“Winnti”
More than just a game

Kaspersky Lab Global Research and Analysis Team

April 2013

Contents

Executive Summary .................................................................................................................. 2
Winnti 1.0 Technical Analysis.................................................................................................. 10
Real Case Investigation (Winnti 2.0) ................................................................................. 26
Source of Attacks ................................................................................................................. 51
The Search for Attackers (XYZ incident) ........................................................................... 53
Conclusions ......................................................................................................................... 80
Appendix ............................................................................................................................ 81
Executive Summary

This research, which started in autumn of 2011 by Kaspersky Lab, is still ongoing. The subject of this research project is a series of targeted attacks against private companies around the world.

In the research, we reveal activity of one of the hacking groups which has Chinese origins. This group was named "Winnti".

According to our estimates, the Winnti group has been active for several years and specializes in cyber-attacks against the online video game industry. The main objective of the group is to steal source code of online game projects as well as digital certificates of legitimate software vendors. Besides that, they are deeply interested in the set-up of network infrastructure (including production gaming servers) and new developments such as conceptual ideas, design and more.

We aren’t the first to investigate the attacks attributed to the Winnti group. It is known that, at least in 2010, the U.S.-based company HBGary investigated information security incidents related to the Winnti group at one of HBGary’s customers – an American video game production company.

In the beginning ...

In the autumn of 2011, a Trojan was detected on a large number of computers – all of them linked by the fact that they were used by players of a popular online game. It emerged that the piece of malware landed on users’ computers as part of a regular update from the game’s official update server. Some even suspected that the publisher itself was spying on its customers. However, it later became clear that the malicious program ended up on the users’ computers by mistake: the cybercriminals were in fact targeting the companies that develop and release computer games.

The computer game publisher whose servers spread the Trojan asked Kaspersky Lab to analyze the malicious program that was found on its update server. It turned out to be a DLL library compiled for a 64-bit Windows environment and even used a properly signed malicious driver.

The malicious DLL infected gamers’ computers running under either 32-bit or 64-bit operating systems. It could not start in 32-bit environments, but it could, under certain conditions, launch without the user’s knowledge or consent in 64-bit environments, though no such accidental launches have been detected.

The DLL contained a backdoor payload, or, to be exact, the functionality of a fully-fledged Remote Administration Tool (RAT), which gave the cyber-criminals the ability to control the victim computer without the user’s knowledge.

The malicious module turned out to be the first Trojan for the 64-bit version of Microsoft Windows with valid digital signature that we have seen. We used to see similar cases before, but in all previous incidents we have seen digital signature abuse, there were only 32-bit applications.

At an early stage of our research, we identified a number of similar backdoors, both 32-bit and 64-bit, in our collection of malware samples. These were detected under various verdicts. We grouped them together into a separate family. Symantec appears to be the first to name these malicious programs; we kept Symantec’s name –
Winnti – in the name of the malware family we created: Backdoor.Win32(Win64).Winnti. As for the people behind these attacks involving this remote administration tool, we ended up calling them “the Winnti group”.

Interestingly, the digital signature belonged to another video game vendor - a private company known as KOG, based in South Korea. The main business of this company was MMRPG (massively multi player online role-playing games) games, which was identical to the business area of the first victim.

We contacted KOG, whose certificate was used to sign malicious software and notified Verisign, which issued the certificate for KOG. As a result, the certificate was revoked.

Digital Certificates

When we discovered the first stolen digital certificate, we didn’t realize that stealing the certificates and signing malware for upcoming attacks against other victims was the modus operandi of that group. In eighteen months, we manage to discover more than a dozen compromised digital certificates.

Moreover, we found that those digital certificates seemed to have been used in attacks organized by other hacking groups, presumably coming from China.

For example, an attack against South Korean social networks Cyworld and Nate in 2011 (http://www.bbc.co.uk/news/technology-14323787) - the attackers used a Trojan that was digitally signed using the certificate of YNK Japan Inc gaming company.

A digital certificate of the same company was used recently (March 2013) in Trojans targeting Tibetan and Uyghur activists (https://www.securelist.com/en/blog/208194165/New_Uyghur_and_Tibetan_Themed_Attacks_Using_PDF_Exploits).

In fact, this story has long roots dating back to 2011. We highly recommend reading this Norman blog post of a similar incident here: http://blogs.norman.com/2011/security-research/invisible-ynk-a-code-signing-conundrum.
At the same time, in March 2013, Uyghur activists were targeted by another malware which was digitally signed by another gaming company called MGAME Corp according to http://www.f-secure.com/weblog/archives/00002524.html

We believe that the source of all these stolen certificates is same group which we call Winnti. This group either has close contacts with other Chinese hacker groups or sells the certificates on the black market in China.

Below is the list of known compromised companies and digital certificates used by the Winnti group in different campaigns:

<table>
<thead>
<tr>
<th>Company</th>
<th>Serial number</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTsoft Corp</td>
<td>30 d3 fe 26 59 1d 8e ac 8c 30 66 7a c4 99 9b d7</td>
<td>South Korea</td>
</tr>
<tr>
<td>Kog Co., Ltd.</td>
<td>66 e3 f0 b4 45 9f 15 ac 7f 2a 2b 44 99 0d d7 09</td>
<td>South Korea</td>
</tr>
<tr>
<td>LivePlex Corp</td>
<td>1c aa 0d 0d ad f3 2a 24 04 a7 51 95 ae 47 82 0a</td>
<td>South Korea/Philippines</td>
</tr>
<tr>
<td>MGAME Corp</td>
<td>4e eb 08 05 55 f1 ab f7 09 bb a9 ca e3 2f 13 cd</td>
<td>South Korea</td>
</tr>
<tr>
<td>Rosso Index KK</td>
<td>01 00 00 00 00 01 29 7d ba 69 dd</td>
<td>Japan</td>
</tr>
<tr>
<td>Sesisoft</td>
<td>61 3e 2f a1 4e 32 3c 69 ee 3e 72 0c 27 af e4 ce</td>
<td>South Korea</td>
</tr>
<tr>
<td>Wemade</td>
<td>61 00 39 d6 34 9e e5 31 e4 ca a3 a6 5d 10 0c 7d</td>
<td>Japan/South Korea/US</td>
</tr>
<tr>
<td>YNK Japan</td>
<td>67 24 34 0d db c7 25 2f 7f b7 14 b8 12 a5 c0 4d</td>
<td>Japan</td>
</tr>
<tr>
<td>Guangzhou YuanLuo</td>
<td>0b 72 79 06 8b eb 15 ff e8 06 0d 2c 56 15 3c 35</td>
<td>China</td>
</tr>
<tr>
<td>Fantasy Technology Corp</td>
<td>75 82 f3 34 85 aa 26 4d e0 3b 2b df 74 e0 bf 32</td>
<td>China</td>
</tr>
<tr>
<td>Neowiz</td>
<td>5c 2f 97 a3 1a bc 32 b0 8c ac 01 00 59 8f 32 f6</td>
<td>South Korea</td>
</tr>
</tbody>
</table>

** Victims **

It’s tempting to assume that Advanced Persistent Threats (APT)s primarily target high-level institutions: government agencies, ministries, military and political organizations, power stations, chemical plants, critical infrastructure networks, and so on. In this context, it seems unlikely that a commercial company would be at risk unless it was operating on the scale of Google, Adobe or The New York Times, which was recently targeted by a cyber-attack, and this perception is reinforced by the publicity that attacks on corporations and government organizations usually receive. However, any company with data that can be effectively monetized is at risk from APTs. This is exactly what we encountered here: it was not a governmental, political, military, or industrial organization. The target was specifically gaming companies.

Analyzing the Winnti samples helped to identify who and what were the targets. We found that we were dealing with targeted attacks: the Winnti team infects companies that develop and release computer games. It appears the team has been active for quite a while – since 2009.
It’s difficult to name all the victims of the Winnti team. Judging by the information that we have at our disposal – namely the tags within malicious programs, the names of the C&C domains, the companies whose digital certificates were stolen to sign malware, and the countries where detection notifications came from – we can say that at least 35 companies were infected by the Winnti malware at some time.

Countries where the affected companies are located:

<table>
<thead>
<tr>
<th>Asia</th>
<th>Europe</th>
<th>South America</th>
<th>North America</th>
</tr>
</thead>
<tbody>
<tr>
<td>China, India</td>
<td>Belarus, Germany, Russia</td>
<td>Brazil, Peru</td>
<td>USA</td>
</tr>
<tr>
<td>Indonesia, Japan, Philippines, S. Korea, Taiwan, Thailand, Vietnam</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This data demonstrates that the Winnti team targets gaming companies located in various parts of the world, albeit with a strong focus on South East Asia.

Countries where gaming companies have been affected

This geographic diversity is hardly surprising. Often, gaming companies (both publishers and developers) are international, having representatives and offices worldwide. Also, it is common practice for gaming companies from various regions to cooperate. The developers of a game may be located in a different country from its publisher. When a game eventually reaches markets in regions away from its initial ‘home’, it is often localized and released by other publishers. In the course of this cooperation, the partner companies often grant each other access to network resources to exchange data associated with the gaming content, including distribution kits,
gaming resources, resource assembly kits, etc. If one company in the network gets infected, it’s easy for the cybercriminals to spread the infection throughout the partnership chain.

**Winnti C&C Structure**

During the investigation, we identified more than a hundred malicious programs, each individually compiled to attack a particular company. Typically, separate command-and-control (C&C) domains were assigned to each targeted company. Virtually all the C&C domains were arranged as follows: a second-level domain was created without a DNS A-record, i.e., there was no IP address assigned to it.

In cases where there was an A-record, the assigned IP address was typically 127.0.0.1. It is also noteworthy that some of the second-level domains that the cybercriminals created for their C&C had very similar names to the domain hosting the site of a certain real gaming company. And the malicious users’ domain was resolved to the same IP address which the site of the real gaming company used. In any case, the third-level domains resolved to IP addresses assigned to the attackers’ actual C&C servers.

<table>
<thead>
<tr>
<th>domain.com: no A-record</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain.com: A-record = 127.0.0.1</td>
</tr>
</tbody>
</table>

**C&C domain naming and resolution**

<table>
<thead>
<tr>
<th>Good</th>
<th>Bad</th>
</tr>
</thead>
<tbody>
<tr>
<td>legal-company-site.com: A-record → Good</td>
<td></td>
</tr>
<tr>
<td>legal-company-site.us: A-record → C&amp;C server</td>
<td></td>
</tr>
<tr>
<td>subdomain.legal-company-site.us: A-record → IP-address #2</td>
<td></td>
</tr>
</tbody>
</table>

Sometimes the Winnti team registered their C&C units with public hosts. Judging by the samples identified, these C&C centers were subdomains of such domains as 6600.org, 8866.org, 9966.org or ddns.net.

