call    ds:GetTickCount
mov     esi, eax
jmp     short loc_4027E5
---------------------------------------------------------------

                        ; CODE XREF: KorDllbot__RecvAndDecode+61↑j
call    dword_413AC0
cmp     eax, 2733h
jnz     short loc_402837
push    1Eh             ; dwMilliseconds
call    ds:Sleep
```

KorDllbot (0075d16d8c86f132618c6365369ff1755525180f919eb5c103e7578be30391d6) vs Dozer (7dee2bd4e317d12c9a2923d0531526822cfd37eabfd7aecc74258bb4f2d3a643).
The function is identical. This is just one out of several such functions in the sample.

We can say with reasonable confidence that the threat actors behind the Dozer attack also were involved in the creation of the KorDllbot family or have had access to the source code.

THE KOREDOS (AKA 3.4 DDOS) ATTACK

Over a few days in the beginning of March 2011, different South Korean organizations were targets of a DDOS attack. The malware launching this attack also contained very destructive components that wiped and deleted files of certain extensions after some time, as well as overwriting the Master Boot Record (MBR) of all physical hard drives. Good write-ups of this incident have been published by McAfee (8) and several others.

Some known *Koredos* malware samples (eg. sha256 48dee93aa3ea847da119f5104e8f96070b03f1d52c46f39dc345f0102bf38836) use the same RC4 file decryption key - "**A39405WKELsdfirpsdLDPskDORkbLRTP12330@3$223%!**" - as malware in the MicrosoftCodeSigningPCA signed KorDllbot cluster mentioned previously (eg. sha256 a795964bc2be442f142f5aea9886ddfd297ec898815541be37f18ffeae02d32f). The RC4 implementation used is identical. The very same KorDllbot also contains an AES key – **"Bb102@jH4$t3hg%6&G1s*2J3gCNwVr*UeI!Dr3hytg^CHGf%ion**" which is used by several Joanap malware samples.

We can say with reasonable confidence that the threat actors behind the Koredos attack, like in the Dozer attack, have been involved in the creation of the KorDllbot family.

Symantec reported another malware to be involved along with the Koredos malware - the stealthy backdoor Prioxer (9). Prioxer made a return in connection with the DarkSeoul (often known as Jokra) attacks in 2013. This relationship has been covered by in studies by both Symantec (10) and McAfee (5).

THE JOONGANG ILBO ATTACK

In 2012, the conservative daily newspaper Joongang Ilbo was subject to a disk wiping attack (11).

Not much technical data is in the public domain about this incident. However, a Korean researcher links this attack to the Sony attack, based on code similarities (12). We have no reason to doubt this assessment.

THE DARKSEOUL (AKA 3.20 OR JOKRA) ATTACK

DarkSeoul was a debilitating and destructive attack in March 2013 that affected several Korean banks and news organizations. It may be the most well-known of all the Korean “wiper” attacks. The incident has been extensively researched by several vendors; notably the mentioned Operation Troy paper (5) by McAfee covered a good deal of the malware involved.

The main malware family connected with that attack – an IRC controlled bot – was a programming project that had been ongoing for years before being employed in the DarkSeoul attack. The earliest sample we have of this family (known as XwDoor or Keydoor) was apparently compiled in January 2009. This family is quite easy to spot, as there are a number of strings that appear consistently re-used. The intrusion also involved a backdoor family named Prioxer. There was no obvious connection to the KorDllbot/Destover complex until Symantec tied the Prioxer malware back to the 2011 Koredos incident (10).

THE KORHIGH MALWARE

The Korhigh malware was identified around June 25 2013 in connection with investigations into other attacks on South Korean targets (13). This date coincided with the 63rd anniversary for the start of the Korean War. It had a destructive component, capable of deleting files and overwriting the Master Boot Record (MBR) of hard drives.

The malware was apparently created by a group calling itself "High Anonymous." The following image was contained as a resource in one of the executables:

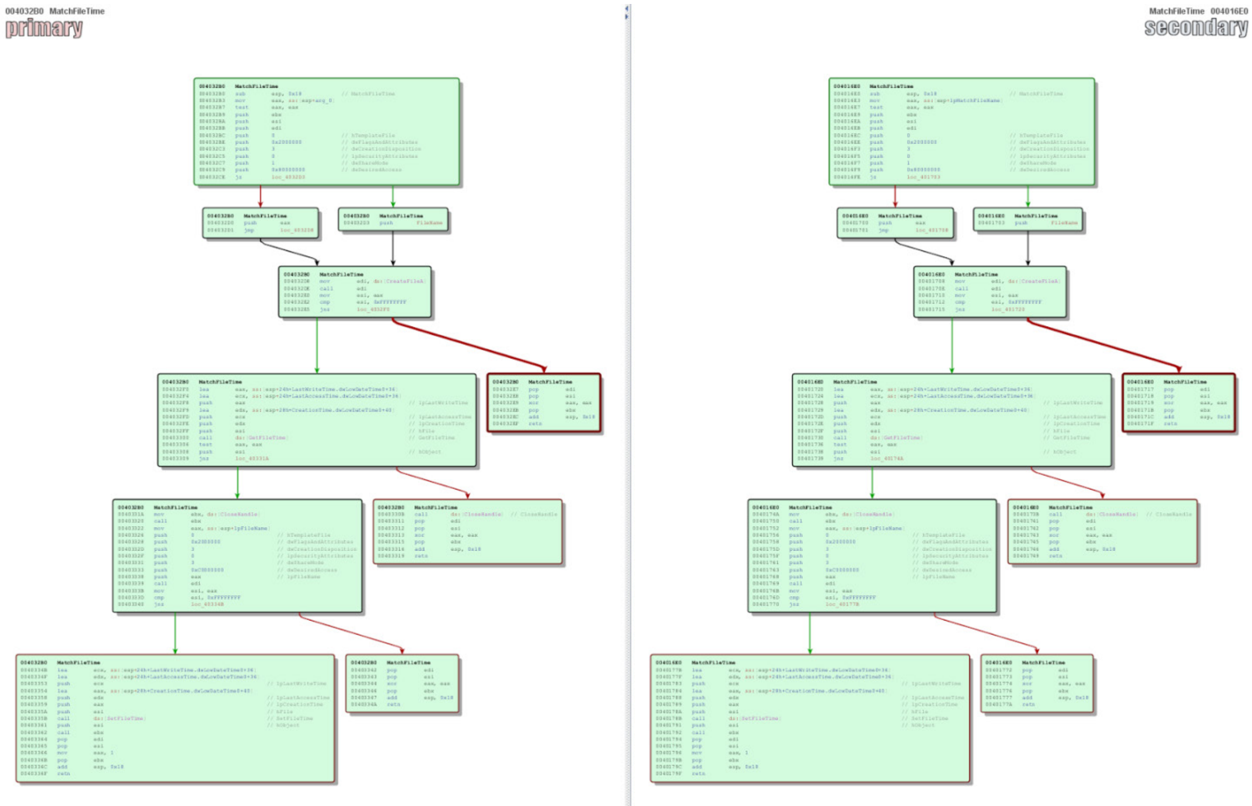
There are strong similarities between the Sony malware and the malware used in the Korhigh campaign. These similarities have been reported by Korean researchers (13), but have gone largely unnoticed in the West.

Comparing 4d4b17ddbcf4ce397f76cf0a2e230c9d513b23065f746a5ee2de74f447be39b9 from the Sony attack with 5b5aede68a6b3aa50cd62c5f4f02078620f0b7be4ceb679b6d5dfe25a44b8cb9 from the Korhigh attack we see code reuse. Specifically, the code used for spreading over the network is almost identical. The technique used by both goes as follows:

1. Scan for computers that have ports 139 and 443 open
2. Test the remote login credentials by attempting to access the admin$ share
3. If successful, create a remote service with the name "RasMgrp " and description "RasSecruity".
4. Use the commands "cmd.exe /q /c net share shared$=%SystemRoot%" and "cmd.exe /q /c net share shared$=%SystemRoot% /GRANT:everyone,FULL" to create a "shared$" share.
5. Copy itself over to the share
6. Match the new file's timestamp to that of the local "calc.exe"
7. Delete the share using the same service name, this time with the command "cmd.exe /q /c net share shared$ /delete"

Even the filenames used when copying itself over the share are similar:

Destover filenames	**Korhigh filenames**
recdiscm32.exe	recdiscm.exe
taskhosts64.exe	taskhosts.exe
taskchg16.exe	taskchg.exe
rdpshellex32.exe	rdpshellex.exe
mobsynclm64.exe	mobsynclm.exe
comon32.exe	comon32.exe
diskpartmg16.exe	diskpartmg.exe
dpnsvr16.exe	dpnsvr32.exe
expandmn32.exe	expandmn.exe
hwrcompsvc64.exe	hwrcompsvc.exe

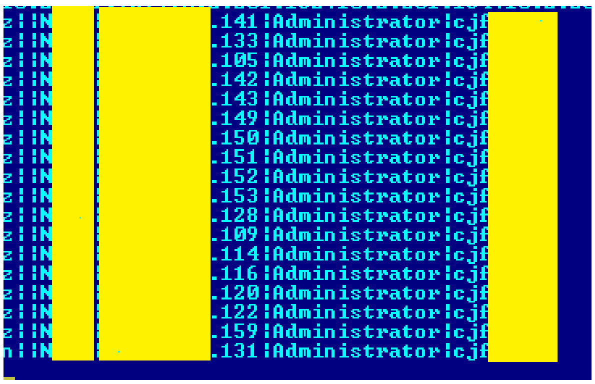


File timestamp matching function comparison

There is little doubt that parts of the same codebase has been used in both of these attacks.

In the Sony incident, several malware samples contained information that seemed to indicate foreknowledge about the layout of the targeted networks. This included local hostnames, usernames and even passwords.

This was also the case in the Korhigh attack. At least two samples (5b5aede68a6b3aa50cd62c5f4f02078620f0b7be4ceb679b6d5dfe25a44b8cb9, d6a07b7ecd5ae7e948cce032603558a5d21100ba5f04056c72aec1ab2d36956e) came with pre-defined configurations containing domain, hostname, username and password combinations. Though we have no hard data to confirm this, it could mean that Korhigh was part of an actual intrusion at the time.



Part of a config resource showing network information.

NOV 2014: SONY ATTACK DESTOVER BACKDOOR SAMPLES ARE BASED ON KORDLLBOT

The Destover “lightweight backdoor” (sha256 4c2efe2f1253b94f16a1cab032f36c7883e4f6c8d9fc17d0ee553b5afb16330c) mentioned in official statements related to the Sony intrusion is a digitally signed file. There is also an almost identical unsigned file in existence with the sha256 eff542ac8e37db48821cb4e5a7d95c044fff27557763de3a891b40ebeb52cc55. This unsigned file is the original. It was established that the signed file was created as a “joke” by a researcher (4).

We were able to locate more malware samples similar to this backdoor. Many of these were created in a timeframe well before the Sony intrusion came to light. Some also match the *import hash* indicators mentioned in the US-CERT advisory, though import hashes are non-unique indicators and cannot always be relied upon.

Closer investigation reveals that this Destover sample is indeed derived from the same source base as KorDllbot. This is based on the following indicators:

- The Chopstring API string obfuscation
- The CMXE command line construction
- Same way of declaring API’s
- Similarities with later samples, such as:
    - A printf “MessageThread” statement in the beginning of the command handling function (similar to Destover “MessageThread” samples)
    - Use of the XOR-A7 encoding to decode strings (similar to Destover “b076e058” samples)

Throughout 2014 and 2015 and still ongoing in 2016, Destover-related backdoors have continued to be used in various campaigns. They share many common traits, but there are also clear differences in functionality, hinting at a common source repository but where customization is added as needed. Some subfamilies have received their own variant names – i.e. Volgmer and Duuzer – while others have no separate moniker. See appendix for detailed descriptions of variants.

OTHER POSSIBLY RELATED MALWARE ACTIVITY

A number of incidents and malware systems have been attributed to either the DarkSeoul group or North Korean threat actors. This chapter will quickly go through some of these.

THE CASTOV AND CASTDOS CAMPAIGNS (AKA 6.25 DDOS ATTACKS)

The Castov campaign mainly targeted South Korean financial corporations and was discovered in May 2013 (16). Notably, these malwares included code to steal banking credentials.

Some were designed to perform DDOS attacks on Korean government servers on June 25th, 2013 (16) (12) – the same date that the destructive Korhigh malware was also uncovered - though we have no information as to whether these cases were connected.

On the face of it, there is little to directly connect the Castov malware with the DarkSeoul/Destover complex, as the codebase is largely different. For example, the initial downloader was a crimeware known as Tijcont, distributed by the Gongda exploit kit. The downloaded banking malware was written in Delphi, uncommon for DarkSeoul projects.

However, Symantec states clearly in their blog post that they attribute Castov to the DarkSeoul group.

THE KIMSUKY SYSTEM

The Kimsuky malware complex was originally detailed in a report from Kaspersky (14) in 2013 and has been an active component of the South Korean threat landscape since then. Ahnlab reported a new campaign in Feb 2014 (15), and an intrusion attempt into South Korean nuclear facilities in Dec 2014 was also identified to involve Kimsuky (16).

The Kimsuky malware is different in structure from the Destover complex. It uses different encoding schemes and algorithms than Destover, and email and FTP is used for C&C communication and exfiltration.

Similar to Destover, Kimsuky has used HWP exploits as infection vector. A number of samples rely on vulnerabilities in the old OLE2-based HWP file format. However, they have not, as far as we have seen, used the recent CVE-2015-6585 HWPX vulnerability which has been used to plant at least three variants of Destover.

There are some similarities in modus operandi, such as

- Encoded API usage.
- Frequent code hand-modifications between samples
- Malware installed as services
- Taunting the victim in public fora
- Posing as hacktivist groups (17)
- Publication of stolen data (17)

Based on the available data we cannot say that the Kimsuky-based campaigns are connected to the DarkSeoul group.

THE BLACKMINE SYSTEM

Blackmine is a South Korean focused malware campaign detailed by Ahnlab (18).

The payload malware in question is a data harvester and uploader, which also allows for download of more malware. In the same way as Kimsuky, there are some similar approaches with Destover – the usage of obfuscated API names for example – but also enough differences to say that Blackmine probably has not originated from the same codebase. Ahnlab does however state that they see these groups as possibly correlated.

CONCLUSION

The attack on Sony Pictures Entertainment incorporated the use of malware which contained a number of commonalities with malware used in previously known attacks.

These previous attacks were mainly focused against South Korean entities such as financial institutions, government sites, think tanks and other important functions. Targets outside South Korea have also been affected, albeit to a lesser extent: Apart from the Sony intrusion, the Dozer DDOS attacks of 2009 were also directed towards US websites.

The amount of common factors between the different incidents makes it in our opinion very likely that these incidents are perpetrated by the same group, or at least cooperating groups.

In this paper, we are not commenting on geographical attribution for the Sony attack. We note that a number of the mentioned previous attacks (Dozer (15), Koredos, Korhigh (16), DarkSeoul (17)) have been associated with North Korean involvement, but these associations have not been examined or validated by us.

It is worth noting that this threat actor is still active. We have seen Destover-samples compiled as recently as January 2016. DarkSeoul should be considered a constant risk factor, particularly for South Korean institutions.

The Destover malware family seems to be the information gathering workhorse of this group – adapted and changed to fit the purpose *du jour*, but retaining a lot of the same overall design and methodology. For specific targets more customized malware is often deployed.

Command and control connections are almost always going to raw IP addresses, and different malware generations tend to use different sets of addresses. It is our assumption that most of these IP's are compromised computers which probably are running proxies, and as such are easily disposable.

WORKS CITED

1. **FBI.** FBI Liaison Alert System #A-000044-mw. [Online] https://publicintelligence.net/fbi-korean-malware/.

2. **US-CERT.** Alert (TA14-353A) Targeted Destructive Malware. [Online] https://www.us-cert.gov/ncas/alerts/TA14-353A.

3. **Symantec.** Duuzer back door Trojan targets South Korea to take over computers. [Online] http://www.symantec.com/connect/blogs/duuzer-back-door-trojan-targets-south-korea-take-over-computers.

4. **Ullrich, Johannes B.** Malware Signed With Valid SONY Certificate. [Online] https://isc.sans.edu/forums/diary/Malware+Signed+With+Valid+SONY+Certificate+Update+This+was+a+Joke/19049/.

5. **Sherstobitoff , Ryan, Liba, Itai and Walter, James.** Dissecting Operation Troy: Cyberespionage in South Korea. [Online] http://www.mcafee.com/us/resources/white-papers/wp-dissecting-operation-troy.pdf.

6. **Jiang, Genwei and Kimble, Josiah.** Hangul Word Processor (HWP) Zero-Day. [Online] https://www.fireeye.com/content/dam/fireeye-www/global/en/blog/threat-research/FireEye_HWP_ZeroDay.pdf.

7. **SecureSoft.** 7.7 DDoS 攻撃分析及び対応策. [Online] https://www.securesoft.co.jp/news_mt/docs/7.7DDOS_2.pdf.

8. **McAfee, Inc.** Ten Days of Rain. [Online] http://www.mcafee.com/us/resources/white-papers/wp-10-days-of-rain.pdf.

9. **Lelli, Andrea.** Backdoor.Prioxer!inf: "Accidentally" the Stealthiest File Infector Ever! [Online] http://www.symantec.com/connect/blogs/backdoorprioxerinf-accidentally-stealthiest-file-infector-ever.

10. **Symantec.** Four Years of DarkSeoul Cyberattacks Against South Korea Continue on Anniversary of Korean War. [Online] http://www.symantec.com/connect/blogs/four-years-darkseoul-cyberattacks-against-south-korea-continue-anniversary-korean-war.

11. **Korea Joongang Daily.** JoongAng hit by major cyberattack. [Online] http://koreajoongangdaily.joins.com/news/article/article.aspx?aid=2954219.

12. **Kwaak, Jeyup S.** Sony Hack Mirrors Attack on South Korean Newspaper, Researcher Says. [Online] http://blogs.wsj.com/korearealtime/2014/12/19/sony-hack-mirrors-attack-on-south-korean-newspaper-researcher-says/.

13. **Boannews.** 소니픽처스 vs. 6.25 사이버테러 악성코드 전격 비교분석. [Online] http://www.boannews.com/media/view.asp?idx=44451.

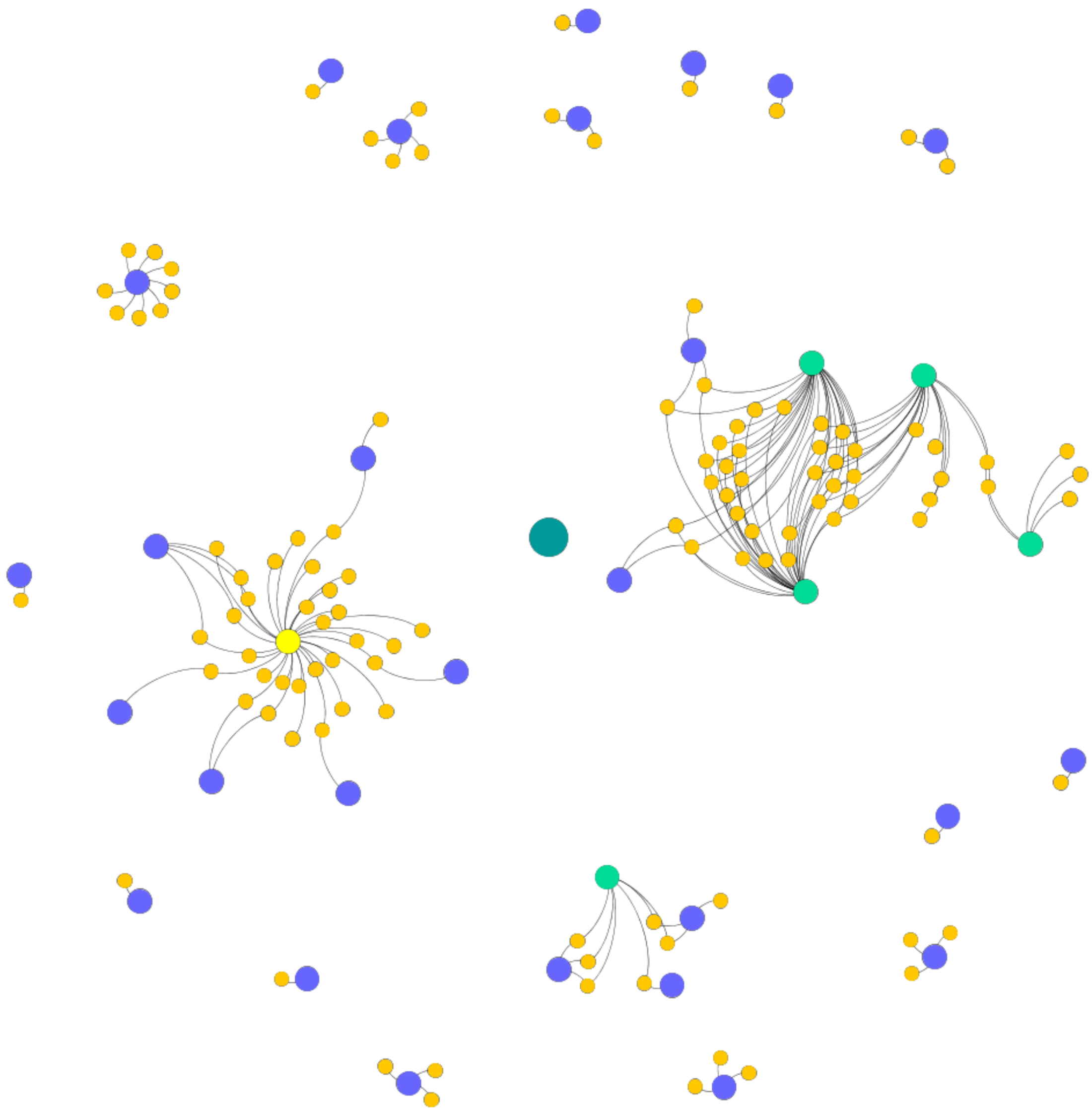
14. **Tarakanov, Dmitry.** The "Kimsuky" Operation: A North Korean APT? [Online] https://securelist.com/analysis/publications/57915/the-kimsuky-operation-a-north-korean-apt/.

15. **Fox News.** S. Korea Analyzes Computers Used in Cyberattacks. [Online] http://www.foxnews.com/story/2009/07/12/s-korea-analyzes-computers-used-in-cyberattacks.html.

16. **v3.co.uk.** South Korea blames cyber attacks on North Korean government hackers. [Online] http://www.v3.co.uk/v3-uk/news/2282616/south-korea-blames-cyber-attacks-on-north-korean-government-hackers.

17. **Chosun Ilbo.** Evidence in Hacker Attack Points to N.Korea. [Online] http://english.chosun.com/site/data/html_dir/2013/04/11/2013041100648.html.

APPENDIX: TECHNICAL DETAILS



Note: Data used for this report has solely come from public or otherwise unrestricted sources.

THE JOANAP FAMILY

JOANAP.A BACKDOOR, JAN 2009

The first version of what could be called a Joanap-related malware was a series apparently compiled January 16th-January 19th 2009. This is actually not a worm at all, as there is no code for network propagation present. Instead, it is a data harvester and backdoor which bears some similarity with KorDllbots – API's are dynamically declared, harvested data is added to ZIP file before exfiltration, and the command structure uses a set of integers (0x1010 – 0x1020).

As previously mentioned, the Joanap malware series contains code snippets from publicly available Rbot code (25). This includes an implementation of the Tiny Encryption Algorithm (TEA) which has been somewhat modified, as well as the Rbot PLAIN_CRYPT algorithm. The default key used in the PLAIN_CRYPT public Rbot source is the string "9024jhdho39ehe2". This key is used if there is no other key passed to the algorithm.

However, this backdoor uses the same default key as later Joanap variants - "902**5**jhdho39ehe2", a one-byte change quite specific to this malware series.

Joanap.A also uses a *custom* key which is used both in the PLAIN_CRYPT algorithm (for string decryption) and in the TEA algorithm (for data file encryption/decryption). This is the string "*hybrid!@hybrid!@#*" – which is visible in cleartext inside the executable.

JOANAP.B WORM, OCT 2009

This malware is significantly different from the A version. The main similarity between them is the use of the Rbot PLAIN_CRYPT algorithm for string decryption with the mentioned "9025jhdho39ehe2" default key. The custom key used is now changed to "*iamsorry!@1234567*".

The executable contains two XOR-encrypted objects in its resource section. One is a dictionary file containing passwords, stored in resource 101. The other, stored in resource 103, is an executable – a copy of the legitimate PsExec tool from SysInternals.

Contrary to the A version, this variant is a true worm. It generates random IP addresses and attempts to connect to these over the SMB port 445/tcp. It uses the WNetAddConnection2A API to map the remote machine as a share, using its dictionary of passwords. If this works, it will copy itself to the system folder of the remote server, and extract its embedded PsExec application to execute the file remotely.

The malware does not connect directly to a C&C server. Instead it sends status mails to its controller via GMail's public mail server *gmail-smtp-in.l.google.com.* The email will appear to be sent FROM *ninja@gmail.com* TO *xiake722@gmail.com.* Content is all in the subject field – initially only version (1.1), time, and local IP address. Upon successful connection and copy to a remote machine, the malware sends mail again – this time also containing remote IP, username and password, in addition to its initial fields.

Stream Content