From the names of the C&C domains or subdomains, the attack targets or countries of residence could be guessed, as in:

ru.gcgame.info
kr.zzsoft.info
jp.xxoo.co
us.nhntech.com
fs.nhntech.com
as.cjinternet.us

The subdomains “ru”, “kr”, “jp” and “us” most probably mean that these C&C servers manage bots hosted on the computers of companies located in Russia, South Korea, Japan and the U.S. respectively, while “fs” and “as” are acronyms for the names of the companies being attacked.
Sometimes Winnti’s malicious programs had a local IP address, such as 192.168.1.136, specified in the settings for the C&C. This could mean that, at some point in time, there was an infected computer that did not have a connection to the Internet, but the cybercriminals needed control over it (it may have been infected while malware was spread via a corporate network). In this case, the cybercriminals deployed a dedicated local C&C server on another compromised computer within the same local network which did have an Internet connection; via that C&C, the first victim computer could be controlled. System administrators often try to isolate critical computers from the outside world. This decreases the probability of haphazard infection, but, apparently, does not always help in a targeted attack.

In the Winnti samples that were detected and analyzed, we found 36 unique C&C domains. Most probably, this is only a small portion of all existing Winnti C&C domains, as we only managed to obtain some of the samples from this malware family. This is hardly surprising since these malicious programs are used to execute targeted attacks, so no information is available about many instances of infection; for this reason, we have no way of obtaining samples of the malware used in these undisclosed attacks.

Domain names used in the attacks we discovered

- newpic.dyndns.tv
- update.ddns.net
- nd.jcrsoft.com
- cc.nexoncorp.us
- kr.zzsoft.info
- as.cjinternet.us
- ca.zzsoft.info
- sn.jcrsoft.com
- lp.apanku.com
- sshd.8866.org
- ftdd.6600.org
- tcpiah.googleclick.net
- rss.6600.org
- lp.zzsoft.info
- lp.gasoft.us
- eya.jcrsoft.com
- ftdd.9966.org
- kr.xxoo.co
- wi.gcgame.info
- tcp.nhntech.com
- ka.jcrsoft.com
- my.zzsoft.info
- jp.jcrsoft.com
- su.cjinternet.us
- vn.gcgame.info
- ap.nhntech.com
- ru.gcgame.info
- kr.jcrsoft.com
- wm.ibm-support.net
- fs.nhntech.com
- docs.nhnclass.com
- rh.jcrsoft.com
- wm.nhn.com
Knowing the 2\textsuperscript{nd} level domains used by Winnti, we brute forced through all third level sub-domains up to 4 symbols long, and identified those that have the IP addresses of real servers assigned to them. Having searched through subdomains for a total of 12 second level domains, we identified 227 “live” third level domains. Many of them are C&C servers for Winnti-class malware that have hitherto remained unidentified.

Analyzing the WHOIS data for the 12 second level domains, we found the following list of email addresses used for registration:

- evilsex@gmail.com
- jslee.jcr@gmail.com
- whoismydns@gmail.com
- googl3@live.com
- wzcc@cnkker.com
- apanku2009@gmail.com

For some of these domains, registration data proved to be the same as those for the domain google.com:

\textit{Registrant: Google Inc.}  
\begin{verbatim}  
1600 Amphitheatre Parkway  
Mountain View, California 94043  
United States  
+1.6503300100  
\end{verbatim}

Judging by the domain registration data, the Winnti team started their criminal activities as far back as 2007. The early domains were involved in spreading rogue anti-virus programs (FakeAV). From 2009 onwards, domains began to emerge hosting C&C servers for bots used to infect gaming companies. Apparently, the cybercriminals graduated to relatively large-scale penetrations into the corporate networks of gaming companies starting from 2010.

\textbf{Known Malware}

The favorite tool of the attackers is a malicious program we call "Winnti". It has evolved since the first use, but we divide all variants into two generations: 1.x and 2.x. Our publication describes both variants of this tool. The second generation (2.x) was used in one of the attacks that we investigated in the active stage and helped the victim to interrupt data transfer and isolate infections in a corporate network.

In addition to that, we observed usage of a popular backdoor known as PlugX, which is believed to have Chinese origins, however used only previously in attacks against Tibetan activists.
The Commercial Interest

As has been stated above, APTs can target any commercial company if cyber-criminals find a way to financially profit from the attack.

So what methods do cyber-criminals use to generate illicit earnings from attacks on gaming companies?

Based on the available information, we have singled out three main monetization schemes that could be used by the Winnti team.

- The unfair accumulation of in-game currency/“gold” in online games and the conversion of virtual funds into real money.
- Theft of source code from the online games server to search for vulnerabilities in games – often linked to point 1.
- Theft of source code from the server part of popular online games to further deploy pirate servers.

Let’s look at an example. During our investigation of an infection at a computer gaming company, we found that malware had been created for a particular service on the company’s server. The malicious program was looking for a specific process running on the server, injected code into it, and then sought out two places in the process code where it could conceal call commands for its function interceptors. Using these function interceptors, the malicious programs modified process data which was processed in those two places, and returned control back. Thus, the attackers change the normal execution of the server processes. Unfortunately, the company was not able to share its targeted application with us, and we cannot say exactly how this malicious interference affected gaming processes. The company concerned told us that the attackers’ aim was to acquire gaming “gold” illegally.

Malicious activity like this has an adverse impact on the game itself, tilting the balance in favor of cheats. But any changes the Winnti team introduces into the game experience are unlikely to be very noticeable. After all, maintaining a skillful balance is the main attribute of online games. Users will simply stop playing if they feel that other players are using non-standard methods to create an advantage beyond normal gameplay or if the game loses its intrinsic competitiveness due to resources or artifacts appearing in the game without the developers’ knowledge. At the same time, the attackers are keen for the game to remain popular; otherwise, they would be unable to effectively turn all the time and effort of infecting a gaming company into financial gain.

Members of the Winnti team are patient and cautious. Cyber-criminals have affected the processes of the online games from the infected companies and stolen money from them for years, but they have found ways of doing this without attracting attention to themselves.
Winnti 1.0 Technical Analysis

The Initial DLL

Everything starts with a DLL. The DLL mimics one of the standard Windows libraries, winmm.dll or apphelp.dll. Since, in the vast majority of cases the samples that we detected disguised themselves as winmm.dll, we would like to fix this name for this malicious library at the end of this document.

Legitimate winmm.dll is a Windows system library that provides multimedia functions. It is located in the %WINDIR%\System32 folder. The attackers counted on this being a library providing basic system functions and hence the probability of it being loaded by some program is very high (this is also valid for apphelp.dll). For example, winmm.dll is loaded by explorer.exe, which is launched during operating system startup and is essential for Windows user interface.

The mechanism to start the malware is simple: if some benign application depends on Windows winmm.dll (located in %WINDIR%\System32\winmm.dll) but the evil twin library with the same name (winmm.dll) is located in the folder of benign application, the malicious library will be loaded instead of the system one.

Taking advantage of their control of an infected computer, the attackers place a malicious library in the %WINDIR% folder. The same folder also contains explorer.exe. This enables the attackers to ensure that the malicious DLL is loaded at system startup: explorer.exe loads the malicious winmm.dll from the %WINDIR% folder as soon as it launches during system startup.

But how can a program which depends on the original library work correctly if a malicious winmm.dll is loaded instead of the original library? Very easy: the malicious library is designed to work as a proxy for the original winmm.dll from the %WINDIR%\System32 folder.

The cyber-criminals did not reinvent the wheel to make sure that everything works properly. They relied on a tool known as AheadLib, which was developed by security researchers to analyze malware.
This program, which is designed to facilitate the analysis of malicious libraries, was created by a Chinese developer employed by an Asian anti-virus vendor. The program accepts a DLL on input and produces a C code which hooks the functions included in the library. The C code is compiled back into a DLL, which can then be used as a proxy and provide flexible way to analyze behavior of malicious file.

```c
#include <Windows.h>

#pragma comment(linker, "EXPORT Noname2_AheadLib_Noname2_@2 NONAME")
#pragma comment(linker, "EXPORT CloseDriver_AheadLib_CloseDriver_@3")
#pragma comment(linker, "EXPORT DefDriverProc_AheadLib_DefDriverProc_@4")
#pragma comment(linker, "EXPORT DriverCallback_AheadLib_DriverCallback_@5")
#pragma comment(linker, "EXPORT ExportDriverCallback_AheadLib_MigrateSoundEvents_@9")
#pragma comment(linker, "EXPORT NotifyCallbackData_AheadLib_MigrateSoundEvents_@10")
#pragma comment(linker, "EXPORT OpenDriver_AheadLib_OpenDriver_@11")
#pragma comment(linker, "EXPORT PlaySound_AheadLib_PlaySound_@12")
#pragma comment(linker, "EXPORT PlaySoundA_AheadLib_PlaySoundA_@13")
#pragma comment(linker, "EXPORT PlaySoundW_AheadLib_PlaySoundW_@14")
#pragma comment(linker, "EXPORT SendDriverMessages_AheadLib_SendDriverMessages_@15")
```

This program, which is designed to facilitate the analysis of malicious libraries, was created by a Chinese developer employed by an Asian anti-virus vendor. The program accepts a DLL on input and produces a C code which hooks the functions included in the library. The C code is compiled back into a DLL, which can then be used as a proxy and provide flexible way to analyze behavior of malicious file.
Hook functions (code generated by the legitimate program AheadLib)

The flexibility of this tool allows to customize the logics of malicious application during analysis and overload functions code to provide some debugging output. Some code can be added to display parameters of the hooked functions in order to find out which values are passed to the original functions when they are called. This method is used in so called dynamic analysis of malicious applications.
Determining the addresses of the real functions
(error message in the frame: “Function %hs cannot be found, the program will not operate correctly”)

```c
// 获取原始函数地址
FARPROC WINAPI GetProcAddress(PCSTR pszProcName)
{
    FARPROC fpAddress;
    CHAR szProcName[18];
    TCHAR tzTemp[MAX_PATH];

    if (m_hModule == NULL)
    {
        if (Load() == FALSE)
        {
            ExitProcess(-1);
        }
    }

    fpAddress = GetProcAddress(m_hModule, pszProcName);
    if (fpAddress == NULL)
    {
        if (HIWORD(pszProcName) == 0)
        {
            wsprintf(szProcName, "%d", pszProcName);
            pszProcName = szProcName;
        }

        wsprintf(tzTemp, TEXT("无法找到函数 %hs, 程序无法正常运行。"), pszProcName);
        MessageBox(NULL, tzTemp, TEXT("AhoadLib"), MB_ICONSTOP);
        ExitProcess(2);
    }

    return fpAddress;
}
```
Ironically, the malware authors have found this to be a convenient application for creating malicious proxy-libraries. They specified a system library (winmm.dll) as a parameter for AheadLib tool and produced a source code template to create a proxy DLL – in the form of C file. By overloading some functions with the malicious payload, the attackers created a complete piece of malware that included all the features of the system DLL.

Strangely, the attackers kept the code for AheadLib debug messages in the early versions of their malware (marked with red in the screenshots above). These strings can also be found in compiled malicious binaries:

```
// 加载原始模块
inline BOOL WINAPI Load()
{
    TCHAR tzPath[MAX_PATH];
    TCHAR tzTemp[MAX_PATH * 2];
    GetSystemDirectory(tzPath, MAX_PATH);
    _strcat(tzPath, TEXT("c:\windows\system32\winmm.dll");
    m_hModule = LoadLibrary(tzPath);
    if(m_hModule == NULL)
    {
        wsprintf(tzTemp, TEXT("无法加载 %s，程序无法正常运行。")), tzPath);
        MessageBox(NULL, tzTemp, TEXT("AheadLib"), MB_ICONSTOP);
    }
    return (m_hModule != NULL);
}
```

Modified module loading the original DLL
(error message in the frame: “%s cannot be loaded, the program will not operate correctly”)

The function %hs cannot be found, the program will not operate correctly
Later, these fragments were removed from the C file generated by AheadLib.

**Control DLL**

The winmm.dll malicious library maintains another library in its body, which is decrypted and loaded into the process memory without creating any files on local disk. According to file version info the original name of this library is "PlusDLL.dll". This is the platform’s main control component. When the additional DLL has been properly allocated in the memory, winmm.dll passes control to it with a parameter – a string which contains bot settings. The settings string, in encrypted form, is also located in the winmm.dll body – after the magic word "PLUSUNIT".

After decryption, the string contains the following:

```
url=lp.gasoft.us:80|ver=1018|tag=33|group=lp80wi
```

Apparently, when the Winnti malware managed to get into focus of security researchers: the authors made modifications of the methods used to store these initial settings. In some samples, the settings were hidden even in the executable file’s header:
Encrypted settings in the header of malicious executable

In other variants, the ‘PLUSUNIT’ magic string was modified:

```
UUUSUN””” instead of PLUSUNIT
```

The PlusDLL library has an embedded driver. The driver is stored in %WINDIR%\System32\<drivername.sys> file, registered as a service and started by NtLoadDriver system API function. Immediately after that, the driver’s file is removed, as well as all the registry entries created during service registration. The executable preserved the original driver names which are "PortLess" and "PointFilter"; however, the driver files used during infection are saved as "sp1itter.sys" and "acplec.sys".

The purpose of the driver is to hide network connections established by the malware. For example, if the user decides to check a list of established connections (e.g., using the ‘netstat –a’ command or the tcpview program) while the bot is communicating to the control center, the driver will protect and hide the malware connections. This approach is used by many rootkits on the Windows platform.

The driver uses an interesting method to get the list of addresses to protect connections with. This information is available in the PlusDLL control library, which normally operates in the context of the explorer.exe process when the infection is active on the computer. The address information is sent from the user space (from PlusDLL) to the kernel space, where the driver works, via call to NtSetQuotaInformationFile API function.
During initialization, the driver hooks the NtSetQuotaInformationFile function:

```
nt\NtSetQuotaInformationFile:
805ff93e  mov  edi edi
805ff940  push  ebp
805ff941  mov  ebp esp
805ff943  push  0
805ff945  push  dword ptr [ebp+14h]
805ff948  push  dword ptr [ebp+10h]
805ff94b  push  dword ptr [ebp+0Ch]
805ff94e  push  dword ptr [ebp+8]
805ff951  call nt\!IoRaiseHardError+0xe8 (804ee9ae)
805ff956  pop  ebp
805ff957  ret  10h
805ff95a  int   3
```

Every time the function is called, the driver checks its parameters: to be precise it is HANDLE FileHandle and PVOID Buffer parameters.

The FileHandle parameter holds a descriptor of the partition on the hard drive where the function is expected to set disk quotas.

The Buffer parameter is a memory buffer with information of new quotas to be set. The driver checks whether the value of the FileHandle parameter is equal to minus two. When the system calls the NtSetQuotaInformationFile function to actually change the quotas, the descriptor must be associated with one of the disks. Normally such descriptors in the Windows system are positive integers which obviously means that it cannot be equal to minus two. The negative value is set by the PlusDLL library in order to make the driver detect that the NtSetQuotaInformationFile function was called by that library. When calling NtSetQuotaInformationFile, PlusDLL sends information about the network addresses to be protected by the driver via the Buffer parameter. If FileHandle is not equal to minus two, the hook function in the driver passes control to system’s original code of NtSetQuotaInformationFile API function and everything works as it should be on an uninfected system.
Note that 64-bit versions of Windows do not allow unsigned drivers to run. The malicious driver’s 64-bit versions were signed using stolen certificates. During the time that we have been tracking the Winniti group, we found 11 certificates that were used to sign the malware used by the group (not necessarily drivers only). Ten of them belong to various companies in the gaming industry.

**Launching the main function**

As mentioned above, the PlusDLL library is a control module. Let’s look at how the cybercriminals implemented the transition to perform the malicious DLL’s main tasks. They could have simply called an appropriate function directly or created a separate thread in which to execute it, but for some reason they resorted to a trick: the code of the SetWindowStationUser function in the user32.dll library was modified. After modification, the function’s first command became jmp <addr>, where <addr> is the address of the function in the PlusDLL library which implements the malicious library’s main features.
Immediately after this modification, a thread is created (CreateThread) executing code starting from the SetWindowStationUser function address. As a result, when control is eventually passed to this function, the inserted command jmp <addr> returns control back to the PlusDLL code.
Malicious DLL launching its own code by creating a thread that supposedly calls SetWindowStationUser

The same method is used to execute two more functions in the PlusDLL library. One of them is used to initialize network routines; the other executes procedures terminating the malicious program at the very end. The only difference is that instead of SetWindowStationUser, the code of two other functions from user32.dll is modified – EndTask and WinHelpW, respectively.

It is likely that this was done in order to hide the real addresses of functions in PlusDLL in case its code was analyzed based on its execution logs using an automatic system (sandbox) that looks at all function calls. If this trick is used, an execution log would only show threads launched from the addresses of the functions SetWindowStationUser, EndTask and WinHelpW, which could potentially confuse researchers.

Another possibility is that this is an anti-emulation feature. Perhaps the emulators built into some anti-virus products are unable to cope with these ‘leaps’ – in this case, emulation will not result in the execution of malicious functions, which also suits the cybercriminals’ purposes.

**Target Functionality**

So what does PlusDLL control? It turns out that the target functionality is implemented in different files. Each file provides a specific remote control feature and is downloaded from the attackers’ server every time the system starts up. These files are not saved on disk or in the registry but are loaded directly into the memory.

At the very start of the operation, after launching the driver, PlusDLL collects information about the infected system. A unique identifier for the infected computer is generated based on information about the hard drive and the network adapter’s MAC address, e.g., TKVFPP-XZTTL-KXFWWH-RBJLF-FXWJR. The attackers are interested primarily in the computer’s name, the program which loaded the malicious library, as well as information about
remote desktop sessions (session name, client name, user name and session time). All of this data is collected in a buffer, which is then compressed and sent to the attackers’ control center. The buffer may look like this:

<table>
<thead>
<tr>
<th>Plugin Name</th>
<th>Plugin Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>CmdPlus</td>
<td>Provide access to the system command line.</td>
</tr>
<tr>
<td>ListFileManager</td>
<td>Provide access to the file system: list directory contents, manipulate files.</td>
</tr>
<tr>
<td>ListProc</td>
<td>List or kill running processes.</td>
</tr>
<tr>
<td>ListService</td>
<td>List system services.</td>
</tr>
<tr>
<td>PortMap</td>
<td>Redirect traffic using port forwarding.</td>
</tr>
<tr>
<td>RemoteDesktop</td>
<td>Enable Remote Desktop service on the infected machine.</td>
</tr>
<tr>
<td>Socks5Client</td>
<td>Library for transferring data over the network using a SOCKS5 proxy server.</td>
</tr>
<tr>
<td>TransPlus</td>
<td>Enables the attacker to transfer files: receive files from the infected machine, download/create/save files, as well as execute programs on the infected computer.</td>
</tr>
</tbody>
</table>

The bot sends information about an infected system to the control center

In reply to this initial message from the bot, the control center sends the list of available plugins. Plugins are DLL libraries that provide specific remote control functions. Upon receiving the list of plugins, the bot downloads them, allocates them in the memory and passes control to these libraries.

Different C2 servers could push different plugins. In total we have discovered eight functional libraries:

These plugins form the core toolkit which is used by the perpetrators during attack.
Operation of the malicious platform

As you can see, the cybercriminals use an entire inventory of malicious tools to effectively control the remote computer. Moreover, they have taken measures to conceal their activities: the plugins do not explicitly appear anywhere except in the computer’s memory; they do not get saved to the hard drive; the driver is deleted immediately after launch; all traces in the registry that could indicate this launch get deleted. Only the initial DLL remains on the disk that kick starts the entire process and contains an encrypted version of PlusDLL which is the control DLL.

One of the weak points in this architecture is that the driver does get saved to the hard drive before it launches, so anti-virus products can detect the emergence of this file. The situation is further exacerbated by the fact that the malicious drivers may be signed (although not all drivers in the Winnti samples that we detected were in fact signed). An unsigned driver in itself does not have the means to counter antivirus products and its code can be easily recognized as malicious, whereas signed drivers stand a better chance of remaining undetected by antivirus products: certain anti-virus products consider properly signed programs legitimate by default, so as to minimize the chances of false positive responses.
Kaspersky Lab’s products detect the malicious programs described above under the following verdicts:

The initial DLLs winmm.dll and apphelp.dll, the PlusDll.dll control DLLs, and functional loadable modules (CmdPlus.dll etc.) are detected as Backdoor.Win32.Winnti or Backdoor.Win64.Winnti.

The drivers sp1itter.sys and acplec.sys are detected as Rootkit.Win32.Winnti or Rootkit.Win64.Winnti.