```
220 mx.google.com ESMTP t10si3662438lat.87 - gsmtp
HELO www.hotmail.com
250 mx.google.com at your service
MAIL FROM: <ninja@gmail.com>
250 2.1.0 OK t10si3662438lat.87 - gsmtp
RCPT TO: <xiake722@gmail.com>
250 2.1.5 OK t10si3662438lat.87 - gsmtp
DATA
354  Go ahead t10si3662438lat.87 - gsmtp
FROM:  <ninja@gmail.com>
TO: Joana <xiake722@gmail.com>
SUBJECT: [T].1.1.20150124004616.[redacted]


.
250 2.0.0 OK 1422175386 t10si3662438lat.87 - gsmtp
QUIT
221 2.0.0 closing connection t10si3662438lat.87 - gsmtp
```

Above: Email transfer between Joanap and the mail server.

A minor sub-variant of this Joanap generation exists. This sends email just the same way as described above, but uses a different TO address (*laohu1985@gmail.com*) during network propagation.

JOANAP.B DOWNLOADED BACKDOOR, SEP 2009

However, spreading is not the main payload of the B version of Joanap. Instead, it attempts to download and install a second stage malware. This malware, with the sha256 hash of *c6d96be46ce3d616e0cb36d53c4fade7e954e74bfd2e34f9f15c4df58fc732d2*, was hosted on the *URL hxxp://www.booklist.co.kr/upload/img/200810/25.gif*. It would be downloaded and saved to disk under the name *sysfault.exe* and executed.

This malware is an installer, installing a service dll in the system folder under the name "*sdnssec.dll*". This is a listen-only backdoor, establishing a listening socket on port 136.

Similar to the Joanap.A variant and other KorDllbot-related backdoors, this supports a number of integer commands. The binary contains quite a lot of debug messages helpfully explaining the functionality of these.

Command	Function
0x1010	List drives
0x1011	File browse
0x1012	File copy
0x1013	File delete
0x1014	File upload (to target)
0x1015	File download (to botmaster)
0x1016	Execute file
0x1017	Change filetime
0x1018	Folder download (to botmaster)
0x1019	Test connect
0x1020	Run shell command
0x1021	Sleep
0x1023	File properties
0x1030	Process view
0x1031	Process kill
0x1032	Process kill by name
0x10FF	Uninstall

JOANAP.C BACKDOOR, JUL 2010

The installer of Joanap.D (next entry) also actively deletes installed files named *signtc.ax*, *signtm.ax*, or *signts.ax*. Searching for these brought up an apparently preceding sample which uses one of these files - *signtc.ax* - for storing data. This sample appears to belong to a series of previous backdoors somewhat related to KorDllbot – example SHA-256 hash is 4b6078e3fa321b16e94131e6859bfca4503bcb440e087d5ae0f9c87f1c77b421.

We have not analyzed this variant in detail.

JOANAP.D BACKDOOR, JUL 2011

This malware arrives as a service installer which extracts and installs a DLL named *scardprv.dll* from its resource section, and writes hardcoded configuration data to a config file named *mssscardprv.ax*. It also attempts to delete files installed by previous Joanap versions.

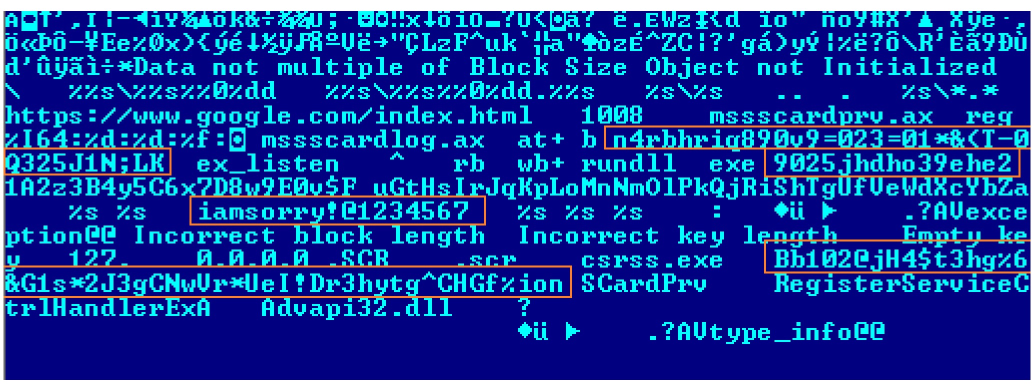
The dropped service DLL has similarities with KorDllbots. It establishes a listening socket on a semi-random port which is either located between 1024 and 2048; or selected from a list of hardcoded port options. It also attempts to connect to C&C servers which are defined in the saved mssscardprv.ax file as raw IP address/port combinations.

All network traffic is encrypted using RC4 with the binary key (0x10,0x20,0x30,0x40,0x50,0x60,0x70,0x80,0x90,0x11,0x12,0x13,0x1A,0xFF,0xEE,0x48), and the backdoor accepts integer commands in the range 0x4001-0x4015.

API strings reside in data blocks encrypted using AES. Network API's are encrypted with the key *"b n4rbhriq890v9=023=01*&(T-0Q325J1N;LK'"*, while all others are encrypted with the key *"Bb102@jH4$t3hg%6&G1s*2J3gCNwVr*UeI!Dr3hytg^CHGf%ion"*. This particular AES key was also found in both Joanap and KorDllbot malware belonging to the previously mentioned MicrosoftCodeSigningPCA certificate cluster.

In addition, this variant includes the Rbot PLAIN_CRYPT decryption keys "*9025jhdho39ehe2*" and "*iamsorry!@1234567*" for one specific decryption scenario. So, even though it is somewhat different from previous variants, it contains enough technical indicators to link it to the Joanap family.

The samples we have seen do not appear to have network spreader capability, though they may have been dropped by other malware.



Above: Indicators in the binary

JOANAP.E WORM, AUG-SEP 2011

Joanap.E was the first variant of this family we tied to this threat complex, due to the fact that several samples are signed using the peculiar MicrosoftCodeSigningPCA certificate format.

This variant is again a worm – as mentioned before, the installer drops three files – one SMB spreading DLL (wmmvsvc.dll), one backdoor DLL (scardprv.dll) and one configuration file (mssscardprv.ax). The backdoor DLL and the configuration file fill the same role as in Joanap.D.

The network spreader module contains some code from the B variant, but a lot of functionality has been reworked. Similarly to B, it generates semi-random IP addresses and attempts to logon to the admin account of these machines using a password dictionary. If it manages to do this, it creates a remote share named "$adnim" (no typo), copies the main installer (and the configuration file) over, and executes it. The authors have moved away from using PsExec for remote execution. Instead they add shares and execute the worm by creating remote service commands via the Service Control Manager.

If this is successful, the worm sends a status mail the same way as the B variant. Mail is this time FROM: *redhat@gmail.com* TO: Joana <*misswang8107@gmail.com*>*.*

This malware uses the same encryption keys as the B variant. This worm sets the mutex "**PlatFormSDK2.1**".

JOANAP.F WORM, MAR 2012

We have only two slightly different samples of this generation. Again, the malware's structure has changed. It is no longer a service DLL, but instead a standalone Windows executable. Contrary to previous versions, this worm requires being started with at least one command line parameter (either –i or -s), if not it just exits.

The –s parameter starts the spreading routine if it is installed correctly and it can find its configuration files. The samples we have come without installer or data files and do not run.

There is no doubt that these samples belong to this malware family – they use the same encryption keys, mutex structures and data file names as the E variant in the series. There is one notable exception: This is the first time we see the file encryption RC4 key "*y0uar3@s!11yid!07,ou74n60u7f001*", which closely matches the key mentioned as belonging to the "SMB Word Tool" in the US-CERT advisory (2) after the Sony incident, *"y0uar3@s!llyid!07,ou74n60u7f001"*. The difference might be due to a typo. The malware appears not to be identical though, as some other strings from the advisory YARA rule are not present.

This worm sets the mutex "**PlatFormSDK2**".

JOANAP.G WORM, OCT 2014

This Joanap variation uses the mutex "**Global\FwtSqmSession106829323_S-1-5-19",** which also matches data from the US-CERT advisory (2). However, this time the worm has switched to a different RC4 key - "y@s!11yid60u7f!07ou74n001". This variation has been detailed by researchers from PriceWaterhouseCoopers (4).

JOANAP.H WORMS, OCT 2014-JAN 2015

This is a series on Joanap executables produced towards the end of 2014 and beginning of 2015. They use the mutex "**Global\FwtSqmSession106839323_S-1-5-20**", but the same RC4 key as the G variants.

Some samples are quite a lot larger than normal on account of including a big chunk of code from the open source FreeRDP remote desktop client. Apart from this we have not analyzed these samples in detail.

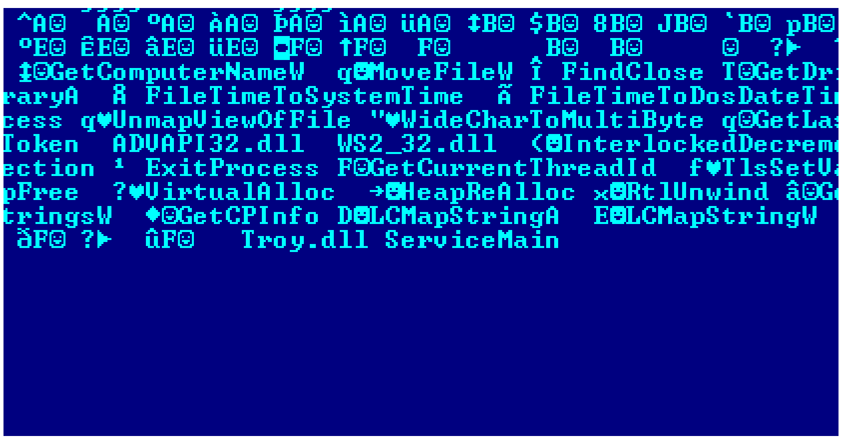
THE DESTOVER FAMILY

DESTOVER “B076E058” BACKDOORS, FEB-JUNE 2014.

This sub variant has been named “b076e058” based on the first portion of the RSA authentication key used for its server handshake.

Most samples share the ChopString and XOR-A7 obfuscation functions with the Sony-associated malware eff542ac8e37db48821cb4e5a7d95c044fff27557763de3a891b40ebeb52cc55. They also declare API calls in the same way.

Samples of this variant were all compiled with the library name *“Troy.dll”* in the Export Table, similar to what McAfee documented in their “Operation Troy” paper (5) on destructive attacks against South Korean targets.



Troy.dll visible in 10d3ab45077f01675a814b189d0ac8a157be5d9f1805caa2c707eecbb2cbf9ac

This variant is typically installed as service, with one export - “ServiceMain”. Its main purpose is to listen on a given port and accept commands. The integer codes used for these commands are:
A variant: 0x54b7- 0x54cb, with the exception of 0x54be and 0x54ca.
B variant: 0x54b7- 0x54cb, with the exception of 0x54be and 0x54ca, and the addition of 0x54d0.

The installation is done by unobfuscated dropper executables, which install the service DLLs after performing some systems checks.

DESTOVER "VOLGMER" BACKDOORS, MAR-SEPT 2014

Volgmer backdoors were quickly connected to the Sony case, since several samples use a C&C IP address (200.87.126.116) in common with the Sony malware droppers. The family is easily recognized by the peculiar UserAgent strings used, which all start with "Mozillar/" instead of "Mozilla/."

These backdoors come in three flavors (that we've found).
The first batch was apparently compiled March 15, 2014. These appear to be prototypes for later versions, and helpfully contain debug strings labeling all major functionality. We have only DLL samples of this variant.

The second batch was apparently compiled in April 2014. The droppers contain a service DLL and a configuration file in a password-protected zip archive embedded as a resource in the dropper executable. The dropper needs to be able to extract these files, so it also contains the password - which in this case is "!1234567890 dghtdhtrhgfjnui$%^^&fdt."

The third batch was apparently compiled in June and July 2014. These droppers contain a regular Win32 executable where the configuration data is contained in the exe. The dropped executable checks the current locale and will not run unless this contains the string "korea."

Each dropper package comes configured with partially different C&C information. True to the standard modus operandi of this group, all C&C servers are defined as raw IP addresses, typically located on ports in the 8000-range, such as 8080, 8088 or 8888.