**Communication with the C&C Server**

The data transmitted during the communication between the bot and the C&C server, naturally, do not manifest themselves in any explicit form in online data traffic. Since an active remote control practice can generate substantial traffic, cybercriminals compress communication data with the algorithm LZMA, though they do not include the appropriate header inherent to this algorithm.

The data is transmitted over the TCP protocol. The samples that we analyzed established connections between C&C servers and ports 53, 80 and 443. This port selection is not surprising: they are associated with the protocols DNS, HTTP and HTTPS respectively. All three are routinely used in everyday operations, so they are enabled under most firewall policies. Besides, large amounts of data typically pass through these ports (with the possible exception of port 53), which makes it easier for the malicious traffic to remain inconspicuous.

Although the ports are associated with certain protocols, the actual content of the traffic generated by the malicious program does not correspond to them. Early versions of the Winnti platform exhibited the following traffic structure when communicating with C&C: each block of transmitted data started with the magic number 0xdeadface, followed by the number of blocks (in a DWORD), then the hash of the transmitted block (8 bytes), the size of compressed data (DWORD), the size of source data (DWORD) and, finally, the actual compressed data.

```
<table>
<thead>
<tr>
<th>Magic number</th>
<th>N of blocks</th>
<th>Block checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000:</td>
<td>CE FA AD DE</td>
<td>01 00 00 00</td>
</tr>
<tr>
<td>00000010:</td>
<td>9F 00 00 00</td>
<td>28 04 00 00</td>
</tr>
<tr>
<td>00000020:</td>
<td>52 B1 46 7D</td>
<td>FA C2 7B 48 7E</td>
</tr>
<tr>
<td>00000030:</td>
<td>2D B2 60 D8</td>
<td>AB 49 B1 A0 8F</td>
</tr>
<tr>
<td>00000040:</td>
<td>0E 19 DA 3D</td>
<td>08 3D A4 7F 53</td>
</tr>
<tr>
<td>00000050:</td>
<td>27 42 F3 1A</td>
<td>1A AE 6F B6 C7</td>
</tr>
<tr>
<td>00000060:</td>
<td>9F 00 00 00</td>
<td>28 04 00 00</td>
</tr>
<tr>
<td>00000070:</td>
<td>41 0B D0 86</td>
<td>26 91 FC 35 8A</td>
</tr>
<tr>
<td>00000080:</td>
<td>72 45 82 9E</td>
<td>F0 B4 ED 89 DF</td>
</tr>
<tr>
<td>00000090:</td>
<td>16 0D 5B 19</td>
<td>67 75 45 4D 2F</td>
</tr>
<tr>
<td>000000A0:</td>
<td>E8 BB 9C 60</td>
<td>BB FB D3 50 CA</td>
</tr>
<tr>
<td>000000B0:</td>
<td>59 81 D0 FD</td>
<td>2D 4A 6B</td>
</tr>
</tbody>
</table>
```

*The unit structure of a data block transmitted online in early versions of Winnti*

This is where another weak point of the Winnti family of backdoors becomes apparent. With this data structure, malicious network traffic could easily be spotted by, for example, the magic number 0xdeadface. The cybercriminals probably lost control over victim computers fairly frequently as corporate system administrators identified the intrusion by the unique headers in data packets with the help of IDS/IPS systems, and cleaned their networks. In 2011, new versions of Winnti backdoors appeared that, while still based on the same platform, started to use an updated protocol which included extra encryption to communicate with C&C, so the transmitted
data no longer had static marks in them. Prior to encryption, the data has the following structure (very similar to the earlier format): the first 4 bytes are taken by the magic number 0xaced1984, then a DWORD of data packet description, the next DWORD carries a zero value, 8 bytes of the hash of the transmitted block, then a DWORD with the size of the compressed data, a DWORD with the size of the source data and then the actual compressed data:

![The unit structure of a data block transmitted online in newer versions of Winnti](image)

Then the data is encrypted with regular XOR with a random DWORD size value, and in this form transmitted to the C&C. Knowing that the first four bytes in the source data must represent the value 0xaced1984, it is easy to restore the key for the XOR operation when the data were encrypted. This is how the above data (the XOR value was 0x002a7b2e) looked when it was intercepted in network traffic:

![Encrypted data block transmitted online, in the newer versions of Winnti](image)

Since the encryption key (the value with which the source data are encrypted with the XOR operation) is different each time a fragment of data is transmitted, no more static unique labels can be found in the network traffic which would quickly identify the transmitted data as belonging to the Winnti backdoor. Employing this fast, basic method, the cybercriminals have made it much harder to expose their programs’ traffic.
Whichever protocol is used (with or without extra encryption), the workflow of communication between the bot and the C&C stays the same at the initial stage of operation:

- The bot sends the first data block, thus signaling itself;
- In response, the C&C sends back the list of available plugins;
- The bot starts to download plugins, sending one request at a time to download each plugin;
- The C&C sends the requested plugin;
- The bot sends a message that the plugin has arrived.

We should note here that, to expedite data downloading, the creators of this platform have quite skillfully implemented asynchronous data transmission in their protocol. For instance, the message that the bot has received the first plugin may only arrive at the C&C when nearly all the plugins have been already sent to the bot.

Having downloaded the malicious payload, the bot deploys the plugins in the memory and initializes them. Now it’s all set for complete remote control over the victim computer, and the bot switches to standby mode, waiting for the operator to connect and maintaining communication with the C&C by sending “empty” messages every 15 seconds or so.

Apart from supplying the plugins, no more automatic actions are performed by the C&C: all of the work to examine the infected computers is done manually by the attackers.
Real Case Investigation (Winnti 2.0)

Please note, that the following is published with approval from one of the attacked companies which preferred to remain anonymous. The real company name was replaced with "CompanyXYZ" or simply "XYZ".

On 21st September 2012, a Security Officer of CompanyXYZ contacted Kaspersky Lab and reported a cyber-attack incident. Anomalous activity was spotted at one of the corporate servers. One of the employees noticed a suspicious directory on the server which was created under his account. The folder had a large archived file with information that was regarded as company’s intellectual property.

The anomalies were also confirmed in the network traffic by monitoring software. Several suspicious network connections were established from several computer systems, including network domain controllers, to IP addresses which were not associated with any corporate resources or any other known trusted networks.

The suspicious connections were established on ports 443 and 53. Below is the list of reported IP addresses:

211.60.126.164 (Seoul, South Korea)
113.196.70.169 (Taipei Taiwan)

The security officer at CompanyXYZ did an on-site analysis and managed to locate the process which initiated the suspicious connections using SysInternals Process Explorer tool. The connections were initiated by a system process (svchost.exe). A full process dump using Process Explorer was made and shared with Kaspersky Lab. Our team immediately started searching for malware in the provided process dump.

A next day, one more dump of svchost.exe from another presumably infected machine was provided.

We also received an IP address and port that was spotted in the suspicious connections coming from infected machines: 188.120.246.88:80 (Russia).

First Step Analysis

Quick search through the dumped processes revealed IP addresses mentioned by the company’s security officers.

Suspected malicious IP address in svchost.exe memory of Machine #1.

Suspected malicious IP address in svchost.exe memory of Machine #2.
We checked memory around location of the IP address and found no signs of executable code. The memory was most likely dynamically allocated on process heap and used as a temporary storage of resolved domain name. That is why we had to find another indicator of malicious module related to those IP addresses.

We initiated a port scan of the suspected hosts in parallel to memory analysis. Below is the result on the time of scanning:

```
Nmap scan report for 113.196.70.169
Host is up (0.29s latency).
Not shown: 997 filtered ports
PORT     STATE  SERVICE       VERSION
21/tcp   open   ftp           Xlight ftpd 2.0
80/tcp   closed http
3389/tcp open   ms-wbt-server Microsoft Terminal Service
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

The server was running Windows Terminal Service or was used as a proxy linked to some Terminal Server. Establishing connection via RDP client usually reveals default system locale which is used on welcome screen. Below is what we found upon connection:

![Chinese locale on terminal server welcome screen at 113.196.70.169](image-url)
Checking one of IP addresses on robtex.com brought two possible domain names:

![Rbtex shared host names for IP 113.196.70.169.](image)

One of these domains was found in the memory of dumped svchost process.

![Part of executable configuration seen in svchost memory dump of Machine #1.](image)

Domain name related to the suspected IP address on Machine #2.

Part of executable configuration seen in svchost memory dump of Machine #1.
Googlefiles.net domain was also found in svchost dump of the Machine #1. Besides that, several other domain names were discovered in the same memory block:

- service.interdriver.net
- service.googlefiles.net
- service.dell-support.org
- service.hp-supports.com

Next step was to locate the nearest PE header in the memory of svchost and extract the executable module. After fixing alignment of the sections the file was ready for further static analysis.

Date and time from PE header showed that the executable was prepared about a year before current attack was revealed:

\[
\text{TimeDateStamp: "2011-10-13 07:21:50"}
\]

The executable was a 64-bit application which means that the attackers had already known that CompanyXYZ used 64-bit systems.

The IP address 188.120.246.88, which was seen in suspicious connection was also checked. Connecting to the port 80 of that address with simple TCP client displayed an HTTP GET request:

```
GET /G-Content_XYZ.rar HTTP/1.1
Accept: */*
Cache-Control: no-cache
Connection: Keep-Alive
Host: 127.0.0.1:81
Pragma: no-cache
Range: bytes=2302198299-27335921161
Referer: http://127.0.0.1:81
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)
```

Usually the request is sent by the connecting client, but here the chat between client and server is obviously inversed. That is probably done by the attackers’ tunneling setup which established a TCP connection with some local web server within the company network and an external host that received the stolen data. According to the request, the attackers were downloading a file called G-Content_XYZ.rar, which seems to be an archive of over 25Gb long. The transfer process was instantly interrupted by Security Officers of the company.

**Tactical Pattern Recognition**

The embedded configuration shows some file names. C_20100.NLS was discovered later as the file hosting the same malicious code on the hard drive. WinIo.sys is a driver module on Microsoft Windows Server systems used to process networking requests.

Another interesting piece of data was in a short string “xyz”, which probably refers to the attack campaign name and was defined by the attackers, who deliberately put that name to tag the malware. The word “xyz” most likely stands for the campaign name which comes from the attacked company’s name “CompanyXYZ”.

That was the first evidence that we were dealing with a well-prepared targeted attack against CompanyXYZ. From our previous experience, we have seen several targeted attacks against gaming companies and some of them were also tagged after the name of the companies. In all those attacks there was a recognizable pattern of the attackers: they always used third-level domain names for the command and control server of the malware while second-level
domain name usually resolved to 127.0.0.1 or was a public DDNS domain. A quick check confirmed that this tactical pattern was present in this case as well. Since then, we believed that it is the same attackers we already knew about. This group of attackers was internally labeled “Winnti” by one of our researchers, who named it after one of the very first discovered executable malicious modules.

Active Attack Countermeasures
As soon as we discovered additional configuration, secondary domain names and IP addresses that could be used to control the infected hosts, we instantly reported it to the CompanyXYZ’s Security Officer, who instantly adjusted network firewall rules to block all connections to the attackers’ hosts.

Assisted remote system analysis of another infected machine resulted in discovery of C_20100.NLS file in the Windows system directory and a reference in the system registry to start malicious module as system service:

```
HKLM\System\CurrentControlSet\services\Nwsapagent\Parameters
ServiceDll = C:\Windows\system32\c_20100.NLS
ServiceMain = StartMain
ServiceDllUnloadOnStop = 1
```

Date of registry key creation was the first discovered time of the attack (however, we found an earlier date later):

```
Thu Sep 6 04:26:19 2012
```

Malicious service registry settings were hidden by a rootkit module, however it helped to identify an infection as the registry key name was the same on all the affected computers. Simple creation of a key named HKLM\System\CurrentControlSet\services\Nwsapagent could fail if the system was infected.

```
Rootkit detection method - registry key renaming fails if the key already exists.
```

The rootkit module protected the registry key, but it didn't protect the executable module stored on the hard drive. It was possible to rename c_20100.NLS file, reboot the machine and clean the registry.

Alternative and even more reliable method was to reboot into Windows Safe Mode, clean the registry key and delete the c_20100.NLS file. This method was used by company's System Administrators to find other modules that were not in c_20100.NLS.

The Infection Vector
Since the infection was located and cleaned, the next step was to locate the breach used by the attackers to penetrate the network. Security Officers of the company suggested to start checking from a distinct host they have
suspected. The host (let's call it Machine #3) belonged to an employee without network administrators' rights. It was known that it had connected to the attackers' IPs like the server systems.

The affected company's security officers obtained a copy of the hard drive of the suspected machine and provided a remote access to the disk image. Browsing through the directory structure based on the suspected and adjacent dates of infection (01-06 September 2012) revealed a couple of suspicious files that could have been related to the attack:

```
C:\RECYCLER\en.exe
Type: PE file
Created: 2012-09-06 04:08:53 UTC
Size: 405504
MD5: cf119a66d4c3e2355c1ec4ac316a7130

C:\WINDOWS\system32\drivers\tcprelay.sys
Type: PE file (native)
Created: 2012-09-05 17:27:04 UTC
Size: 99912
MD5: 0b105cd6edc85724c7d52135aa47ef
```

Preliminary analysis of `tcprelay.sys` proved that it was a malicious file which had another encrypted executable file embedded in it. This gave an even earlier suspected timestamp of infection:

**2012-09-05 17:27:04 UTC or 2012-09-05 20:27:04 (local system timezone, UTC+3)**

At the time of check there was no reference in the registry that was linked to `tcprelay.sys`, perhaps due to the fact that system administrators had already cleaned the registry. This was confirmed by a file in local Administrator's Desktop folder:

```
C:\Documents and Settings\Administrator\Desktop\1.reg (created on 2012-09-24 12:44:07 UTC)
```

The file had an exported registry data, which had been removed from the registry during system cleanup on 24th September 2012. Here is the original contents of the registry key *(HKLM\SYSTEM\CurrentControlSet\Services\tcprelay)* before it was removed:

```
\DisplayRegEntry\DisplayName: REG_SZ: tcprelay
\DisplayRegEntry\ErrorControl: REG_DWORD: 0x00000000 (0)
\DisplayRegEntry\ImagePath: REG_EXPAND_SZ: \%\WINDOWS\system32\drivers\tcprelay.sys
\DisplayRegEntry\Start: REG_DWORD: 0x00000001 (1)
\DisplayRegEntry\Type: REG_DWORD: 0x00000001 (1)
```

**Tcprelay.sys registry settings with original file path.**

Once the infection on the machine was confirmed we started looking for the origins of the malicious files. From our previous experience of Winnti gang tactics, we knew that they are keen on sending targeted emails with attached executables. Security Officers helped us check all the emails stored in local Outlook database file on suspected dates of infection, however that didn't reveal anything suspicious.

We have also found system event log files which were copied and analyzed. Event logs had records of tcprelay service start timestamps which confirmed the discovered date of infection. User SID corresponded to the local user account according to the registry.
The Machine #3 had an anti-virus program installed. Checking detection logs of the anti-virus on the suspected date of infection (05.09.2012) showed that there was a single detection right before tcprelay service first start.

We recovered the PDF document called “Transmission with Steps, Realited and Compressed.pdf” from the anti-virus quarantine and prepared to find an exploit inside. The PDF had a lot of obfuscated JavaScript code inside, however we believe that it was not related to the original infection of the system. It was clean and the anti-virus detected it by mistake, probably because of some suspicious obfuscated JavaScript code.
PDF document detected by the antivirus as malicious.

The JavaScript code inside the PDF was used to process an interactive form inside the PDF and support dynamic interactive 3D model embedded in the document using Adobe 3D technology.

After that, we checked the infected machine’s browser history. The Internet Explorer history log files showed that the user was reading email right before the infection of his machine.

Internet Explorer log history record: html file from Outlook.
With that in mind, we analyzed the Outlook local database again. This time we used several techniques to recover emails that were deleted from the Trash folder. This helped to partly recover a message which arrived on the day of infection.

Recovered targeted attack email on Machine#3.

The text of the message supposed to contain an attachment, however the attachment and MIME headers of the email were completely lost and couldn't be recovered. However, it was clear that the email was a targeted attack against the employee of the company. It was sent from companyxxyz@163.com and replaced “From” field in the email body which made it look like a legitimate email in the list of messages in Outlook.

Targetted attack email in the list of Outlook messages.
We discovered a Windows prefetch file in the system directory, that was created when the malicious attachment was opened. The timestamp correlates with the time of infection.

\[C:\\WINDOWS\\Prefetch\\CompanyXYZ \text{ EMPLOYEE SALARY ADJ-1AF9D56A}.pf\]

**Time of creation: 2012-09-05 19:52:00 (local timezone, UTC+03)**

Unfortunately, the prefetch file format is proprietary and there is nothing interesting in those files, except the original executable file name. Full path of the malicious executable that infected the first computer in the company was:

\[C:\\Documents and Settings\\<Username>\\LocalSettings\\Temp\\RAR$EX00.156\\\text{CompanyXYZ EMPLOYEE SALARY ADJUSTMENTS EBOOK.EXE}\]

According to the file path, this executable was a part of an archive, which was opened with WinRAR installed on the system.

Upon discovery, we requested the Security Officers to provide us with full MIME as well as to check who else may have received the same message. The check discovered series of emails sent to several publicly known email addresses. In all cases the text message was the same as shown above, however sent from different mailboxes. The Return-Path MIME filed seemed to have the original email addresses of the attackers:

- companyxxyz@163.com
- company.xyz@gmx.com

The attackers used the same IP to send out emails: **118.142.11.14**

inetnum: 118.140.0.0 - 118.143.255.255
netname: HGC
descr: Hutchison Global Communications
country: HK
person: ITMM HGC
nic-hdl: IH17-AP
e-mail: hgcnetwork@hgc.com.hk
address: 9/F Low Block,
address: Hutchison Telecom Tower,
address: 99 Cheung Fai Rd, Tsing Yi,
address: HONG KONG
phone: +852-21229555
fax-no: +852-21239523

The emails we checked had the same attachment of 96782 bytes named “Salary adjustments.zip”. There was only one file inside ZIP archive, called “CompanyXYZ Employee Salary Adjustments Ebook.exe”. Full details about this application are provided further down in current report.

To summarize, the targeted attack started from an email sent at 05.09.2012 19:12 (UTC+03). It resulted in system infection at 05.09.2012 19:52 (UTC+03).
Full File Analysis

Salary adjustments.zip File
Size: 96782
MD5: 1b56416fefa2d2c863f3b46dfb6dc353
Location: targeted attack email message attachment
Creation time (author’s timezone): 2012-09-05 14:29:10
This file is just a container for “CompanyXYZ Employee Salary Adjustments Ebook.exe”.

CompanyXYZ Employee Salary Adjustments Ebook.exe File
Size: 122880
MD5: 6ef66c2336b2b5aaa697c2d0ab2b66e2
Location: “Salary adjustments.zip”
Creation time: unavailable
Link time (UTC): 2012-07-21 18:50:18
Internal name: FlashUpdate.EXE
This application is a wrapper for another embedded executable modules. It serves as a dropper of malware.

![Malware dropper file structure](image)

Notable fact: this application has a resource section inside and the default locale is set to Chinese Simplified.

The file creates three long binary data registry keys, two of which are encrypted executable modules and one encrypted config from the body of the original dropper. These values are encrypted with simple 1-byte XOR.
Sysinfo config module is used by sysbin01. Apparently it starts with the company name and has three domain names, one of which is most likely used to check Internet connectivity (update.microsoft.com).

Sysbin01 module is a loader component. It creates several threads running various jobs.

**Sysbin01.thread#1** attempts to load `%TEMP%\<ComputerName>.ax` file and decrypts it.

We checked the system but couldn’t find `<ComputerName>.ax` file in the Temp folder of the user, however we found other .ax-files that seemed to be related because of the date of file creation.

- **File name:** C:\Documents and Settings\%User%\Local Settings\Temp\%ComputerName%_p.ax
- **File size:** 2660
- **Creation time (UTC):** 2012-09-06 06:22:42
- **MD5:** unavailable (the system went offline before we discovered the filepath).

- **File name:** C:\Documents and Settings\%User%\Local Settings\Temp\uid.ax
- **File size:** 16
- **Creation time (UTC):** 2012-09-06 05:03:06
- **MD5:** unavailable (the system went offline before we discovered the filepath).

According to the code that loads `<ComputerName>.ax` it is an encrypted executable file, which is decrypted and loaded to memory by own loader routine in the sysbin01 module.

**Sysbin01.thread#2** spawns a new instance of **Sysbin01.thread#3** every 10 seconds during, that is done 3 times.
Sysbin01.thread#3

This thread is the most important. It reads the configuration from the registry and connects to the C&C servers specified in the config via direct tcp connection or via proxy that is fetched from the settings of locally logged in user profile. The config had the following C&C: tank.hja63.com. It sends a “POST /<HEXNUMBER>” request with User-Agent “lynx”, the data after HTTP header is just “AA”, expected answer is also “AA”.

This thread also creates %TEMP%\uid.ax and stores current system unique ID, which is generated by CoCreateGuid system API (16 bytes). It is able to receive and save data from the C&C server to a file. It also monitors windows of explorer.exe and copies textual data from password fields if the user types in, stolen data is saved to a file first.

After all threads are launched, the main thread waits for termination of Sysbin01.thread#3, which is created first and then exits.

Sysbin02 module behaviors depends on currently running processes. There is an embedded DLL file according to Figure 15 in sysbin02.

If the system has a running process named “360tray.exe”, then the embedded file is stored in %SYSTEM%\MFC42LOC.DLL, then copies the source executable (FlashUpdate.exe) to %TEMP%\Flash.tmp and runs a new process from that location via WMI Win32_Process.Create method.

If the system has a running process named “bdagent.exe”, then it copies the source executable (FlashUpdate.exe) to %TEMP%\Flash.tmp, decodes an embedded Base64 string and executes. The string has the following text after decoding:

\reg add \"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\" /v FlashUpdate /t REG_EXPAND_SZ /d "\\%APPDATA%\\FlashUpdate.exe\\-update activex" /f

The module also saves current module file path to the registry in the following key location:
\HKCU\Software\Classes\path

Next it patches the tmp file with two dword “AAAA” values which looks like corruption of embedded encrypted sysbin modules inside. The meaning of this action is currently unclear.

Then it moves Flash.tmp file to FlashUpdate.exe by and starts a new process from new location.

Finally, if there is not “qqpctray.exe” process running, and this seemed to be the case for the analyzed system, it copies the source executable (FlashUpdate.exe) to %TEMP%\Flash.tmp, patches the new file and increases its size by adding system explorer.exe file contents to the resource section “RC Data” 20 times. The purpose of this is to make the new executable look like the real update service of Adobe Flash, it simply stuffs the file with executable code of another application. Then it moves the file to new location %APPDATA%\FlashUpdate.exe, saves new module file path to the registry in the following key location: HKCU\Software\Classes\path and starts a new process from there.

c_201000.NLS (aka SrvCore.dll) File

Size: 15847156
MD5: 5778178a1b259c3127b678a49cd23e53
Location: C:\WINDOWS\system32\c_201000.NLS
Creation time (UTC): unavailable
Link time (UTC): 2011-09-16 13:23:34

PAGE 38 | “Winnti” | More than just a game
**Summary**

c_20100.NLS works in two modes. The first mode is a load as a dynamic library and the second is a launch as a service. Both branches have the same core functionality.

This module is a universal executable code loader with no embedded payload. Its main purpose is to connect to the C&C server, download and store the encrypted payload in the system registry. It is also responsible for loading, decrypting and running the payload module from the registry after system restart.

**Details**

c_20100.NLS contains a ciphered block with initial settings. This ciphered block resides at the very end of the file of this malicious program and is decrypted in the beginning of execution. Structure of block:

<table>
<thead>
<tr>
<th>Magic number</th>
<th>Ciphered and packed initial settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x19860609</td>
<td></td>
</tr>
</tbody>
</table>

The malicious program XORs the magic number with a hardcoded value 0x19860609, converts a resulted value into a HEX-string and uses that string as a key for RC4 cipher algorithm. In this case string-key represents “00000000” because of the magic number is equal to the hardcoded XORing value. With that key malicious program decrypts (RC4) ciphered archive. The archive has the following data:
The malicious program tries to read registry value “SrvCode” by registry path:
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion. That value is expected to contain ciphered with RC4 data.

To decrypt it program uses 2\textsuperscript{nd} integer of initial settings (in this case 0x3514) XORed by hardcoded byte 0x12. Result is converted into a HEX-string and is used as RC4 key for further decryption (here it is “00003506”). That registry value appears if this malicious program had already worked on the system and received data from the C&C server in the past. Content of “SrvCode” poses a ciphered executable which should be loaded into the memory and run.

If “SrvCode” is not found malware makes attempts to connect to one of the specified C&C servers.
C&C Server Address Selection

Initial settings define the type of C&C format. Byte at offset 0x24C stores the C&C type value:

**0x00**: the malware uses 4 URL-based C&Cs placed at 0x4C, 0x8C, 0xCC and 0x10C offsets. By all appearances these are public resources (forums, blog platforms and so on) where the attackers leave messages with specially crafted content for a bot. If connection fails, the malware tries another approach.

**0x01**: the malware uses attackers’ hardcoded servers and connects to host and port specified at offsets: 0x14C, 0x18C, 0x1CC and 0x20C. If connection fails the malware tries another approach.

If URL-based scheme is used then malware loads a web-page by specified in settings URL. The target text has to begin and end with special hardcoded delimiters: “B9273C17” – start, “B6A74634” – end. The malware reads contents of the webpages until it finds a proper page with delimiters. If found, the malware takes the text between delimiters and treats it as data of hex string, converts it to the binary data and decrypts resulted data using RC4 algorithm with hardcoded key “rtyr_45_trf”. For example:

```
"B9273C17E67024277AE02E2A8A780B243C0BCA88FE85A1B6A7463",
```

The data between delimiters:

```
"E67024277AE02E2A8A780B243C0BCA88FE85A1",
```

It is converted into binary: 0xe6 0x70 0x24 ... 0xa1 and this buffer is decrypted with RC4 key “rtyr_45_trf”. Result is a host and port of C&C: “nx2.intercpu.com:25”.

If the host-port schema is used then malware simply connects to the hardcoded C&C servers directly.

Communication with C&C Server

Once a working C&C server is found the malware sends specially crafted ciphered buffer to via TCP/IP. On request from a bot a C&C server replies with several blocks of data described below:

**1st block**

0xC bytes of header: 0x1000010, 0x1000010, <reserved 4 bytes>.

**2nd block**

0x1C bytes (due to absence of real respond from the C&C I’m presenting an example buffer of this block containing bytes 0x00, 0x11, 0x22, ..., 0xFF, 0x00, 0x0, 0x0, ..., 0x00):

```
<table>
<thead>
<tr>
<th>Magic number</th>
<th>Checksum</th>
<th>Size of original</th>
<th>Size of archive</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000000000</td>
<td>00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>000000010</td>
<td>00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**2nd block of 0x1c bytes: example**

First DWORD of this buffer (here, 0x33221100) is a magic number which is XORed with the value 0x1986052. Resulted lowest byte is used to XOR unpacked data.
Word at offset 0x4 (here, 0x5544) poses a checksum of unpacked data which should correspond with actual received content.

DWORD starting at offset 0x8 (here, 0xBBAA9988) represents a size of unpacked data.

Value at marked place at 0xC offset (in example picture it is dword 0xFFEEDDCC) represents a size of next block of data to be received. That data will pose an archive, hence this value represents a size of packed data.

3\textsuperscript{rd} block

The 3\textsuperscript{rd} block poses an archive of ciphered data. Being received, unpacked and decrypted, data is ciphered again with RC4 and stored into “SvcCode” value of registry by mentioned above registry path.

The eceived data is processed as an executable file to run. The malware places the executable in memory, prepares for running and makes call to the entry point of the new code. Then it waits when following event will be triggered:

\texttt{Global\%D5ACF9F6-C8B3-47d1-9768-57162E1F5FDB}

When triggered, the malware finishes execution. During the process of finishing it deletes registry value “SvcCode” along with values “DrvCode” and “KeyCode” from the same registry path although this malware was not creating them.

\textbf{Tcprelay.sys File}

Size: 99912
MD5: 0b105cd6ecdfe5724c7db52135aa47ef
Location: C:\WINDOWS\system32\drivers\tcprelay.sys
Creation time (UTC): 2012-09-05 17:27:04
Link time (UTC): 2011-12-21 13:55:03

This file is a Microsoft Windows native application, which is loaded as a driver and had a valid digital signature in 2012.

The certificate was issued by LivePlex Corp, which can be found online by searching for the company name. One of their webpages is here: \url{http://www.linkedin.com/company/liveplex}
About LivePlex

LivePlex has prepared its online game business since 2007 by operating its subsidiary and development studios. LivePlex took its first step in to the game industry by publishing 'TZ Online' followed by other online games such as 'Genkis Khan' and 'The Invincible' in 2009.

In 2011, the company launched 'Dragona Online', its first in-house development in Korea and now being operated throughout various countries with remarkable performances. LivePlex released its second in-house development named 'Queens Blade' demonstrating its advanced development capabilities.

In addition to current line-ups, LivePlex is gearing up to launch its new title 'Aurora World' in the 2nd half of 2012. With the achievement of successful service in Korea and active expansion into various markets, LivePlex is now positioning itself to become a renowned global game company.

Game portals:
- http://kr.gameclub.com (Korea)
- http://ph.gameclub.com (Philippines)

Specialties
PC Online Games Publishing/Development, Mobile Games Publishing/Development

Digital certificate of Tcprelay.sys

LivePlex profile page on LinkedIn
When the driver is loaded it decrypts an embedded DLL file, which is immediately injected into the address space of services.exe process. Then the driver sets up some rootkit functionality to hide TCP connections by patching the system tcp/ip driver.

The injected DLL was called s.dll at the time of compilation and is yet another module for analysis.

**S.dll File**

Size: 77825
MD5: 1716889fcee461e7cde5128c14d206cb
Location: inside tcprelay.sys
Creation time (UTC): 2012-09-05 17:27:04
Link time (UTC): 2011-03-01 09:07:12

This opens system event named “401d-b49a-93cf7a18e5b3” and sets event to fired state if it exists. The code checks for proxy server configuration by impersonating a logged in user and fetching settings from the registry. It can work both with Socks and HTTP proxies. The module attempts to connect to the list of 8 domains, consisting of the following command and control servers (some of them are used more than once):

- a1.googletrait.com
- a1.nexongame.net
- a1.reegame.net
- mail.nexongame.net

It automatically looks for open C&C ports in the following order 53,443,8080,25,80,3690,1433,80.

During connection over HTTP proxy it uses the following User-Agent string: “MyApp/0”.

The application is linked with libmysql.dll and Zlib (v.1.2.3). Current Zlib version is 1.2.7 and was released on 2nd May 2012, while version 1.2.3 seems to be released in July 2005. Zlib version 1.2.4 was released on March 2010, so the original module was probably designed somewhere after July 2005 and before March 2010.