```
0000000000: 63 67 69 5F 63 6F 6E 66 | 69 67 00 00 00 00 00 00  cgi_config
0000000010: 00 00 67 10 DF 23 90 1F | 00 00 71 1C F4 C2 90 1F    g►ß#?▼  q∟ôÂ?▼
0000000020: 00 00 74 30 91 B3 90 1F | 00 00 BA 74 09 14 40 1F    t0'³?▼  ºt○¶@▼
0000000030: 00 00 BA 95 C6 AC 90 1F | 00 00 BE D2 27 A9 90 1F    º•Æ¬?▼  ¾Ò'©?▼
0000000040: 00 00 C3 1C 5B E8 98 1F | 00 00 C7 0F EA 78 90 1F    Ã∟[è˜▼  Ç☼êx?▼
0000000050: 00 00 C8 2A 45 85 90 1F | 00 00 CB 83 DE 63 90 1F    È*E.?▼  ËƒÞc?▼
0000000060: 00 00 D4 21 C8 56 90 1F | 00 00 D5 CF 8E 52 B8 22    Ô!ÈV?▼  ÕÏZR¸"
0000000070: 00 00 53 E7 CC 9D 98 1F | 00 00 54 E8 E0 DA 98 1F    SçÌ?˜▼  TèàÚ˜▼
0000000080: 00 00 59 7A 79 E6 90 1F | 00 00 59 BE BC 2A 90 1F    Yzyæ?▼  Y¾¼*?▼
0000000090: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
00000000A0: 00 00 00 00 00 00 00 00 | 00 00 00 00 00 00 00 00
```

Configuration file from the first batch of Volgmer droppers - after the cgi_config marker follow IP/port pairs.

Main functionality involves gathering system information and uploading this to the two main C&C servers in an encoded ZIP-archived format. They accept commands in the range 0x1000-0x1008 (A) and 0x1000-0x1012 (B/C).

DESTOVER “WINDOWSUPDATETRACING” BACKDOORS, SEPT-OCT 2014

This malware is somewhat different in design than previously mentioned variants. The installer package installs the backdoor along with legitimate packet filtering components, and there is code to steal credentials from a great deal of different products, some of which are Korean. One interesting feature with this malware is that it has some limited support for other languages - it contains some user folder names in ex. Spanish and Portuguese in addition to English. The name “WindowsUpdateTracing” is derived from a mutex created by this variant – typically this will be “*WindowsUpdateTracing0.5*” but the suffixes “0.6” and “0.7” also exist. Chopstring API obfuscation is also present.

Command integers are in the range 0x58692ab8-0x58692ac0.

This trojan uses a semi-traditional Command and Control model, with connections seemingly going to a number of DynDNS domains that are defined in an accompanying configuration file named *msxml15.xml.* This configuration file is encrypted using RC4; typically with the RC4 key “*BAISEO%$2fas9vQsfvx%$”* though some samples use the API name “GetFileAttributesW” as key – possibly a bug.

Known C2 domains:

iphoneserver.lflink.com
dns05.mefound.com
mx1.mefound.com
dns01.vizvaz.com
myserver.mrbonus.com
game.dnsrd.com
dns01.zzux.com
exchange01.toh.info
exchange04.yourtrap.com

However, the DNS resolution for these domains is misleading. The IP address returned by the DNS server will be XOR’ed with a 32-bit key (we have seen two different keys, depending on variant type), which yields the correct C2 IP address to use. This means that relying on DNS resolution to identify C&C hosts will not work.

```
push    1                   ; _DWORD
push    edx                 ; _DWORD
mov     [esp+24h+cp], 0
call    DnsQuery_A
test    eax, eax
pop     esi
jz      short loc_10005FBD
push    eax                 ; ArgList
push    offset aDnsqueryFailed ; "DnsQuery() Failed. [%d]"
call    sub_10002880
mov     eax, [esp+28h+var_C]
add     esp, 8
mov     dword ptr [eax], 0
xor     eax, eax
add     esp, 8
retn
---------------------------------------------------------------------------

_10005FBD:                          ; CODE XREF: sub_10005F40+5D↑j
mov     eax, [esp+20h+var_10]
mov     edx, [esp+20h+var_C]
push    1                   ; _DWORD
push    eax                 ; _DWORD
mov     ecx, [eax+18h]
xor     ecx, 1AB9C2D8h
mov     [edx], ecx
mov     ecx, DnsFree
test    ecx, ecx
jz      short loc_10005FE8
```

IP longint returned in the DNS response is XOR'ed with a dword integer.

This bogus DNS response can be used in an interesting fashion. The domain *mx1.mefound.com* has resolved to the bogus IP 44.58.156.86. When this IP is converted using the corresponding XOR key 0x579C3A53 it becomes 127.0.0.1 – i.e. localhost. Presumably this is done when the bot is not active. The IP 44.58.156.86 belongs to University of California at San Diego (UCSD) and have as far as I can tell never been used to host any publicly available domain. Still, passive DNS data shows that this IP has been the DNS response of a number of DynDNS domains; many of which we had not seen before. We may thus assume that these domains are used in backdoors containing the same XOR key as this particular Destover sample. This applies to the following additional domains:

update03.compress.to
baid.otzo.com
mx2.mefound.com
facebok.mrbasic.com
report01.onedumb.com
appinfo.yourtrap.com
gupdate.yourtrap.com
status01.instanthq.com
eschool.toythieves.com
gogle.jungleheart.com
mycompany.moneyhome.biz

Since we know the XOR key used, we can also translate any *other* IP's associated with these domains to presumably correct C&C IP addresses (see appendix). If we repeat this process with the other XOR key we know of - 0x1AB9C2D8 - we end up with the localhost IP 127.0.0.1 translating to the bogus IP of 167.194.185.27. No additional data was found at this time using this method, but any DynDNS domain resolving to this IP in the future might be interesting to look at.

DESTOVER “MESSAGETHREAD” BACKDOORS, MAY 2014-MAR 2015

These Destover backdoors contain the Chopstring obfuscation, as well as XOR-A7 encoding.
They are straight remote control tools of the basic KorDllBot model. The name stems from the Unicode string “MessageThread” present in all samples of this type. The Sony Destover sample belonged to this variation.

The command integers used by this variant are typically in the range 0x523b-0x5249.

Unlike many other Destover trojans, some of these installers come with embedded decoy documents, hinting at intended target audience. The decoys are all in Korean language – one document lists telephone numbers belonging to personnel in government and other public functions; other samples contain an invitation to the Korean Government 3.0 expo that was to be held in in Seoul.

Gov 3.0 expo invitation

DESTOVER “B8AC0905” BACKDOOR, MAR 2015

We have only a single sample of this variant. The name b8ac0905 is derived from the authentication key string contained in the file (See appendix). The API obfuscation is here done via an encoding scheme which appears unique, but bears some similarity with RC4. We call this encoding “Intbox” as the S-Box is not populated using a string as input, but instead is a function of an integer key.

This is a “listen only” backdoor, and does not call out to any C&C server directly. We do not have the configuration data that presumably was installed along with this sample, so no more details are available at this time.

The integer commands it expects are 0x00-0x0f, 0x12 and 0x15.

DESTOVER “B59D1659” BACKDOOR, APR 2015

We have only one sample of this variant too – a Win64 DLL exporting the functions *ServiceMain*, *RasmanStart* and *RasManEnd*.Of these, only ServiceMain has any real function. The sample attempts to impersonate the legitimate appmgmts.dll from X64 Windows 7. It is even of the exact same size as the original. The name b59d1659 is derived from the RSA authentication key string contained in the file (see appendix).

The command words used by this variant are in the range 0x2638000-x236801b.

The C&C configuration is read from a data flle - appmgmts.rs - which presumably is created by the installer, and which we do not have a copy of. Thus, C&C information and distribution method is unknown for this variant.

DESTOVER “RANDOMDOMAIN” BACKDOORS, MAR-APR 2015. VERSION C JAN 2016

Destover “Randomdomain” backdoors have also evolved from the original KorDllbots. They come in both x86 and x64 versions.

There seems to be three distinct variants of this class of backdoors with slightly different obfuscation methods used and C&C configuration, though most variants use the same API obfuscation – an inline character replacement technique resulting in *almost* recognizable API strings in the file. We name this technique “*CharSwap*” for the purpose of this paper.

They connect to their C&C servers using *what appears to be* SSL/TLS. This includes a remote server name indication (SNI) extension in the initial Client Hello. This server name is randomly picked from an internal list of domain names – thus the name “Randomdomain.” A list of such names can be found in the appendix. When I say “appears to be” SSL/TLS, this is because the encryption actually used is not secure. The malware can choose between different simple encryption modi, and these are somewhat different between the known variants.

Variant A uses either RC4 with the string “TCPPROCESSREADY.” as encryption key, or a XOR 0x28, SUB 0x28 encoding, or a segmented XOR encoding . Variant B uses either simple byte wise XOR encoding with a shifting key, or an even simpler XOR 0x25, SUB 0x25 encoding. Variant C uses only one – the same shifting XOR encoding used by variant B.

Variant C checks auto proxy settings and will connect through the configured proxy if possible. This code is not seen in earlier versions.

The command words used by these backdoors are in the range 0x123459 - 0x12348a (some files to 0x123488).

The two first variants were apparently in use in the first half of 2015. Variant C has been used more recently – we have seen only two samples, the first date stamped May 2015, the last Jan 12th, 2016.

DESTOVER “DUUZER” BACKDOORS, MAR-OCT 2015 , JAN 2016

The Duuzer variation of Destover backdoors have evolved quite a bit from the original KorDllbot basis. They use more in-code obfuscation and are somewhat more complex. For example, string references are stored as encoded local variables in special functions. Access to these variables is obtained by calling the containing function with an offset into the variable blob, and the function decodes the correct string.

Similar to the “RandomDomain” and “e4004c1f” these backdoors use specially crafted SSL headers to initiate communication with their C&C servers, but the encryption is custom. The command scheme is also somewhat unique – instead of a digit to indicate which function to perform, these backdoors use binary multibyte command statements.

There are several sub variants of Duuzer. One sample . (sha256 f31d6feacf2ecece13696dcc2da15d15d29028822011b45045f9efa8a0522098) appears to be a predecessor and somewhat simpler than later samples. Later variants include the “live” and the “naver” versions - based on the server name they use in their faked SSL handshake, either “login.live.com” or “ad.naver.com”. The latest versions we have seen – compiled January 2016 – don’t even bother with these strings.

As previously mentioned, Duuzer has been detailed in a report from Symantec (3). This report also mentions the connection to the Joanap malware family, and details examples of live usage of the “CMXE” command line execution mentioned before.

This variant has been seen as the payload of trojanized HWPX documents exploiting the CVE-2015-6585 vulnerability as documented by FireEye (6). Decoy documents include invitations to events like Korean *Aerospace Systems Engineering 2015*, and *Aeroseminar 2015*; a Korean Aerospace Weapon System Development Seminar (below). An email found on VirusTotal shows that an exploited document containing this exact decoy was attempted sent to the Korean Atomic Energy Research Institute (KAERI).

김종현 선임 042) 821-3376

준비위원장 조창민 부장 042) 821-3322, 간사

34186 대전 유성우체국 사서함 35-7호
E-mail: aeroseminar2015@add.re.kr

준비위원장 대령 최상욱 042) 552-5500, 간사 소령 이성남 042) 552-5525

32800 충남 계룡시 계룡대로 663 사서함 501-317호
E-mail: l1s2n3@gmail.com

대한민국공군 연구분석 평가단

등록안내

-사전등록시만 명찰, 논문집, 기념품, 식권을 제공합니다.
-사전등록 마감 2015년 10월 16일
-참가신청
군)11s2n3@gmail.com
국과연)aeroseminar2015@add.re.kr
상기메일로 성명,주소,근무처,직책, 연락처를작성하시어 송부하여 주십시오.

• 찾아오시는 길

자가차량이용

- 경부고속도로 이용시
판교 IC → 시흥사거리 → 한성대 체력단련장 코스 게이트 → 전시장 20분
- 서울 외곽순환도로 이용시
성남IC, 성남시청 방향 진행 → 여수삼거리 우회전 → 시흥사거리 우회전
→ 한성대 체력단련장 코스 게이트 → 전시장, 15분
네비게이션은 '한성대 체력단련장' 또는 한성대 골프연습장 지정

대중교통이용

-지하철
태평역/모란역 하차 후 셔틀버스 이용, 도보 13분
-버스
태평역 정류장 버스 하차 후 셔틀버스 이용, 도보 13분
3-1, 30-1, 82, 200,220,240,462,4419
여수대교 정류장 버스하차, 도보 20분
-셔틀버스 운행 : 전시장 → 모란역 → 태평역 → 전시장(순환), 배차간격 5분 단위(07:30 ~ 18:00)

2015

항공우주 무기체계 발전세미나

창조국방을 선도하는
항공우주무기체계 발전 방향

미래 전장을 고려한
유・무인 항공기
개발 방향

일자 2015년 10월 21일(수)

DESTOVER “E4004C1F” BACKDOOR, JUL-SEP 2015

The main differences in this backdoor arise from the inclusion of what appears to be modified open source SSL/TLS code. This is used to construct legitimate SSL headers, though the communication itself is encrypted by a homegrown encoding scheme. This backdoor is found in both x86 and x64 variants.

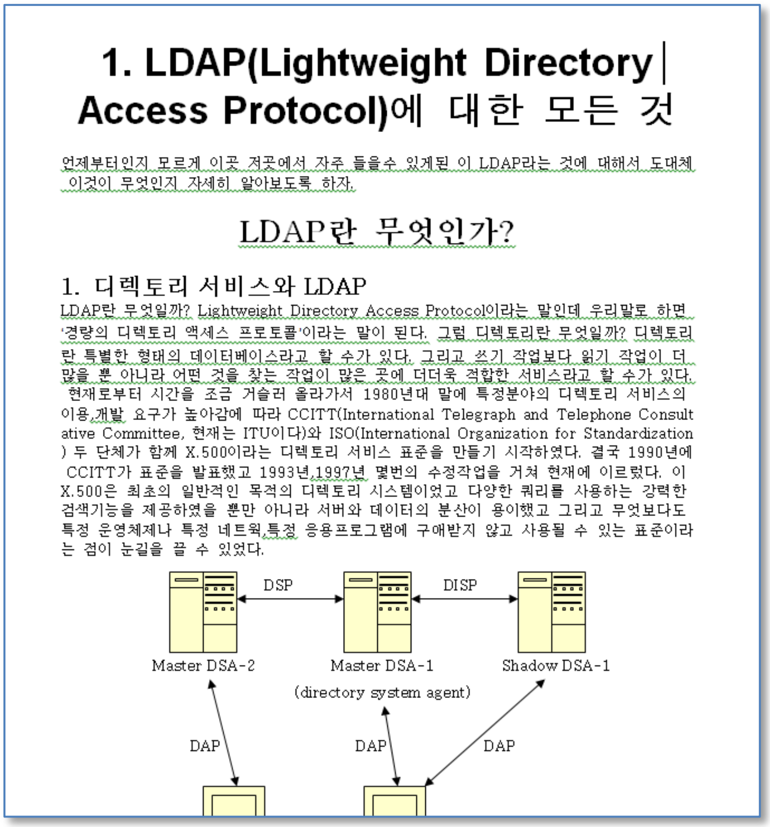
The name e4004c1f is taken from the start of the authentication key found in all these samples.

The command integers vary somewhat between sub variants:

Variant A samples use the range 0x00-0x0f, with *addition* of bytes 0x12, 0x1b, and 0x64.
Variant B samples use the range 0x0a-0x24, with *exception* of bytes 0x18, 0x1c, and 0x1d
Variant C samples use the range 0x0a-0x26, with *exception* of bytes 0x18, 0x1c, and 0x1d

This family has also been used as the payload of CVE-2015-6585 trojanized HWP documents. The FireEye write-up on this mentions a backdoor they name HANGMAN (7). FireEye uses a proprietary malware naming scheme which makes it somewhat difficult to correlate, but we believe this corresponds to the “e4004c1f” variant. In the same blog post FireEye mentions a backdoor they call PEACHPIT. Based on the code snippet shown, we believe PEACHPIT to belong to one of the early KorDllbot generations. As mentioned, the exact same CMXE code has been used in several generations from 2011 and onwards.

Decoy documents used by “e4004c1f” include descriptions of the LDAP protocol, and a text on the virtues of Scrum vs Kanban. The latter was attempted sent to the Korean Google group “*sysadminstudy*”. It is possible that this generation of malware has been aimed at the IT/software industry.



1. LDAP(Lightweight Directory Access Protocol)에 대한 모든 것

언제부터인지 모르게 이곳 저곳에서 자주 들을수 있게된 이 LDAP라는 것에 대해서 도대체 이것이 무엇인지 자세히 알아보도록 하자.

LDAP란 무엇인가?

1. 디렉토리 서비스와 LDAP

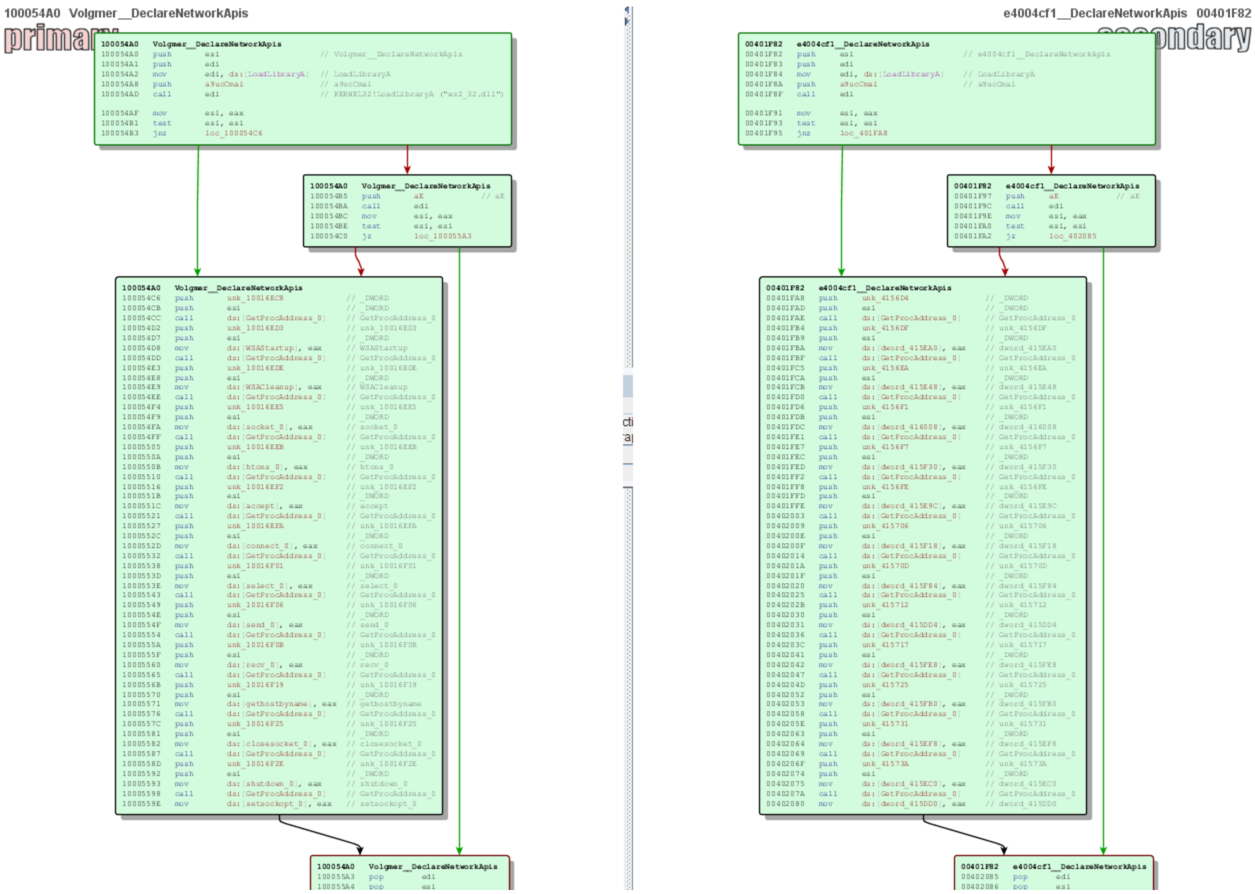
LDAP란 무엇일까? Lightweight Directory Access Protocol이라는 말인데 우리말로 하면 '경량의 디렉토리 액세스 프로토콜'이라는 말이 된다. 그럼 디렉토리란 무엇일까? 디렉토리란 특별한 형태의 데이터베이스라고 할 수가 있다. 그리고 쓰기 작업보다 읽기 작업이 더 많을 뿐 아니라 어떤 것을 찾는 작업이 많은 곳에 더더욱 적합한 서비스라고 할 수가 있다.
현재로부터 시간을 조금 거슬러 올라가서 1980년대 말에 특정분야의 디렉토리 서비스의 이용,개발 요구가 높아감에 따라 CCITT(International Telegraph and Telephone Consultative Committee, 현재는 ITU이다)와 ISO(International Organization for Standardization) 두 단체가 함께 X.500이라는 디렉토리 서비스 표준을 만들기 시작하였다. 결국 1990년에 CCITT가 표준을 발표했고 1993년,1997년 몇번의 수정작업을 거쳐 현재에 이르렀다. 이 X.500은 최초의 일반적인 목적의 디렉토리 시스템이었고 다양한 쿼리를 사용하는 강력한 검색기능을 제공하였을 뿐만 아니라 서버와 데이터의 분산이 용이했고 그리고 무엇보다도 특정 운영체제나 특정 네트웍,특정 응용프로그램에 구애받지 않고 사용될 수 있는 표준이라는 점이 눈길을 끌 수 있었다.

Decoy documents used by the "e4004c1f" variant include a Korean text on the LDAP protocol.

Apart from the similarities with other malware established in the publications mentioned above, this variant has been distributed in a particular installer which includes the backdoor in an embedded password-protected zip archive. The password for this zip archive is *"!1234567890 dghtdhtrhgfjnui$%^^&fdt*" - identical to the password used by Destover "Volgmer" backdoors already detailed in this paper. There are also code similarities with Volgmer elsewhere – for example, the function to declare network API's from ws2_32.dll is identical, and the API names are encoded using the same API obfuscation scheme.

The C&C configuration can be hardcoded, or stored in a data file and subkeys under the registry key **HKLM\SYSTEM\CurrentControlSet\Control\WMI\Security.**

Some variant A samples uses subkey **a57890bc-ca23-3453-a23c-d385e9058fdf**
Some variant C samples uses subkey **821d1af-7a08-4b06-81cd-869365cdf713**



The network API declaration function of a Destover "Volgmer" and a Destover "e4004c1f" backdoor.

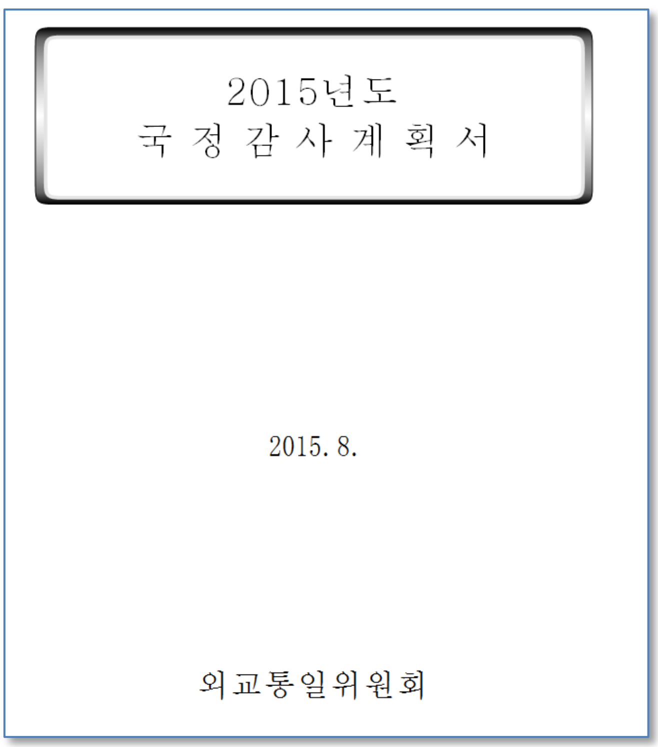
DESTOVER "BASICHWP" BACKDOOR, SEP 2015

This generation of backdoors is similar to the previous ones in that they use a custom SSL-like protocol for C&C communication. They have been further simplified, but use more C++ classes, and the 256-bit stream cipher *Caracachs* (hardcoded password "abcdefghijklmnopqrstuvwxyz012345") is used for both network traffic and API obfuscation. The same password is used in the example code for Caracachs found online (8), so no great effort has been taken to protect the encryption.

This variety of Destover is the third we have seen installed by documents exploiting the CVE-2015-6585 HWP vulnerability.

Command word set for this generation of backdoors is 0x8378-0x8390.

Decoy document content include a CV from an apparently South Korean individual, and a document apparently from the South Korean ***Foreign Affairs and Unification Committee**,* as seen below**.**


2015년도
국 정 감 사 계 획 서

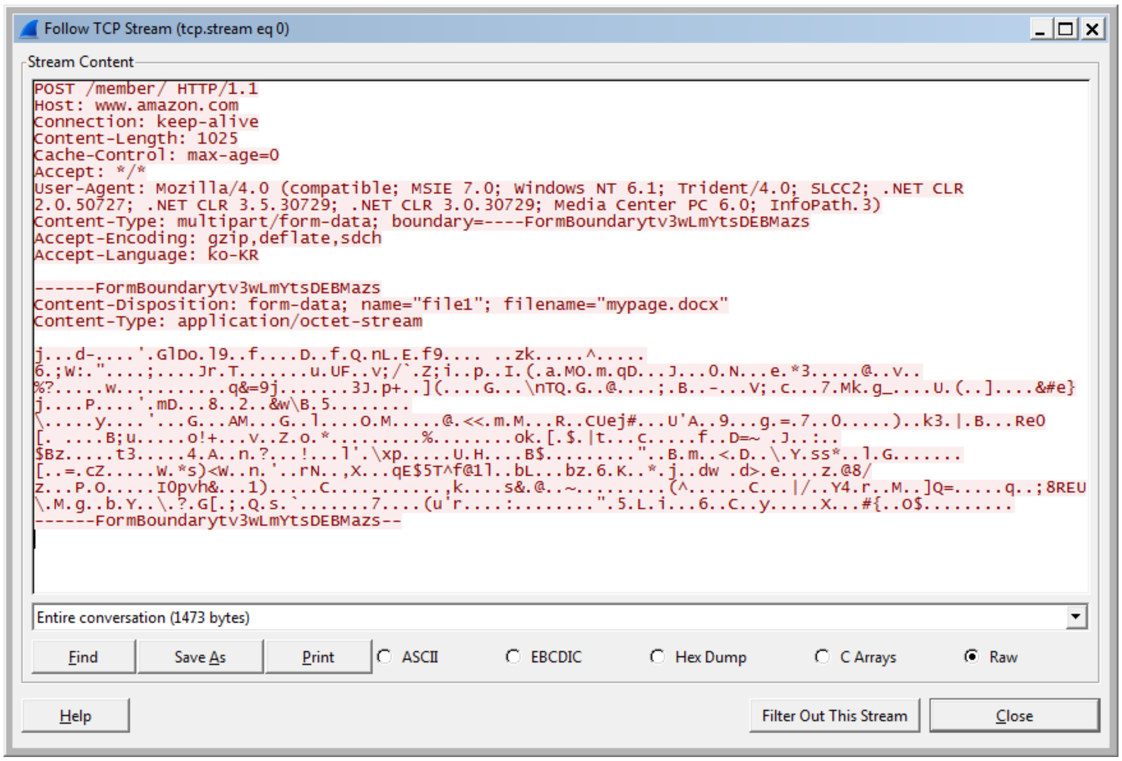
2015. 8.

외교통일위원회

Decoy: State information systems audit planning document, Aug 2015

DESTOVER "FORMBOUNDARY" BACKDOOR, NOV 2015

This backdoor has many code overlaps with *RandomDomain.B* – for example, it uses *CharSwap* API obfuscation, and uses the same set of integer commands. It has evolved away from the use of faked SSL, which means whole segments of code have been removed, including most of the domain names used for the SSL handshake. Instead, it connects to the C&C server via regular HTTP on port 80 and initially posts a blob of random data disguised as a legitimate file. Any real content is sent encrypted afterwards, using one of the bytewise XOR encodings known from *RandomDomain.*

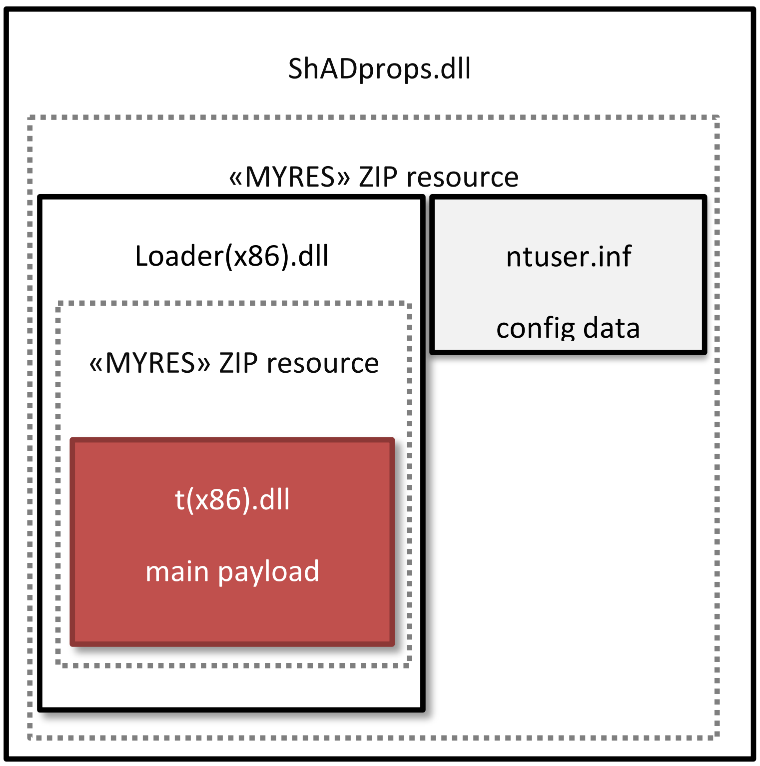


Sending initial POST statement to C&C server

The HTTP header fields can vary – many are selected from hardcoded lists, including the "Host" field. The FormBoundary string is terminated by a randomly generated character sequence, and the malware queries the system via the API call ***ObtainUserAgentString*** to get the current default User Agent. If this call fails, the hardcoded User Agent "AgentString" is used instead.

DESTOVER "VOLGMER2" BACKDOOR, JAN 2016

This was found as a DLL backdoor sample "t(x86).dll" which contained several traits in common with the Volgmer series. Further data mining revealed that identically to Volgmer, the sample is installed by a dropper which contains the DLL in an embedded zip file resource named "MYRES" in its body. This dropper is *again* extracted by another outer dropper with a similar embedded zip inside, which also in addition contains a configuration file *ntuser.inf*.



This config file contains - among other things - C&C IP and port information, which is read and written to a registry key before being used by the main payload component.

HKLM\SYSTEM\CurrentControlSet\Control\WMI\Security subkey = "**72ca1d1af-7afc-4c06-cc1d-8feaac5cdf764**".

Volgmer2 shares API declaration functions and string decode algorithms with the original Volgmer. However, there are also clear differences. Its network behavior has moved away from HTTP post with the recognizable "Mozillar" UserAgent. Instead, C&C traffic is performed via faked SSL with another encryption twist – RC4 with a layer of XOR on top. They RC4 key is binary, and hardcoded in the executable: 0x0d, 0x06 ,0x09, 0x2a, 0x86, 0x48, 0x86, 0xf7, 0x0d, 0x01, 0x01, 0x01, 0x05, 0x00, 0x03, 0x82. Similarly to the RandomDomain series, Volgmer2 uses domain names chosen randomly from a list in its SSL handshake.

The dropper executables in the “Volgmer 1” series contained some checks for VM environments. Volgmer2 has taken this further, and included a number of anti-debugging tricks and of checks for what appears to be known sandbox environments.

```
Volgmer__IsInsideVM proc near           ; CODE XREF: sub_10003810+45↓p

arg_0           = dword ptr  4

                push    esi
                mov     esi, [esp+4+arg_0]
                mov     dword ptr [esi+18h], 0
                call    Volgmer__IsVMWare
                test    eax, eax
                jz      short loc_100031DD
                mov     eax, [esi+18h]
                or      al, 1
                mov     [esi+18h], eax

loc_100031DD:                           ; CODE XREF: Volgmer__IsInsideVM+13↑j
                call    Volgmer__IsVBOX
                test    eax, eax
                jz      short loc_100031EE
                mov     eax, [esi+18h]
                or      al, 2
                mov     [esi+18h], eax

loc_100031EE:                           ; CODE XREF: Volgmer__IsInsideVM+24↑j
                call    Volgmer__IsVirtualPC
                test    al, al
                jz      short loc_100031FF
                mov     eax, [esi+18h]
                or      al, 4
                mov     [esi+18h], eax

loc_100031FF:                           ; CODE XREF: Volgmer__IsInsideVM+35↑j
                pop     esi
                retn
Volgmer__IsInsideVM endp
```

```
Volgmer2__VMCheck proc near             ; CODE XREF: wWinMain(x,x,x,x)+23↓p
                call    Volgmer2__IsVMWare
                test    eax, eax
                jnz     short loc_401D47
                call    Volgmer2__IsVBOX
                test    eax, eax
                jnz     short loc_401D47
                call    Volgmer2__IsVirtualPC
                test    al, al
                jz      short loc_401D50

loc_401D47:                             ; CODE XREF: Volgmer2__VMCheck+7↑j
                                        ; Volgmer2__VMCheck+10↑j
                call    Volgmer2__IsComputerName_KnownSB
                test    eax, eax
                jnz     short bailout

loc_401D50:                             ; CODE XREF: Volgmer2__VMCheck+19↑j
                call    Volgmer2__IsDebuggerPresent
                test    eax, eax
                jnz     short bailout
                call    Volgmer2__IsRemoteDebuggerPresent
                test    eax, eax
                jnz     short bailout
                call    Volgmer2__CheckNtGlobalFlags
                test    eax, eax
                jnz     short bailout
                call    Volgmer2__CheckProcessDebugPort
                test    eax, eax
                jnz     short bailout
                call    Volgmer2__CheckProcessDebugHandle
                test    eax, eax
                jnz     short bailout
                call    Volgmer2__CheckProcessDebugFlags
                test    eax, eax
                jnz     short bailout
                retn
; ---------------------------------------------------------------------------

bailout:                                ; CODE XREF: Volgmer2__VMCheck+22↑j
                                        ; Volgmer2__VMCheck+2B↑j ...
                push    1
                pop     eax
                retn
Volgmer2__VMCheck endp
```

Volgmer1 vs Volgmer2 dropper evasions.

The change also means that the malware *continues to work* if under a virtualized environment, if there are no other indicators that there is monitoring or debugging activity going on. The check for known sandbox environments is done by comparing the computer name with the names in the following list:

MARS53
35347
JOHN-PC
TVMCOM
PLACEHOL-6F699A
WIN7PRO-MALTEST
WINDOWS-F99AACA
XELRCUZ-AZ
RATS-PC
PXE472179

The command integers used by Volgmer2 are in the range 0x09-0x27 with the exception of 0x17, 0x1b and 0x1c.

APPENDIX: ALGORITHMS AND OTHER INDICATORS

Chopstring obfuscation

```
Destover__SkipPeriods proc near          ; CODE XREF: sub_404180+24↓p
                                         ; sub_404180+3E↓p ...

arg_0           = dword ptr  4

                mov     edx, [esp+arg_0]
                push    esi
                mov     esi, offset unk_413E80
                push    edi
                mov     ecx, 14h
                xor     eax, eax
                mov     edi, esi
                rep stosd
                cmp     byte ptr [edx], 0
                jz      short loc_404170

loc_40415B:                              ; CODE XREF: Destover__SkipPeriods+2E↓j
                mov     al, [edx]
                cmp     al, 2Eh
                jz      short loc_404168
                cmp     al, 20h
                jz      short loc_404168
                mov     [esi], al
                inc     esi

loc_404168:                              ; CODE XREF: Destover__SkipPeriods+1F↑j
                                         ; Destover__SkipPeriods+23↑j
                mov     al, [edx+1]
                inc     edx
                test    al, al
                jnz     short loc_40415B

loc_404170:                              ; CODE XREF: Destover__SkipPeriods+19↑j
                pop     edi
                mov     eax, offset unk_413E80
                pop     esi
                retn
Destover__SkipPeriods endp
```

Chopstring deobfuscator

```
call    ds:LoadLibraryA
mov     esi, eax
test    esi, esi
jz      loc_100031E5
push    offset aGet_Pr_oc_ad_d ; " Get. Pr.oc  .Ad.dr ess"
call    Destover__SkipPeriods
add     esp, 4
push    eax             ; lpProcName
push    esi             ; hModule
call    ds:GetProcAddress
```

Deobfuscation of the API name before it is sent to GetProcAddress. Yes, they look up GetProcAddress using GetProcAddress. Go figure.

XOR-A7 obfuscation

This is a forward bytewise XOR encoding using 0xA7 as key.

```
; =============== S U B R O U T I N E =======================================

; [IDASUB] MD5 present in Idasub database, hits :  2
; [IDASUB] Last synced with Idasub Sat Jan 24 23:19:21 2015
; [IDASUB] Match: Fuzzy
;

Destover__DecodeString proc near        ; CODE XREF: sub_401730+62↑p
                                        ; sub_401730+7B↑p ...

arg_0           = dword ptr  4

                push    ebx
                mov     ebx, [esp+4+arg_0]
                push    ebp
                push    esi
                push    edi
                mov     edi, ebx
                or      ecx, 0FFFFFFFFh
                xor     eax, eax
                repne scasb
                not     ecx
                push    ecx             ; size_t
                call    _malloc
                mov     ebp, eax
                mov     edi, ebx
                or      ecx, 0FFFFFFFFh
                xor     eax, eax
                add     esp, 4
                xor     esi, esi
                repne scasb
                not     ecx
                dec     ecx
                jz      short loc_404127
                mov     edi, ebx
                mov     edx, ebp
                sub     edi, ebp
                mov     [esp+10h+arg_0], edi
                jmp     short loc_40410E
; ---------------------------------------------------------------------------

loc_40410A:                             ; CODE XREF: Destover__DecodeString+55
                mov     edi, [esp+10h+arg_0]

loc_40410E:                             ; CODE XREF: Destover__DecodeString+38
                mov     al, [edi+edx]
                mov     edi, ebx
                xor     al, 0A7h
                inc     esi
```

```
; =============== S U B R O U T I N E =======================================

; [IDASUB] Name imported from Idasub database. Hits in the DB :  2
; [IDASUB] Last synced with Idasub Sun Jan 25 00:06:44 2015
; [IDASUB] Match: Fuzzy
;

Destover__DecodeString proc near        ; CODE XREF: sub_10001760+46↑p
                                        ; sub_10001760+60↑p ...

arg_0           = dword ptr  4

                push    ebx
                mov     ebx, [esp+4+arg_0]
                push    ebp
                push    esi
                push    edi
                mov     edi, ebx
                or      ecx, 0FFFFFFFFh
                xor     eax, eax
                repne scasb
                not     ecx
                push    ecx             ; size_t
                call    _malloc
                mov     ebp, eax
                mov     edi, ebx
                or      ecx, 0FFFFFFFFh
                xor     eax, eax
                add     esp, 4
                xor     esi, esi
                repne scasb
                not     ecx
                dec     ecx
                jz      short loc_100047B7
                mov     edi, ebx
                mov     edx, ebp
                sub     edi, ebp
                mov     [esp+10h+arg_0], edi
                jmp     short loc_1000479E
; ---------------------------------------------------------------------------

loc_1000479A:                           ; CODE XREF: Destover__DecodeString+
                mov     edi, [esp+10h+arg_0]

loc_1000479E:                           ; CODE XREF: Destover__DecodeString+
                mov     al, [edi+edx]
                mov     edi, ebx
                xor     al, 0A7h
                inc     esi
```

String deobfuscation functions in the Sony Destover (left) malware and Destover “b076e058” (right). They are identical, even down to using 0xa7 as xor key.

XOR-XX-SUB-XX obfuscation

This is a forward bytewise XOR, SUB encoding, usually used in communication encryption/decryption. The inverse is usually also present in the form of ADD, XOR. Many different byte combinations are used in the various variants.

```
Destover__XOR73SUB3A proc near          ; CODE XREF: sub_4042C0+D0↑p

arg_0           = dword ptr  4
arg_4           = dword ptr  8

                cmp     ecx, [esp+arg_4]
                jz      short loc_404670
                cmp     ecx, [esp+arg_4]
                jz      short loc_404670
                mov     ecx, [esp+arg_4]

loc_404670:                             ; CODE XREF: Destover__XOR73SUB3A+4↑j
                                        ; Destover__XOR73SUB3A+A↑j
                cmp     eax, [esp+arg_0]
                jz      short loc_404682
                cmp     eax, [esp+arg_0]
                jz      short loc_404682
                xor     eax, eax
                add     eax, [esp+arg_0]

loc_404682:                             ; CODE XREF: Destover__XOR73SUB3A+14↑j
                                        ; Destover__XOR73SUB3A+1A↑j
                test    ecx, ecx
                jle     short locret_404694

loc_404686:                             ; CODE XREF: Destover__XOR73SUB3A+32↓j
                mov     dl, [eax]
                xor     dl, 73h
                sub     dl, 3Ah
                mov     [eax], dl
                inc     eax
                dec     ecx
                jnz     short loc_404686

locret_404694:                          ; CODE XREF: Destover__XOR73SUB3A+24↑j
                retn
Destover__XOR73SUB3A endp
```

BC-SUB API Obfuscation

This is a forward bytewise decoding where the each character value is subtracted from 0xBC to arrive at a cleartext character.

This decoding is used instead of ChopString in some KorDllbot variants.

```
                push    ebp
                mov     ebp, esp
                sub     esp, 100h
                push    [ebp+lpString2] ; lpString2
                lea     eax, [ebp+String1]
                push    eax             ; lpString1
                call    ds:lstrcpyA
                cmp     [ebp+String1], 0
                lea     eax, [ebp+String1]
                jz      short loc_10001259

loc_10001246:                           ; CODE XREF: sub_1000121E+39↓j
                mov     cl, [eax]
                cmp     cl, 'A'
                jl      short loc_10001253
                mov     dl, 0BCh
                sub     dl, cl
                mov     [eax], dl

loc_10001253:                           ; CODE XREF: sub_1000121E+2D↑j
                inc     eax
                cmp     byte ptr [eax], 0
                jnz     short loc_10001246

loc_10001259:                           ; CODE XREF: sub_1000121E+26↑j
                lea     eax, [ebp+String1]
                push    eax             ; _DWORD
                push    [ebp+arg_0]     ; _DWORD
                call    GetProcAddress
                leave
                retn
```

DB-SUB API Obfuscation

This is a forward bytewise decoding where the each character value above ‘a’ and below ‘z’ is subtracted from 0xDB to arrive at a cleartext character.

```
; int __cdecl Destover__DB_SubDecode(int, LPCSTR lpString2)
Destover__DB_SubDecode proc near        ; CODE XREF: sub_40127D+2C↑p
                                        ; sub_40127D+3C↑p ...

String1         = byte ptr -100h
arg_0           = dword ptr  8
lpString2       = dword ptr  0Ch

                push    ebp
                mov     ebp, esp
                sub     esp, 100h
                push    [ebp+lpString2] ; lpString2
                lea     eax, [ebp+String1]
                push    eax             ; lpString1
                call    ds:lstrcpyA
                cmp     [ebp+String1], 0
                lea     eax, [ebp+String1]
                jz      short loc_4013FE

loc_4013E6:                             ; CODE XREF: Destover__DB_SubDecode+3E↓j
                mov     cl, [eax]
                cmp     cl, 'b'
                jl      short loc_4013F8
                cmp     cl, 'y'
                jg      short loc_4013F8
                mov     dl, 0DBh
                sub     dl, cl
                mov     [eax], dl

loc_4013F8:                             ; CODE XREF: Destover__DB_SubDecode+2D↑j
                                        ; Destover__DB_SubDecode+32↑j
                inc     eax
                cmp     byte ptr [eax], 0
                jnz     short loc_4013E6

loc_4013FE:                             ; CODE XREF: Destover__DB_SubDecode+26↑j
                lea     eax, [ebp+String1]
                push    eax
                push    [ebp+arg_0]
                call    GetProcAddress_0
                leave
                retn
Destover__DB_SubDecode endp
```

CharSwap API Obfuscation

This is an encoding where some character ASCII values are increased or decreased by nine.

```
.text:004017D0 ; int __cdecl Destover__CharSwap(int, LPCSTR lpString2)
.text:004017D0 Destover__CharSwap proc near             ; CODE XREF: DestoverRandom__DeclareNetworkApis+261↑p
.text:004017D0                                          ; DestoverRandom__DeclareNetworkApis+271↑p ...
.text:004017D0
.text:004017D0 String1         = byte ptr -100h
.text:004017D0 arg_0           = dword ptr  4
.text:004017D0 lpString2       = dword ptr  8
.text:004017D0
.text:004017D0                 mov     eax, [esp+lpString2]
.text:004017D4                 sub     esp, 100h
.text:004017DA                 lea     ecx, [esp+100h+String1]
.text:004017DE                 push    eax             ; lpString2
.text:004017DF                 push    ecx             ; lpString1
.text:004017E0                 call    ds:lstrcpyA
.text:004017E6                 mov     al, [esp+100h+String1]
.text:004017EA                 lea     ecx, [esp+100h+String1]
.text:004017EE                 test    al, al
.text:004017F0                 jz      short loc_401818
.text:004017F2
.text:004017F2 loc_4017F2:                              ; CODE XREF: Destover__CharSwap+46↓j
.text:004017F2                 mov     al, [ecx]
.text:004017F4                 cmp     al, 'b'
.text:004017F6                 jl      short loc_401810
.text:004017F8                 cmp     al, 'y'
.text:004017FA                 jg      short loc_401810
.text:004017FC                 cmp     al, 'i'
.text:004017FE                 jl      short loc_401808
.text:00401800                 cmp     al, 'p'
.text:00401802                 jg      short loc_401808
.text:00401804                 add     al, 9
.text:00401806                 jmp     short loc_40180E
.text:00401808 ; ---------------------------------------------------------------------------
.text:00401808
.text:00401808 loc_401808:                              ; CODE XREF: Destover__CharSwap+2E↑j
.text:00401808                                          ; Destover__CharSwap+32↑j
.text:00401808                 cmp     al, 'r'
.text:0040180A                 jl      short loc_401810
.text:0040180C                 sub     al, 9
.text:0040180E
.text:0040180E loc_40180E:                              ; CODE XREF: Destover__CharSwap+36↑j
.text:0040180E                 mov     [ecx], al
.text:00401810
.text:00401810 loc_401810:                              ; CODE XREF: Destover__CharSwap+26↑j
.text:00401810                                          ; Destover__CharSwap+2A↑j ...
.text:00401810                 mov     al, [ecx+1]
.text:00401813                 inc     ecx
.text:00401814                 test    al, al
.text:00401816                 jnz     short loc_4017F2
.text:00401818
.text:00401818 loc_401818:                              ; CODE XREF: Destover__CharSwap+20↑j
.text:00401818                 mov     eax, [esp+100h+arg_0]
.text:0040181F                 lea     edx, [esp+100h+String1]
.text:00401823                 push    edx             ; _DWORD
.text:00401824                 push    eax             ; _DWORD
.text:00401825                 call    GetProcAddress_0
.text:0040182B                 add     esp, 100h
.text:00401831                 retn
```

CharSwap is used for obfuscation of both APIs and regular strings. Above figure shows API de-obfuscation.

The CharSwapped API names GetDriveTypeA, SetFileTime and Process32Next.

Intbox encoding

This encoding is used instead of ChopString in some Destover variants.

```
__int16 __cdecl Destover__IntboxDecode(int buffer, int bufferlength, int key_int)
{
  int bufferlength_; // edx@1
  int i2; // ecx@1
  int v5; // edi@1
  __int16 result; // ax@2
  signed int i1; // ebx@2
  char sbox[256]; // [sp+Ch] [bp-104h]@2
  int v9; // [sp+10Ch] [bp-4h]@1
  signed int i; // [sp+11Ch] [bp+Ch]@1

  bufferlength_ = bufferlength;
  i2 = -47 * (bufferlength + key_int) & 0xFF;
  v5 = 0;
  v9 = -33 * (bufferlength + key_int) & 0xFF;
  i = 0;
  do
  {
    result = 0x1D * (char)(key_int * (i + 1));
    i1 = i++;
    sbox[i1] = result;
  }
  while ( i < 256 );
  if ( bufferlength_ > 0 )
  {
    do
    {
      *(_BYTE *)(v5 + buffer) ^= sbox[i2];
      result = v9;
      i2 = (v9 + i2) & 0xFF;
      ++v5;
    }
    while ( v5 < bufferlength_ );
  }
  return result;
}
```

RC4+XOR encryption

This encryption is used by Volgmer2 on network traffic data.

```
// [IDASUB] MD5 present in Idasub database, hits :  0
// [IDASUB] Last synced with Idasub Mon Feb 08 21:06:52 2016
// [IDASUB] Match: Fuzzy
unsigned int __cdecl Volgmer2__EncryptData(int key, int keylength, _BYTE *output, unsigned int msglength)
{
  unsigned int result; // eax@1

  Volgmer2__RC4Crypt(key, keylength, (int)output, msglength);
  *output ^= output[msglength - 1];
  for ( result = 1; result < msglength; ++result )
    output[result] ^= output[result - 1];
  return result;
}
```

KorDllbot / Joanap AES keys

"Bb102@jH4$t3hg%6&G1s*2J3gCNwVr*UeI!Dr3hytg^CHGf%ion"
"b n4rbhriq890v9=023=01*&(T-0Q325J1N;LK'"

Koredos RC4 key

"A39405WKELsdfirpsdLDPskDORkbLRTP12330@3$223%!"

Joanap PLAIN_CRYPT keys

"9025jhdho39ehe2"
"hybrid!@hybrid!@#"
"iamsorry!@1234567"

Destover "b076e058" RSA authentication key string

"b076e0580463a202bad74cb9c1b85af3fb4d1be513ccca3ae8b57d193be77b4ab63802b3216d3a80b00827b693593
a76be884f41b491ee1f6136b3755add91e2de9b0f5b3849d463fcd7b9a3b6cd0744caf809f510ee04ab3c714f53422d2
4f33361f75145b08286d2d7d99704684ed1d25fd5a9dc7b993f8e4d074234fd82d3"

Destover "Volgmer.A" RSA authentication key string

"bc9b75a31177587245305cd418b8df78652d1c03e9da0cfc910d6d38ee4191d40bd51483321ebe44595f799da8421
5ebd7137c9e267f54a342048e510fddfdec2404764fdf128c330862e747d7a98cd557a15500051a5b6651572a398bbe
5a51d52dc7af3b34b06b68c7974b9f8e45fd3636fd628c1dbcf65bbb68b2dd058017"

Destover "Volgmer.B/C" RSA authentication key string

"b50a338264226b6d57c1936d9db140ba74a28930270a083353645a9b518661f4fcea160d73469b8beabc14b90e907
88c28f2d7c660e43db2e6f81aa05a08cae4517845ba4b9fc614e77e39d502003fcc6712d45428f339bcc06787745f734
1e9884fae803ad2fbb9670acb15b2da62735081fb2bc2a9b8b434dbe211a4b59b03"

Destover "b59d1659" RSA authentication key string

"b59d165982e3d5721c4d40195f85aedf2a12d6616be11a2c19fa11821604edc4675bdca4f9b9cbfb27244203ca8e21
500ae592d7bb2776e8ed9179dc1fb47819f140d0052f28865c201a036f3f698d0c256c3446e09c83eda056c91ee9e25
927148a3521439d57b0682a4c2723bd18dcd37c0f9b08ff8c7c3bc37684d2b4d241"

Destover "b8ac0905" RSA authentication key string

"b8ac0905cda0360fc115f614119da76d84e2277762bd7558b2650a79013fb50138f732d5a03730d7d5b173a12d9a8
42353ca433758d417fa8b452ec075f87bf76a7056ecdd2b063432f414e4ad52fdb078b8a9d84635774e5234ce28a762
d91af1cb9c026ffd68b88f1032c9c2c8fa1d187a054f906781c56fb07b0f6bb908cb"

Destover "e4004c1f" RSA authentication key string

"e4004c1f94182000103d883a448b3f802ce4b44a83301270002c20d0321cfd0011ccef784c26a400f43dfb901bca753
8f2c6b176001cf5a0fd16d2c48b1d0c1cf6ac8e1da6bcc3b4e1f96b0564965300ffa1d0b601eb2800f489aa512c4b248c
01f76949a60bb7f00a40b1eab64bdd48e8a700d60b7f1200fa8e77b0a979dabf"

Destover "Randomdomain.A/B" SSL remote server names contained in Client Hello

wwwimages2.adobe.com
www.paypalobjects.com
www.paypal.com
www.linkedin.com
www.apple.com
www.amazon.com
www.adobetag.com
windowslive.tt.omtrdc.net
verify.adobe.com
us.bc.yahoo.com
urs.microsoft.com
supportprofile.apple.com
support.oracle.com
support.msn.com
startpage.com
sstats.adobe.com
ssl.gstatic.com
ssl.google-analytic.com
srv.main.ebayrtm.com
skydrive.live.com
signin.ebay.com
securemetrics.apple.com
secureir.ebaystatic.com
secure.skypeassets.com
secure.skype.com
secure.shared.live.com
secure.logmein.com
sc.imp.live.com
sb.scorecardresearc.com
s1-s.licdn.com
s.imp.microsoft.com
pixel.quantserve.com
p.sfx.ms
mpsnare.iesnare.com
login.yahoo.com
login.skype.com
login.postini.com
login.live.com
l.betrad.com
images-na.ssl-images-amazon.com
fls-na.amazon.com
extended-validation-ssl.verisign.com
daw.apple.com
csc.beap.bc.yahoo.com
by.essl.optimost.com
b.stats.ebay.com
apps.skypeassets.com
api.demandbase.com
ad.naver.com
accounts.google.com

Destover "Randomdomain.C" SSL remote server names contained in Client Hello

myservice.xbox.com
uk.yahoo.com
web.whatsapp.com
www.apple.com
www.baidu.com
www.bing.com
www.bitcoin.org
www.comodo.com
www.debian.org
www.dropbox.com
www.facebook.com
www.github.com
www.google.com
www.lenovo.com
www.microsoft.com
www.paypal.com
www.tumblr.com
www.twitter.com
www.wetransfer.com
www.wikipedia.org

Destover "Volgmer2" SSL remote server names contained in Client Hello

ad.naver.com
all.baidu.com
www.amazon.com
www.apple.com
www.bing.com
www.dell.com
www.hp.com
www.microsoft.com
www.oracle.com
www.paypal.com
www.uc.com
www.yahoo.com

(Note that domain names included in Destover SSL handshakes are legitimate and used only as disguise.)

APPENDIX: THE MICROSOFTCODESIGNINGPCA SELF-SIGNED SAMPLE CLUSTER

Group: 03c64293830f4c8f43666b3901d02332	
87bae4517ff40d9a8800ba4d2fa8d2f9df3c2e224e97c4b3c162688f2b0d832e	KorDllbot v1.1 backdoor service, listening on port 179
Group: 3d348a74aab5359d422da7fad24b8c2c	
a7d088bf3ae2a82f711f816922779ac7b720170298ac43c76cf8c6e1aa8dfadd	Proxymini 0.2.1, Luigi Auriemma
fd95e095658314c9815df6a97558897cb344255bd54d03c965fa4cbd16d7bafd	NoiseSin data stealer
82169a2d8f15680c93e1436687538afa01d6a2ecfe7a7cb613817c64a1a82342	NoiseSin data stealer
792b484ac94f0baefc7e016895373ba92c2927e3463f62adb701ddbe4c90604c	KorDllbot backdoor (Unobfuscated API loading)
162d6223c1c1219ca81a77e60e6b776058517272fe7cac828a3f64dcacd87811	KorDllbot backdoor (XOR-obfuscated API loading)
56e0b1794a588e330e32a10813cdc9904e472c55f17dd6c8de341aeaf837d077	Keylogger
c16a66c1d8e681e962f03728411230fe7c618b7294c143422005785d3a724ec4	Dropper for **162d6223c1c1219ca81a77e60e6b776058517272fe7cac828a3f64dcacd87811**
57b4c2e71f46fe3e7811a80d19200700c15dd358bdf9d9fdf61f1c9a669f7b4b	NoiseSin data stealer
Group: 09b075a5393e93a3479a00051714de52	
2d9edf45988614f002b71899740d724008e9a808efad00fa79760b31e0a08073	Joanap backdoor and SMB worm
006e0cc29697db70b2d4319f320aa0e52f78bf876646f687aa313e8ba04e6992	Joanap backdoor and SMB worm
dda136bc51670e57a4b2f091f83ab7b44291a9323d5483abd9e91b78221e027f	Data harvester
Group: 17522941a80c25ab4c9cfe5f28d9361f	
163571bd56001963c4dcb0650bb17fa23ba23a5237c21f2401f4e894dfe4f50d	SMB worm and backdoor dropper for **f901083da11222e3221f5d3e5d5f79d7ea3864282ea565e47c475ad23ef96ff4**
Group: 9d0550e00b6d5da9407e28bca4336cc9	

3d2a7ea04d2247b49e2dcad63a179ae6a47237eddbfd354082f1417a63e9696e	Joanap backdoor and SMB worm
ea46ed5aed900cd9f01156a1cd446cbb3e10191f9f980e9f710ea1c20440c781	Joanap backdoor and SMB worm
Group: e7d382fb2e1ea4a44a8d193f4014e514	
6e8a2329567cdbbba68460ccb97209867d7508983cb638662b33bfe90d0134d4	KorDllbot backdoor dropper, disguised as a Korean Windows hotpatch
af7b53ce584b83085488e1190e1458948eaf767631f766e446354d0d5523e9d0	Dropped KorDllbot component
69300a42e055f68a8057192077fbbef3be5b66514ea9ca258b077c5c7e9417a9	KorDllbot backdoor dropper
Group: 14ccfa0756059e93469bfef60935d999	
e0cd4eb8108dab716f3c2e94e6c0079051bfe9c7c2ed4fcbfdd16b4dd1c18d4d	SMB worm and backdoor dropper for **a795964bc2be442f142f5aea9886ddfd297ec898815541be37f18ffeae02d32f**
96c35225dc4cac65cc43a6cc6cdcce3d13b3bda286c8c65cad5f2879f696ad2a	Backdoor dropper for **0075d16d8c86f132618c6365369ff1755525180f919eb5c103e7578be30391d6**
Group: c23d8473c335159a435b5c920b961971	
29355f6d4341089b36834b4a941ef96b3bf758a4fe35fbb401cc4e74b9b1c90f	Yahoo IM backdoor service
9e226a5eb4de19fcb3f7ecc3abcf52ea22a1f1a42a08dd104f5f7a00164e074e	Yahoo IM backdoor exe
041605e498bb41b07d2d43003152cc2a992e7e2ade7a47ee9aef2570bdb16d94	Yahoo IM backdoor exe
82fe3a8f2248643505e8de1977b734f97eb38225e6d3df6ea8f906430514b4f5	Yahoo IM backdoor exe
Group: a02925c39912b68a4a0555246a031abb	
08203b4ddc9571418b2631ebbc50bea57a00eadf4d4c28bd882ee8e831577a19	Joanap dropper, backdoor and SMB worm
Group: f487c2cfd330cf8e4f9171672d99cecd	
8e3c3398353931c513c32330c07f65b6ee6f62fc7a56edac7cbe4edb1bf4c74e	KorDllbot backdoor dropper
bb4204dd059849848e9492523ce32520bf37cb80974320c0ca71f3b79e83f462	Downloader and backdoor

2f8c448bb05ed1218e638c61bb56ebb953b962ed5e065b08fa03cfcf6f6a1c68	Downloader and backdoor
Group: e4046a19ef86378a43907279d072e5fb	
f98c67c4cf9b02acaabb555664a0d9d648a1e43f681f9bf234af066d5451be8d	KorDllbot 1.05.2 downloader and backdoor
Group: 33f8c3f1b7df61b949ed876422818bb1	
1226d3635c1a216be9316c9dfa97f103c79ed4c44397e5e675d3b1e37786bf31	KorDllbot backdoor
Group: de85322cb067a1aa41af54c2de87fb03	
c5baece9978649659220af2681a3a43b83f8ae47afdd3862185d1fec7735a7d2	Dropped KorDllbot component
a4b982d4e7137d7d3687f3127e6d5c2a8b2be1f53daeebce9175461c7e6a53cd	KorDllbot backdoor dropper
9bcecd6afa54eb4f343b7eb82a86ceee189cc10bc91fa83f8cdc98cc5aaef117	KorDllbot backdoor dropper, disguised as a Korean Windows hotpatch
Group: dde039353663cdb14337e6793ca2a8cf	
b7f2595dd62d1174ce6e5ddf43bf2b42f7001c7a4ec3c4cbe3359e30c674ed83	KorDllbot backdoor
Group: 940888706c199a8342ef85eb60fecbb6	
b039383a19e3da74a5a631dfe4e505020a5c5799578187e4ccc016c22872b246	KorDllbot backdoor service installer
f4a06dd6ebfd0805d445f45ce33d7bba4a33c561111c39a347024069a78169e9	KorDllbot backdoor service
3acaea01fd79484d5a72c72e1b9c2fbf391145fb1533c17a8a83e897d8777f82	Removes backdoor service
81067f057d523fdcddf7df1da39a7c3614c45f6bff6bd387274c049244efda3b	Removes backdoor service
Group: 7940994b304aa1ac4d2d64e6b7b8890d	
218ee208323dc38ebc7f63dba73fac5541b53d7ce1858131fa3bfd434003091d	KorDllbot backdoor service installer
73edc54abb3d6b8df6bd1e4a77c373314cbe99a660c8c6eea770673063f55503	KorDllbot backdoor service
Group: 328e8fb5f3ec48894f6af0eb0a821d01	

6d5d706f5356e087f5961ba2ed808c51876d15c2e09eb081618767b36b1d012f	KorDllbot backdoor service
Group: 7301505ed41ad49a4b379588d64be787	
7a538c3eed1f01b62a19226750c1369e4e9210b1331d5829ca91fe2b69087f06	Downloader
6059cb08489170aea77caf0940131e5765b153a593e76d93a0f244e89ddb9e90	Uploader
e97a8909349a072ed945899fbe276fc27e9c5847bc578b0abccf017da3fd680c	Dropper for **7a538c3eed1f01b62a19226750c1369e4e9210b1331d5829ca91fe2b69087f06**
Group: f0eeae68ca747c804b6a1d078525ebd1	
c4852ddba88e5c53a8711c4c7540b7ac98dac6b9e31d10dd999a81a4f0e117c3	KorDllbot backdoor service
3ebb3d8292a1aa5dc81b028beeefdec0f0448516d6225b336ee37d550ab8c3ab	KorDllbot backdoor service
Group: 61fd3dc8a14f3a9f4ffbb82b6b9165c2	
87e68055959328d857b287e797896d9a96695b69ed300a843eee73319427b3b3	KorDllbot 1.03 backdoor service
94e14a85a2046b40842f6c898c5f6c3200de3d89c178a9a9f9a639c1d3de9ee9	KorDllbot 1.04.4 backdoor
Group: 00f70a83e7c9fbb54ea74e8bbc14c609	
cd8c729da299b29618819afeef8b2a79451e6c3d35dea3769ef638c649c69001	KorDllbot 1.04.4 backdoor service
Group: b46daf51cd766faa487311beac043847	
9d9889585f1a4048a3955d3a9cead2f426a509afaeacad27540382cc3266f0fa	KorDllbot backdoor service
Group: 10cc28f0b769aba64fe81a0cd640122f	
888844c040be9d0fc3dab00dd004aa9e8619f939aff2eba21e4f48ca20e13784	KorDllbot 1.2 backdoor service
Group: db8c962c5c8366854f9b052dab52d54a	
d7044a35e76543a03cd343d71652c7bbd9a28e246d7f3a43f4a2e75cd0ef7366	KorDllbot 1.04.5 backdoor service
Group: 206f156f15bb3c814f24bebf69ec04c7	

50974c15a546e961fbee8653e5725960a77b79e0f7c8eadf3b6d35ba3a46dd57	KorDllbot backdoor service
Group: 7c4a1d98042a2d814c93e8d8f78ee6fe	
bfb5fa2a09ac60efcc0e9f05e781bd22cae0b8f6ba356d7819285f073845a0eb	KorDllbot 1.03 backdoor service
Group: 888ba4e41cd689a14ee48b2dbe87428e	
9bc8fe605a4ad852894801271efd771da688d707b9fbe208106917a0796bbfdc	KorDllbot service dropper. Drops **0a27acaaebc7db0878239b40ab9d2feff13888839c05a03348fc09b78de6ced5**
7b171a160cb2a17f87ca6a4a1c62b4cd9e718f987b7278d3effe0614b5b51be4	KorDllbot service dropper. Drops **0a27acaaebc7db0878239b40ab9d2feff13888839c05a03348fc09b78de6ced5**
0a27acaaebc7db0878239b40ab9d2feff13888839c05a03348fc09b78de6ced5	KorDllbot backdoor service

APPENDIX: MALWARE HASHES

KorDllbot-related samples

87bae4517ff40d9a8800ba4d2fa8d2f9df3c2e224e97c4b3c162688f2b0d832e
fd95e095658314c9815df6a97558897cb344255bd54d03c965fa4cbd16d7bafd
82169a2d8f15680c93e1436687538afa01d6a2ecfe7a7cb613817c64a1a82342
792b484ac94f0baefc7e016895373ba92c2927e3463f62adb701ddbe4c90604c
162d6223c1c1219ca81a77e60e6b776058517272fe7cac828a3f64dcacd87811
56e0b1794a588e330e32a10813cdc9904e472c55f17dd6c8de341aeaf837d077
c16a66c1d8e681e962f03728411230fe7c618b7294c143422005785d3a724ec4
57b4c2e71f46fe3e7811a80d19200700c15dd358bdf9d9fdf61f1c9a669f7b4b
2d9edf45988614f002b71899740d724008e9a808efad00fa79760b31e0a08073
006e0cc29697db70b2d4319f320aa0e52f78bf876646f687aa313e8ba04e6992
dda136bc51670e57a4b2f091f83ab7b44291a9323d5483abd9e91b78221e027f
163571bd56001963c4dcb0650bb17fa23ba23a5237c21f2401f4e894dfe4f50d
3d2a7ea04d2247b49e2dcad63a179ae6a47237eddbfd354082f1417a63e9696e
ea46ed5aed900cd9f01156a1cd446cbb3e10191f9f980e9f710ea1c20440c781
6e8a2329567cdbbba68460ccb97209867d7508983cb638662b33bfe90d0134d4
af7b53ce584b83085488e1190e1458948eaf767631f766e446354d0d5523e9d0
69300a42e055f68a8057192077fbbef3be5b66514ea9ca258b077c5c7e9417a9
e0cd4eb8108dab716f3c2e94e6c0079051bfe9c7c2ed4fcbfdd16b4dd1c18d4d
96c35225dc4cac65cc43a6cc6cdcce3d13b3bda286c8c65cad5f2879f696ad2a
29355f6d4341089b36834b4a941ef96b3bf758a4fe35fbb401cc4e74b9b1c90f
9e226a5eb4de19fcb3f7ecc3abcf52ea22a1f1a42a08dd104f5f7a00164e074e
041605e498bb41b07d2d43003152cc2a992e7e2ade7a47ee9aef2570bdb16d94
82fe3a8f2248643505e8de1977b734f97eb38225e6d3df6ea8f906430514b4f5
08203b4ddc9571418b2631ebbc50bea57a00eadf4d4c28bd882ee8e831577a19
8e3c3398353931c513c32330c07f65b6ee6f62fc7a56edac7cbe4edb1bf4c74e
bb4204dd059849848e9492523ce32520bf37cb80974320c0ca71f3b79e83f462
2f8c448bb05ed1218e638c61bb56ebb953b962ed5e065b08fa03cfcf6f6a1c68
f98c67c4cf9b02acaabb555664a0d9d648a1e43f681f9bf234af066d5451be8d
1226d3635c1a216be9316c9dfa97f103c79ed4c44397e5e675d3b1e37786bf31
c5baece9978649659220af2681a3a43b83f8ae47afdd3862185d1fec7735a7d2
a4b982d4e7137d7d3687f3127e6d5c2a8b2be1f53daeebce9175461c7e6a53cd
9bcecd6afa54eb4f343b7eb82a86ceee189cc10bc91fa83f8cdc98cc5aaef117
b7f2595dd62d1174ce6e5ddf43bf2b42f7001c7a4ec3c4cbe3359e30c674ed83
b039383a19e3da74a5a631dfe4e505020a5c5799578187e4ccc016c22872b246
f4a06dd6ebfd0805d445f45ce33d7bba4a33c561111c39a347024069a78169e9
3acaea01fd79484d5a72c72e1b9c2fbf391145fb1533c17a8a83e897d8777f82
81067f057d523fdcddf7df1da39a7c3614c45f6bff6bd387274c049244efda3b
218ee208323dc38ebc7f63dba73fac5541b53d7ce1858131fa3bfd434003091d
73edc54abb3d6b8df6bd1e4a77c373314cbe99a660c8c6eea770673063f55503
6d5d706f5356e087f5961ba2ed808c51876d15c2e09eb081618767b36b1d012f
7a538c3eed1f01b62a19226750c1369e4e9210b1331d5829ca91fe2b69087f06
6059cb08489170aea77caf0940131e5765b153a593e76d93a0f244e89ddb9e90
e97a8909349a072ed945899fbe276fc27e9c5847bc578b0abccf017da3fd680c
c4852ddba88e5c53a8711c4c7540b7ac98dac6b9e31d10dd999a81a4f0e117c3
3ebb3d8292a1aa5dc81b028beeefdec0f0448516d6225b336ee37d550ab8c3ab
87e68055959328d857b287e797896d9a96695b69ed300a843eee73319427b3b3
94e14a85a2046b40842f6c898c5f6c3200de3d89c178a9a9f9a639c1d3de9ee9
cd8c729da299b29618819afeef8b2a79451e6c3d35dea3769ef638c649c69001
9d9889585f1a4048a3955d3a9cead2f426a509afaeacad27540382cc3266f0fa
888844c040be9d0fc3dab00dd004aa9e8619f939aff2eba21e4f48ca20e13784

d7044a35e76543a03cd343d71652c7bbd9a28e246d7f3a43f4a2e75cd0ef7366
50974c15a546e961fbee8653e5725960a77b79e0f7c8eadf3b6d35ba3a46dd57
bfb5fa2a09ac60efcc0e9f05e781bd22cae0b8f6ba356d7819285f073845a0eb
9bc8fe605a4ad852894801271efd771da688d707b9fbe208106917a0796bbfdc
7b171a160cb2a17f87ca6a4a1c62b4cd9e718f987b7278d3effe0614b5b51be4
0a27acaaebc7db0878239b40ab9d2feff13888839c05a03348fc09b78de6ced5

Joanap-related samples

29b8c57226b70fc7e095bb8bed4611d923f0bcefc661ebae5182168613b497f8
66d44e2bc7495662d068051c5a687d17c7e95c8f04acb0f06248b34cd255cd25
fae77c173815b561ad02d8994d0e789337a04d9966dd27a372fd9055f1ac58b1
c1c56c7eb2f6b406df908ae822a6ea936f9cc63010ee3c206186f356f2d1aa94
4c5b8c3e0369eb738686c8a111dfe460e26eb3700837c941ea2e9afd3255981e
113d705d7736c707e06fb37ac328080b3976838d0a7b021fd5fb299896c22c7c
1a6c3e5643d7e22554ac0a543c87a2897ea4ea5a07bc080943a310a391e20713
0b860af58a9d2d7607f09022aa69508b0966a1cc8d953d3995a5fe07f8fabcac
5d73d14525ced5bdf16181f70f4d931b9c942c1ae16e318517d1cd53f4cd6ea9
c34ad273d836b2f058bbd73ea9958d272bd63f4119dacacc310bf38646ff567b
500c713aa82a11c4c33e9617cad4241fcef85661930e4986c205233759a55ae8
5f5acf76a991c1ca33855a96ec0ac77092f2909e0344657fe3acf0b2419d1eea
c6d96be46ce3d616e0cb36d53c4fade7e954e74bfd2e34f9f15c4df58fc732d2
d558bb63ed9f613d51badd8fea7e8ea5921a9e31925cd163ec0412e0d999df58
006e0cc29697db70b2d4319f320aa0e52f78bf876646f687aa313e8ba04e6992
2d9edf45988614f002b71899740d724008e9a808efad00fa79760b31e0a08073
3d2a7ea04d2247b49e2dcad63a179ae6a47237eddbfd354082f1417a63e9696e
ea46ed5aed900cd9f01156a1cd446cbb3e10191f9f980e9f710ea1c20440c781
f4113e30d50e0afc4fa610a3181169bb03f6766aea633ed8c0c0d1639dfc5b29
08203b4ddc9571418b2631ebbc50bea57a00eadf4d4c28bd882ee8e831577a19
a3992ed9a4273de53950fc55e5b56cc5b1327ffee59b1cea9e45679adc84d008
575028bbfd1c3aaff27967c9971176ae7038902f1a67d70def55ae8456e6166d
428cf6ec1a4c947b51ec099a656f575ce42f67737ee53f3afc3068a25adb4c0d
f53e3e0b3c524471b1f064aabd0f782802abb4e29534a1b61a6b25ad8ec30e79

Destover "b076e058" samples

Droppers:
6e93d7bdb01af596019fa48986544ca24aa06463f17975a084b28ce9ab3cf910
e0066ddc9e6f62e687994a05027e3eaa02f6f3ad6d71d16986b757413f2fb71c

Dropped components:
9ec83d39d160bf3ea4d829fa8d771d37b4f20bec3a68452dfc9283d72cee24f8
10d3ab45077f01675a814b189d0ac8a157be5d9f1805caa2c707eecbb2cbf9ac
33207f4969529ad367909e72e0f9d0a63c4d1db412e41b05a93a7184ec212af1
389ee412499fd90ef136e84d5b34ce516bda9295fa418019921356f35eb2d037
e0ce1f4b9ca61747467cee56307f9ea15dd6935f399837806f775e9b4f40e9ca
54ab7e41e64eb769b02b855504c656eaaff08b3f46d241cb369346504a372b4f
47830371f6f3d90d6a9fbe39e7f8d43a2e126090457448d0542fcbec4982afd6

Destover "Volgmer" samples

Droppers:
37dd416ae6052369ae8373730a9189aefd6d9eb410e0017259846d10ac06bff5
87db427b1b44641d8c13be0ba0a2b2f354493578562326d335edfeb998c12802
e40a46e95ef792cf20d5c14a9ad0b3a95c6252f96654f392b4bc6180565b7b11
53e9bca505652ef23477e105e6985102a45d9a14e5316d140752df6f3ef43d2d
8fcd303e22b84d7d61768d4efa5308577a09cc45697f7f54be4e528bbb39435b

Dropped components:
6dae368eecbcc10266bba32776c40d9ffa5b50d7f6199a9b6c31d40dfe7877d1
b987f7e6467704029c7784e9beb9ad3aa6e1375a661dc10b5f3d11c6a8fc1ef2
1d0999ba3217cbdb0cc85403ef75587f747556a97dee7c2616e28866db932a0d
9f177a6fb4ea5af876ef8a0bf954e37544917d9aaba04680a29303f24ca5c72c
78af649d3d6a932bcf53cfe384ce6bf9441f4d19084692b26b7e28b41f7a91bd
5d617f408622afc94b1ca4c21b0b9c3b17074d0fcd3763ee366ab8b073fc63e9
fee0081df5ca6a21953f3a633f2f64b7c0701977623d3a4ec36fff282ffe73b9
c5946116f648e346b293e2e86c24511a215ebe6db51073599bba3e523fb0d0a8
eab55bded6438cd7b8a82d6447a09bba078ded33049fca22d616a74bb2cad08f
ff2eb800ff16745fc13c216ff6d5cc2de99466244393f67ab6ea6f8189ae01dd

Destover "Windowsupdatetracing" samples

Droppers:
83e507104ead804855d07bc836af4990542d1eac5ac2a8ce86f985d082199f6f
d94ceade521452864ae8daae9d6b202a79d4761f755c7c769ec4e103c7c3127d
bebf6266e765f7a0eefcde7c51507cc9f6e3b5d5b82a001660454e4e84f6e032
4166f6637b3b11f69cccbeb775f9ee6987a5a30475c54db189b837ee3fbbf0d1
eeb146ebbc3f144f5a6156d07322a696eead9c4895a9a6f94212d24056acd41c

Destover "Messagethread" samples

Droppers, var A
6959af7786a58dd1f06d5463d5ba472396214d9005fce8559d534533712a9121
68006e20a2f37609ffd0b244af30397e18df07483001150bcc685a9861e43d44
d8fedef123b3d386f0917f11db9fae0956ffe5b16a9aaad8805f72309437d066

Droppers, var B
2368ee0e0001599b7789d8199c7b19f362a87925118ae054309d85f960d982ec
6e3db4da27f12eaba005217eba7cd9133bc258c97fe44605d12e20a556775009
98abfcc9a0213156933ccd9cb0b85dc51f50e498dbfdec62f6a66dc0660d4d92
d36f79df9a289d01cbb89852b2612fd22273d65b3579410df8b5259b49808a39

Destover "b8ac0905"

X86 Service DLL sample:
696ff9dda1ce759e8ff6dd96b04c75d232e10fe03809ba8abac7317f477f7cf5

Destover "b59d1659"

X64 Service DLL sample:
7501c95647cef0c56e20c6d6a55de3d23f428e8878a05a603a0b37ea987a74e2

Destover "e4004c1f"

HWP dropper documents:
3c3d2ab255daa9482fd64f89c06cdbfff3b2931e5e8e66004f93509b72cf1cc7
7d9631a62ae275c58e7ad2a3e5e4c4eac22cff46c077410ad628be6c38dd5e08

Dropper executables:
ca4b4a3011947735a614a3dc43b67000d3a8deefb3fffa95b48f1d13032f2aea
31a76629115688e2675188d6f671beacfe930794d41cf73438426cc3e01cebae

Dropped components:
7cea18dce8eb565264cc37bfa4dea03e87660b5cea725e36b472bafdcfe05ab1
757cd920d844fdcb04582a89b55f62b9a3e9bf73804abf94c9a9e15d06030b93
8a4f000049ad2a6c4eeac823c087b1c6e68c58b241c70341821cceccdf0f2d17
0654d112c17793c7a0026688cee569e780b989a9eb509585a977efd326dc2873
453d8bd3e2069bc50703eb4c5d278aad02304d4dc5d804ad2ec00b2343feb7a4
1f689996439db60970f4185f9cfc09f59bfe92650ba09bda38c7b1074c3e497b

Destover "Duuzer" samples

X86 samples:

029f93b7b7012777ee9fb2878d9c03b7fc68afad0b52cdc89b28a7ea501a0365
5831e614d79f3259fd48cfd5cd3c7e8e2c00491107d2c7d327970945afcb577d
6b70aa88c3610528730e5fb877415bc06a16f15373c131284d5649214cd2e96b
9b4c90ca8906e9fea63c9ea7a725db5fc66e1ca6c2a20bec2e8c1749b0000af5
b0cfaab0140f3ea9802dc6ed25bf208a2720fb590733966b7a3e9264a93a4e66
b3c0b7e355bee34cdb73d0bbdb1ba1b61797c035db31f0c82b19f9aa6a7abcc7
36844e66e5f4d802595909e2cbe90a96ad27da6b254af143b6611ab9ee85a13e
4efeea9eeae3d668897206eeccb1444d542ea537ca5c2787f13dd5dadd0e6aaa
5b28c86d7e581e52328942b35ece0d0875585fbb4e29378666d1af5be7f56b46
66df7660ddae300b1fcf1098b698868dd6f52db5fcf679fc37a396d28613e66b
72008e5f6aab8d58e4c8041cde20ee8a4d208c81e2b3770dbae247b86eb98afe
822a7be0e520bb490386ad456db01f26c0f69711b4ac61ba2cb892d5780fe38f
899ff9489dde2c5f49d6835625353bfe5ea8ca3195ca01362987a9d4bdac162d
8b50d7d93565aab87c21e42af04230a63cd076d19f8b83b063ef0f61d510adc7
90d8643e7e52f095ed59ed739167421e45958984c4c9186c4a025e2fd2be668b
ac27cfa2f2a0d3d66fea709d7ebb54a3a85bf5134d1b20c49e07a21b6df6255a
c5be570095471bef850282c5aaf9772f5baa23c633fe8612df41f6d1ebe4b565
ce0e43c2b9cb130cd36f1bc5897db2960d310c6e3382e81abfa9a3f2e3b781d7
facb32efc05bc8c4f3cb3baa6824db0f7effc56c02dbc52c33bafe242a1def77
763d1cb589146dd44e082060053ffbf5040830c79be004f848a9593d6be124ac
02d1d4e7acd9d3ec22588d89aed31c9a9d55547ef74fa3749659b610893f5405
47181c973a8a69740b710a420ea8f6bf82ce8a613134a8b080b64ce26bb5db93
e187811826b2c33b8b06bd2392be94a49d068da7f703ae060ee4faffde22c2fe

X64 samples:

2811fdceb8a8aa03bbf59c0b01a43bd1f2aee675a8f20d38194258046987e5fa
39e53ba6984782a06188dc5797571897f336a58b8d36020e380aa6cd8f1c40a2
530a0f370f6f3b78c853d1e1a6e7105f6a0f814746d8a165c4c694a40c7ad09a
7a2a740d60bd082c1b50ab915ef86cc689ba3a25c35ac12b24e21aa118593959
eaea45f8bfb3d8ea39833d9dcdb77222365e601264575e66546910efe97cba99
ee49322ed9fb43a9a743b54cc6f0da22da1d6bc58e87be07fd2efe5e26c3ef8a
ef07d6a3eb4a0047248c845be3da3282c208ede9508a48dbb8128eacc0550edf
477ca3e7353938f75032d04e232eb2c298f06f95328bca1a34fce1d8c9d12023
5a69bce8196b048f8b98f48c8f4950c8b059c43577e35d4af5f26c624140377c
89b25f9a454240a3f52de9bf6f9a829d2b4af04a7d9e9f4136f920f7e372909b
a01bd92c02c9ef7c4785d8bf61ecff734e990b255bba8e22d4513f35f370fd14
b93793e3f9e0919641df0759d64d760aa3fdea9c7f6d15c47b13ecd87d48e6a9
d589043a6f460855445e35154c5a0ff9dbc8ee9e159ae880e38ca00ea2b9a94f

Destover "Randomdomain" samples

X86 samples:

92cc25e9a87765586e05a8246f7edb43df1695d2350ed921df403bdec12ad889
f2a14c5ef6669d1eb08fababb47a4b13f68ec8847511d4c90cdca507b42a5cf3
520778a12e34808bd5cf7b3bdf7ce491781654b240d315a3a4d7eff50341fb18
e55fff05de6f2d5d714d4c0fa90e37ef59a5ec4d90fdf2d24d1cb55e8509b065
e506987c5936380e7fe0eb1625efe48b431b942f61f5d8cf59655dc6a9afc212
2477f5e6620461b9146b32a9b49def593755ac9788fc4beeee81bf248aa2e92a
f69747d654acc33299324e1da7d58a0c8a4bd2de464ec817ad201452a9fa4b54
44884565800eebf41185861133710b4a42a99d80b6a74436bf788c0e210b9f50
2f629c3c65c286c7f55929e3d0148722c768c730a7d172802afe4496c0abd683
b5e1740312b734fb70a011b6fe52c5504c526a4cccb55e154177abe21b1441c9

X64 samples:

0e162a2f07454d65eaed0c69e6c91dd10d29bdb27e0b3b181211057661683812
a53e33c77ecb6c650ee022a1311e7d642d902d07dd519758f899476dbaae3e49
c95eaedaafd8041bb0fea414b4ebc0f893f54cdec0f52978be13f7835737de2a
da2558662466895724 74d13d3408c954b17d4cc969c45d6f45827799e97ed116
8465138c0638244adc514b2722fcb60b2a26a8756aa7d97f150e9bdc77e337cc

Destover "FormBoundary" sample

77a32726af6205d27999b9a564dd7b020dc0a8f697a81a8f597b971140e28976

Destover "BasicHwp" samples

HWP dropper document:

794b5e8e98e3f0c436515d37212621486f23b57a2c945c189594c5bf88821228

Droppers:

c248da81ba83d9e6947c4bff3921b1830abda35fed3847effe6387deb5b8ddbb
794b5e8e98e3f0c436515d37212621486f23b57a2c945c189594c5bf88821228
fba0b8bdc1be44d100ac31b864830fcc9d056f1f5ab5486384e09bd088256dd0

Dropped components:

c3f5e30b10733c2dfab2fd143ca55344345cc25e42fbb27e2c582ba086fe3326

Destover "Volgmer2" samples

Droppers:

1ee75106a9113b116c54e7a5954950065b809e0bb4dd0a91dc76f778508c7954
f71d67659baf0569143874d5d1c5a4d655c7d296b2e86be1b8f931c2335c0cd3

Dropped components:

96721e13bae587c75618566111675dec2d61f9f5d16e173e69bb42ad7cb2dd8a

APPENDIX: C&C DATA

Joanap-related C&C addresses

110.164.115.177
118.102.187.188
118.70.143.38
119.15.245.179
122.55.13.34
168.144.197.98
189.114.147.186
196.44.250.231
201.222.66.25
60.251.197.122
62.135.122.53
62.150.4.42
62.87.153.243
63.131.248.197
63.149.164.98
64.71.162.61
66.210.47.247
69.15.198.186
72.156.127.210
75.145.139.249
78.38.221.4
80.191.114.136
81.130.210.66
81.83.10.138
83.211.229.42
92.253.102.217
92.47.141.99
93.62.0.22
94.28.57.110
96.39.78.157

Volgmer C&C addresses (dynamic normal, hardcoded bold)

103.16.223.35
113.28.244.194
116.48.145.179
117.239.214.162
12.217.8.82
123.176.38.17
123.176.38.175
134.121.41.45
186.116.9.20
186.149.198.172
190.210.39.16
195.28.91.232
199.15.234.120
200.42.69.13
200.42.69.133
203.131.222.99
206.123.66.136
206.163.230.170
212.33.200.86
213.207.142.82
220.128.131.251
24.242.176.130
41.21.201.101
64.3.218.243
78.93.190.70
83.231.204.157
84.232.224.218
89.122.121.230
89.190.188.42
200.87.126.116
194.224.95.20

Destover "MessageThread" C&C IP addresses:

101.76.99.183
112.206.230.54
124.47.73.194
165.138.120.35
175.45.4.158
177.189.204.214
187.176.34.40
202.182.50.211
203.131.222.102
208.105.226.235
209.237.95.19
211.76.87.252
213.42.82.243
31.210.53.11
59.125.119.135
59.125.62.35
61.91.100.211
62.141.29.175
65.117.146.5
71.40.211.3
85.112.29.106
91.183.41.5
93.157.14.154

Destover "WindowsUpdateTracing" real C&C IP addresses (after XOR translation). Addresses in red are inferred from pDNS only (no sample).

1.202.129.201
110.78.165.32
113.10.158.4
124.81.92.85
140.134.23.140
196.36.64.50
199.83.230.236
201.22.95.127
202.9.100.206
185.20.218.28
200.55.243.150
122.179.175.224
124.123.219.216
108.166.93.13
14.141.129.116
217.128.80.228
58.137.122.226
2.224.202.27
14.2.240.20
59.125.75.217
41.38.151.7
201.203.27.170
64.206.243.35
184.180.159.183
24.77.32.241
64.228.222.61
217.8.95.250
180.26.59.158
41.41.29.214

Destover "RandomDomain" C&C IP addresses:

103.233.121.22
187.111.14.62
187.54.39.210
206.248.59.124
37.34.176.14
94.199.145.55
200.202.169.103
202.152.17.116
203.131.210.247

Destover "Duuzer" C&C IP addresses:

110.77.140.155
113.160.112.125
114.143.184.19
148.238.251.30
161.139.39.234
161.246.14.35
175.111.4.4
177.0.154.88
177.19.132.216
177.52.193.198
184.173.254.54
185.20.218.28
185.30.198.1
185.81.99.17
186.167.17.115
194.165.149.51
196.202.33.106
200.87.126.117
201.163.208.37
202.39.254.231
203.113.122.163
203.115.13.105
203.170.66.206
210.211.124.229
223.255.129.230
31.210.54.14
37.148.208.67
37.58.148.34
41.21.201.107
41.76.46.182
5.22.140.93
62.0.79.45
67.229.173.226
78.38.114.213
87.101.243.246
90.80.152.49
203.132.205.250
59.90.208.171
201.25.189.114

Destover "BasicHwp" C&C IP addresses:

91.183.71.18
184.20.197.204
208.87.77.153
201.216.206.49
87.101.243.252
208.69.30.151
69.54.32.30

Destover "Volgmer2" C&C IP addresses:

121.170.194.185
222.236.46.5

APPENDIX: YARA RULES

```
rule Destover : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Used for attacks on Sony Pictures Entertainment and targets in South Korea"
	strings:
		$a1= "recdiscm32.exe"
		$a2= "taskhosts64.exe"
		$a3= "taskchg16.exe"
		$a4= "rdpshellex32.exe"
		$a5 ="mobsynclm64.exe"
		$a6 ="comon32.exe"
		$a7 ="diskpartmg16.exe"
		$a8 ="dpnsvr16.exe"
		$a9 ="expandmn32.exe"
		$a10="hwrcompsvc64.exe"
		$a12="cmd.exe /c wmic.exe /node:\"%s\" /user:\"%s\" /password:\"%s\" PROCESS CALL CREATE \"%s\" > %s"
		$a13="#99E2428CCA4309C68AAF8C616EF3306582A64513E55C786A864BC83DAFE0C78585B692047273B0E55275102C66"
		$a14="b8ac0905cda0360fc115f614119da76d84e2277762bd7558b2650a79013fb50138f732d5a03730d7d5b17"
		$a15="b076e0580463a202bad74cb9c1b85af3fb4d1be513ccca3ae8b57d193be77b4ab63802b3216d3a80b0082"
		$a16="bc9b75a31177587245305cd418b8df78652d1c03e9da0cfc910d6d38ee4191d40bd51483321ebe44595f7"
		$a17="b50a338264226b6d57c1936d9db140ba74a28930270a083353645a9b518661f4fcea160d73469b8beabc1"
		$a18="b59d165982e3d5721c4d40195f85aedf2a12d6616be11a2c19fa11821604edc4675bdca4f9b9cbfb27244"
		$a19="e4004c1f94182000103d883a448b3f802ce4b44a83301270002c20d0321cfd0011ccef784c26a400f43df"
		$b1 = "---------------End--------------!"
		$b2 = "WaitRecv End" wide
	condition:
		any of ($a*) or all of ($b*)
}

rule Destover2 : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Used for attacks on Sony Pictures Entertainment and targets in South Korea"
	strings:
		$a1 = "%sd.e%sc" fullword ascii wide
		$a2 = "xe" fullword ascii wide
		$a3 = "cm" fullword ascii wide
		$b1 = "%smd.e%sc" fullword ascii wide
		$c1 = "%sm%se%sc" fullword ascii wide
		$d = "ChfTime Success" ascii wide
		$e = {FF15????????6A3EFF75??FF15????????5985C0598D85????????50FF75??68????????68????????75}
		$f = "%s \"%s > %s 2>&1\"" ascii wide

	condition:
		all of ($a*) or ($b1 and $a2) or ($c1 and $a2) or $d or $e or $f
}

rule DarkSeoul_Obf_ChopString : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Obfuscation method used by the DarkSeoul group"
	strings:
		$a1={8B54240456BE????????57B91400000033C08BFEF3AB803A0074158A023C2E74073C2074038806468A42014284C075EB}
	condition:
		any of them
}


rule DarkSeoul_Obf_BCSUB : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Obfuscation method used by the DarkSeoul group"
	strings:
		$a1="pM[XpSZJ[JC{"
	condition:
		any of them
}

rule DarkSeoul_Obf_XORA7 : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Obfuscation method used by the DarkSeoul group"
	strings:
		$a1={E0C2D3F7D5C8C4E6C3C3D5C2D4D4}
	condition:
		any of them
}
```

```
rule DarkSeoul_Obf_Caracachs : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Obfuscation method used by the DarkSeoul group"
	strings:
		$a1={F3EEAEFFFBB821BF9AE3D820FDC0}
	condition:
		any of them
}


rule DarkSeoul_Keystrings : Backdoor
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "Encryption keys used by the DarkSeoul group"
	strings:
		$a1 = "Bb102@jH4$t3hg%6&G1s*2J3gCNwVr*UeI!Dr3hytg^CHGf%ion"
		$a2 = "BAISEO%$2fas9vQsfvx%$"
		$a3 = "A39405WKELsdfirpsdLDPskDORkbLRTP12330@3$223%!"
	condition:
		any of them
}


rule Joanap :
{
	meta:
		author = "Blue Coat Systems, Inc."
		info = "SMB worm family used by the DarkSeoul group"
	strings:
		$a1="NTLMSSP"
		$a2="MiniDumpWriteDump"
		$a3="password <=14"
		$a4="KGS!@#$%"
		$b1="9025jhdho39ehe2"
		$b2="y@s!11yid60u7f!07ou74n001"
		$b3=“y0uar3@s!11yid!07,ou74n60u7f001”
	condition:
		all of ($a*) or any of ($b*)
}
```