Then it collects system information, which includes the following:

- Host name
- OS Service Pack version
- System default language ID and Code page
- List of local drives with free space
- Internal hardcoded identifier (“12-21”)
- Process commandline
- Logged in user name
- System directory path
- Amount of free system memory
- CPU name
- Terminal services port number
The information is stored in a buffer that begins with hardcoded header magic number: 0xDF1F1ED3. The block is compressed using Zlib (v.1.2.3) compress2 method with compression level 8. The data is compressed later and prepended by a 4-bytes header as shown below.

<table>
<thead>
<tr>
<th>SIZE of DATA</th>
<th>00 BYTE</th>
<th>RND BYTE</th>
<th>DATA (zlib compressed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td></td>
<td></td>
<td>Variable length (less than 65535)</td>
</tr>
</tbody>
</table>

**Format of a message sent to C&C**

After submitting system information the module expects 4 byte response code from the server after which it sends one 00 byte to complete the handshake procedure.

Then the module expects an interactive communication session with the remote operator. It provides capability to run various commands including (command names were defined during reverse engineering):

- process_list
- kill_process
- dir_list
- smbshare_list
- smbshare_mount
- dir_make
- file_delete
- file_move
- file_upload
- file_open
- file_write
- file_close
- file_find
- url_download_to_file
- process_start
- process_start_and_get_output
- dll_load
- dll_call_export
- screen_getsnapshot
- screen_set_cursor_position
- screen_send_input
- tcpproxy_open_connection
- tcpproxy_close_connection
- mysql_connect
- mysql_fetch
- mysql_disconnect
- driver_tcpreplay_interact
- tcpsession_close
- quit

A command output is compressed using Zlib and sent to the server in asynchronous mode. To summarize, it is obvious that this executable module is a backdoor, capable of taking screenshots, stealing files, downloading new
files from the Internet, starting and killing processes, including interactive Windows shell commands, file search and interaction with mysql database server.
An icon embedded in en.exe is a default application icon from MS Visual Studio

This application is a dropper, it fetches a resource called EXEFILE from current application and saves it into following paths:

<CURRENT DIR>\dllcache\sethc.exe
C:\WINDOWS\system32\sethc.exe

Then the module uses undocumented Windows API from SFC_OS.dll, a function called SfcFileException to update the system version of C:\WINDOWS\system32\sethc.exe.

The file C:\WINDOWS\system32\sethc.exe (SET High Contrast) is to enable the High Contrast accessibility feature in order to allow people with visual impairments to log in. SETHC is activated at logon screen with LeftAlt+LeftShift+PrintScreen key combination.

By replacing C:\Windows\SYSTEM32\SETHC.EXE with a custom application an attacker can run an arbitrary application with SYSTEM privileges running in zero session (in separate desktop space from normal applications). After the new file replaced the system sethc.exe application, current module adjusts the privileges of sethc.exe to disable access to the file from any other application. This is achieved by calling external system tools cacls.

Replace access rights to the files, allow everyone full access:

calcs C:\WINDOWS\system32\sethc.exe /c /e /p everyone:f
calcs <CURRENT DIR>\dllcache\sethc.exe /c /e /p everyone:f
Revoke access to the file for everyone, leave only system readonly access:

```plaintext
calcs C:\WINDOWS\system32\sethc.exe /t /c /e /r everyone
calcs C:\WINDOWS\system32\sethc.exe /t /c /e /r administrators
calcs C:\WINDOWS\system32\sethc.exe /t /c /e /r users
calcs C:\WINDOWS\system32\sethc.exe /t /c /e /r system
calcs C:\WINDOWS\system32\sethc.exe /t /c /e /r “Power Users”
calcs C:\WINDOWS\system32\sethc.exe /c /e /p system:r
```

The dropper also changes the file timestamp. It is set identical to C:\WINDOWS\system32\ntvdm.exe.

The dropper application has a resource section with Menu, Dialog templates and other information put by the MS Visual Studio Application Wizard. It includes default system locale from the developer’s system, which is Chinese Simplified.

```
<table>
<thead>
<tr>
<th>EXEFILE</th>
<th>Icon Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu</td>
<td>Dialog</td>
</tr>
<tr>
<td>Sting</td>
<td>Group Icon</td>
</tr>
<tr>
<td>Manifest</td>
<td></td>
</tr>
</tbody>
</table>

Chinese locale in resource section of En.exe

**Sethc.exe File**

Size: 20480
MD5: 3ba06424e8244f17a8d269c4d404c39c9
Location: resource section of En.exe
Link time (UTC): 2009-05-16 07:09:35

This small file has very basic functionality. It is written using MS Visual C++ with MFC and is used to render a simple dialog window. Like En.exe it has resource section, describing the dialog window and default locale is set to Chinese Simplified.

Once it replaced local system sethc.exe tool it can be invoked when the desktop is locked with LeftCtrl+LeftShift+PrintScr key combination. This brings a dialog Window similar to system StickyKeys application. However, if you press Ctrl+Alt+F you will immediately see a hidden input box. If you enter “ydteam” in the input box and press Ctrl+Alt+K, the application will welcome you with a message box and will execute a TaskManager.
As far as sethc.exe is executed with privileges of local system, the task manager also inherits these privileges and is capable of killing any other process as well as starting any other application with system rights. Apparently, this is a backdoor to the system. An attacker can run cmd.exe, add local users with administrative privileges and log in. We checked if the tool was publicly shared on the Internet, but couldn't find a page distributing it freely. That is why we assume that it is developed and used privately.

**Full list of C&Cs**

Below is full list of all collected domains and IP-addresses of C&C servers have they been mentioned in initial settings of `c_20100.nls` or hidden in text messages at public places in Internet:

**C&Cs from public resources:**

27.115.103.198:8885
27.115.103.195:8885
114.222.36.32:10000
27.115.103.195:23456
27.115.103.195:10000
nx2.joymax.in:80
nx3.joymax.in:80
nx2.intercpu.com:25 (174.36.138.30)
nx3.intercpu.com:25 (174.36.138.30)
nx3.interdriver.net:53 (119.240.212.110)
stan227.guicp.net:8008
Hardcoded C&C from the malware:

service.interdriver.net:443 (98.126.218.64, 199.188.106.231)
service.googlefiles.net:53 (98.126.218.64, 199.188.106.231)
service.dell-support.org:25
service.hp-supports.com:80
tank.hja63.com
a1.googletrait.com
a1.nexongame.net
a1.reegame.net
mail.nexongame.net

Interestingly, there is an overlap of C&Cs from public resources and hardcoded domains:

nx3.interdriver.net:53 <====> service.interdriver.net:443

The nx3.interdriver.net was published by awertasegfae@yahoo.com and was discovered at

http://awertasegfae.blogspot.ru/2011/10/first-test.html. This means that at least the individual who owns
awertasegfae@yahoo.com for sure belongs to the same gang who attacked CompanyXYZ.
Source of Attacks

So, who is behind Winnti? While analyzing the malicious files that we detected during our investigations we found some details which may cast some light on the source of the attacks.

As part of our investigation, we monitored exactly what the cybercriminals did on an infected PC. In particular, they downloaded an auxiliary program ff.exe to the Config.Msi folder on the infected machine. This code searches for HTML, MS Excel, MS Word, Adobe, PowerPoint and MS Works documents and text files (.txt) on the hard drive.

Debugging lines were found in ff.exe that possibly point to the nationality of the cybercriminals. They were not immediately noticeable because they looked like this in the editor:

```
\%s\%s\ $Bitmap \$ I 3 0 mnt
\%w\%d .pptx .pdf .docx .chm
```

However, during a detailed analysis it emerged that the text is in Chinese Simplified GBK coding. This is what these lines look in Chinese:

![Chinese characters]

- 未识别的文件系统类型
- 打开卷失败
- 获取文件系统类型失败
- 读卷失败
- 卷没有打开或打开失败
- 定位到根目录错误
- 错误的内存读指针
- 内存太小
- 文件不存在
- 获取文件mft索引扇区失败
- 获取文件数据运行失败
- 卷与打开卷不相同
- 卷与打开卷相同
Below is a machine translation of this text into English:

Not identify the type of file system
Below is a translation of the text by interpreter
Open the volume failed
Failed to get the file system type
Failed to read volume
Volumes do not open or open failed
Navigate to the root directory of the error
Error memory read pointer
Memory is too small
File does not exist
Failed to get the file mft index sector
Access to file data fail
Volume and open volumes are not the same
The same volume and open volume

In addition, cybercriminals used the AheadLib program to create malicious libraries (for details, see the second part of the article). This is a program with a Chinese interface.

Chinese text was also found in one of the components of the malicious program CmdPlus.dll plug-in:

```plaintext
explorer.exe...\cmd.exe...\cmd.exe 进程已经退出!! .exit
....???.??????????????????????????????LISTENING.....
SYN_SENT????????????SEN_RECEIVED.....ESTABLISHED....
...FIN_WAIT????????????FINT_WAIT2????????CLOSE_WAIT.....
........CLOSING?????????LAST_ACK????TIME_WAIT.....
```

Translation: The process is complete!!

It would appear that the attackers can at least speak Chinese. However, not everything is so clear cut: because the file transfer plug-in has not been implemented entirely safely, a command which includes the attackers’ local path (where the file comes from and where it is saved to) arrives during the process of downloading/uploading files on the infected system. While monitoring the cybercriminals’ activity on the infected machine, we noticed they uploaded the certificate they found in the infected system, and the network traffic reflected the local path indicating the place where they saved the file on their computer:

```
C:\Documents and Settings\Administrator\바탕 화면\funshion.txt
```

These characters appear to be Korean, meaning “desktop”. This means the attackers were working on a Korean Windows operating system. Therefore, we can presume that the attack is not exclusively the work of Chinese-speaking cybercriminals.
The Search for Attackers  (XYZ incident)

Locating the attacker is one of the most non-trivial parts of the research. The attackers normally do not leave any traces in the malware that can be directly bound to their real identities. That is why we have to use all available bits of information that seems to find other unique related content on the Internet or any other available data sources. One of the important stages is to extract unique identifiers/nicknames/tags that can be discovered on the Internet and after that find individuals who are related to creation or distribution of this content.

YDTeam Hacking Group

The string “ydteam” looked non-random and we decided to check it on the Internet. It turned out that YDteam is a hackers group name and has a lot of references on Chinese segment of the Internet:

http://zhikou.yo2.cn/ - probably a team member web blog

http://www.exploit-db.com/exploits/11053/ - PoC exploit for Chinese media player by the team member called “t-bag”

Another team member called “b4che10r” according to

http://zzsky.5d6d.net/archiver/tid-127.html

http://hi.baidu.com/0x255/item/22cbbfe97ca9963c87d9de41

http://www.indetectables.net/viewtopic.php?f=87&t=22185&view=print

b4che10r's personal blog: http://blog.taskkill.net/

Another team member called “Shalyse” according to

http://forum.cnsec.org/thread-50222-1-1.html

Another team member called “killer” according to

http://zzsky.5d6d.net/archiver/tid-127.html

There was a website ydteam.cn that seems to be related to the activity of the group. According to the domaintools.com database, it was registered on 2009-10-06 15:12 and put on hold around 2010-10-08. The original WHOIS information from domaintools.com:

Domain Name: ydteam.cn
ROID: 20091006s10001s23027085-cn
Domain Status: ok
Registrant Organization: 魏楠
Registrant Name: 魏楠
Administrative Email: wn6805@126.com
Sponsoring Registrar: 北京新网数码信息技术有限公司
Name Server:ns.xinnetdns.com
Name Server:ns.xinnet.cn
Registration Date: 2009-10-06 15:12
Expiration Date: 2010-10-06 15:12

Registrant name 魏楠 (Wei Nan) seems to be represented in the mailbox wn6805@126.com, which could mean the owner of the website used real identity. The domain was most likely registered by the team leader.

The email itself was used on several other websites. For example

http://tieba.baidu.com/f?ct=335544320&lm=0&rn=30&tn=baiduPostBrowser&sc=0&z=633089789&pn=0&word=%BC%AF%C4%FE%D2%BB%D6%D0

The webpage above has a post offering to “help with cheap shopping online”. That is most likely related to a fraudulent activity of the email owner (stolen Internet-banking credentials or credit card information). The same page reveals a QQ id of that individual and a username:

QQ: 97676416
Username: 大头禹

Another page http://www.gtvod.com/gtvod/jsp/public/personal/index.jsp?id=20100127213936126005 shows information about the user registered with name “wn3118” and the same email:

E-mail: wn6805@126.com
Date of Birth: 1992-12-21
Marital Status: Unmarried

Another page http://tieba.baidu.com/p/652667782 has a message from profile “灬low-key, wn” (which links to wn6805@126.com). Profile information reveals gender of the individual:

http://www.baidu.com/p/%E7%81%AC%E4%BD%8E%E8%B0%83%E4%B8%B6wn/detail

Gender: Male

There are few essays in Chinese probably written by the individual owning wn6805@126.com while studying at Junior High School:


A page from zww.cn also shows some details about the author:
http://www.zww.cn/zw/myzw.asp?url=%CA%A7%C8%A5%B0%AE

Birthday: 1992-12-21 (confirms previous finding)
QQ: 251985076
Joined: 2008-09-16 22:35:00
Last login: 2009-06-09 10:37:00
Searching for the QQ id 251985076 brings to [http://blog.sina.com.cn/dahuadl](http://blog.sina.com.cn/dahuadl) that has
User mobile number: 13847416805

The hackers team also seemed to own [ydteam.com](http://ydteam.com) for some time according to reference at [http://zzsky.5d6d.net/archiver/tid-127.html](http://zzsky.5d6d.net/archiver/tid-127.html)

Domaintools.com shows that the domain was registered to a Chinese individual from 2009-06-03 to 2011-08-22. After that WHOIS information was protected by a Privacy protection service. Here is WHOIS data at the time of domain registration:

Admin Name........... zheng wenlong
Admin Address........ tianjin jiefangdongjie 63hao
Admin Address........ yancheng
Admin Address........ 300560
Admin Address........ fujian
Admin Address........ CHINA
Admin Email.......... vydteam@yahoo.cn
Admin Phone.......... +86.13652452428

Please note, that +8613652452428 is a Chinese local cell phone number.

Domaintools.com has also preserved a screenshot of the website while it was online on 2010-02-25. It shows some of the team member names mentioned above.
Ydteam website as it was in 2010

Another trace to the source of attack is based on email sender IP address. The emails were sent from 118.142.11.114. According to robtex.com, there are 2 domain names that share this IP:

- pad62.com
- ru.pad62.com

Pad62.com was created in 2011-06-05, on the date of registration if had non-protected WHOIS information, according to domaintools.com:

- Registrant: ji shao
- Xuan Die Xiao Jie 418 Kao
- peng hu, xiang gang 064562
- China
Registered through: GoDaddy.com, Inc.
Domain Name: PAD62.COM
Created on: 05-Jun-11
Expires on: 05-Jun-12
Last Updated on: 05-Jun-11

Administrative Contact:
shao, ji  huisengaunr@sina.com
Xuan Die Xiao Jie 418 Kao
peng hu, xiang gang 064562
China
1-330-040-0367

We checked which other domains are associated with the WHOIS information above and found the following domain names:

- 100-d.com
- sm08.com
- cx-cx.com
- 6-pro.com
- aohoe.info
- besheo.info
- dyyerre.info
- jiaoyouliaotian.org
- tao5178.info

One more route is to check the C&C of the initial dropper/downloader module. This was **tank.hja63.com**.

According to domain tools, hja63.com had non-protected WHOIS information in 2011:

Registrait: ji shao
Xuan Die Xiao Jie 418 Kao
peng hu, xiang gang 064562
China

Registered through: GoDaddy.com, Inc.
Domain Name: HJA63.COM
Created on: 05-Jun-11
Expires on: 05-Jun-12
Last Updated on: 05-Jun-11

Administrative Contact:
shao, ji  huisengaunr@sina.com
Xuan Die Xiao Jie 418 Kao
peng hu, xiang gang 064562
China
1-330-040-0367

When we checked, **tank.hja63.com** resolved to **173.234.184.45** (owned by DiaHosting Limited, USA), while hja63.com resolved to 68.178.232.100 (GoDaddy ISP server).
Bot Control Messages On Public Resources

Analysis of the file c_20100.nls revealed additional information leading to probable attackers. Looking for identifiers (used as message boundaries, or delimiters) **B9273C17** and **B6A74634** specified in this malicious file on Internet we found the following pages where the attackers left messages for the bots:


An encoded C&C address for a bot on a public webpage

Another place of just mentioned forum thread:

- https://groups.google.com/group/openmeetings-dev/browse_thread/thread/ccfeb8242a4f11ec/a700f22be192482a?show_docid=a700f22be192482a&pli=1
Some more server addresses for the bot

Here, we see these emails used as commenters’ identifiers:

Jimycoco...@gmail.com
awertase...@yahoo.com

Jimycoco...@gmail.com most probably refers to Jimycocowell which is a username that pops up further.

Searching for “awertase” brought another forum thread where ciphered data for the same bot appeared:


Yet another message for bots from awertase...
The full email behind awertase...@xxxxxxxx seems to be awertasegfae@yahoo.com according to http://awertasegfae.blogspot.ru/2011/10/first-test.html

http://hi.baidu.com/alonecode/item/6936f85a3d98ce3533e0a9ed

Another webpage with message for bots

According to Figure 32, “mer4en7y” and “alonecode” (from the URL of the page) are nicknames which are related to the user of the Baidu blog platform where messages for a bot were left. Google Search for the nickname “mer4en7y” returned 5490 results. This is a very active user that posts messages for this type of bot. The first results lead to hacker forums and IT-security specific web-platforms. The same nickname has appeared on a well-known Romanian Security Team forum.

Mer4en7y Individual Activity

mer4en7y username at Romanian hackers forum
According to the following, Mer4en7y submitted a vulnerability found in Weihai City Commercial Bank system:

http://wooyun.org/bugs/wooyun-2010-011002

The vulnerabilities Summary

- Defect Number: WooYun-2012-11002
- Vulnerability title: Weihai City Commercial Bank struts command execution vulnerability
- Manufacturer: Weihai City Commercial Bank
- The loopholes Author: mer4en7y
- Submitted: 2012-08-17
- Open time: 2012-10-01
- Vulnerability Type: command execution
- The hazard rating:
- Self-evaluation Rank: 5
- Vulnerability Status: was handed over to third-party vendors the (ncert National Emergency Response Center) processing
- The loopholes Source: http://www.wooyun.org
- The Tags label: Nil
- Share vulnerabilities: 0 people Favorite
Vulnerability details

Disclosure of state:

- **2012-08-17**: The details have been notified to manufacturers and wait for the vendor processing.
- **2012-08-20**: The vendor has confirmed that the details are only open to vendors.
- **2012-09-30**: Details exposed to the core white hat and experts in related fields.
- **2012-09-09**: Details open to the general white hat.
- **2012-09-19**: Details open to internship white hat.
- **2012-10-04**: Details to the public.

Brief Description:

- Struts loopholes not complement.

Mer4en7y's activity on vulnerability research

Favorite videos and tutorials of Mer4en7y:

http://www.tdcqjslt.com/u.php?uid=1918

Mer4en7y’s favorites confirm malware-related activities

Mer4en7y’s micro-blogging page at t.qq.com: http://t.qq.com/mer4en7y

Alias of that profile is translated as “watching a rain”.
A user with nickname “d4nr4n” (http://t.qq.com/d4nr4n) is posting a message where mer4en7y is mentioned:

Google translation: “%mentioned individuals% go to Nanjing tomorrow xx training institutions to maintain four months … C++ learning, seeking Nanjing-based friends of the exchange”

Mer4en7y at yoyo2008.com:

Mer4en7y profile at yoyo2008.com

One of two friends of Mer4en7y in yoyo2008 social network is a user named “mayuan” which seems to be from Xinjiang and a graduate of Judicial Police School according to shared private information out there:

Mer4en7y's contact profile at yoyo2008.com

http://u.pintour.com/uid-b1bf56e230cc42d9bfa003a7718888d2/
Another Mer4en7y profile show **Nanjing** as a hometown

Mer4en7y’s exploit has been involved in the penetration of public radio service ftp server (according to WHOIS information this domain belongs to Xi’an Municipal Bureau of Radio and Television).

A trace of cyberattack based on Mer4en7y’s code

As we can see here Mer4en7y had an email address associated with 90sec hackers team.

Another reference on the net shows that Mer4en7y is after sourcecode of proprietary products (probably udf.dll from Roxio Inc):

http://www.uedbox.com/udf-dll-source/

Mer4en7y discussing udf.dll source-code and cmdshell
The following confirms that **Mer4en7y is a member of 90sec group**. The group website is located at [http://www.90sec.org/](http://www.90sec.org/):

**90Sec team about-page**

Mer4en7y replies on job offer posted at 90sec forum (someone wanted to hire computer exerts with **very special knowledge**):


Rough translation of job offer from Chinese:

“Subject: Looking for information security researcher

*From: Southland sword*

*Time: 2012-04-06 00:38*

**Subject: Security researcher job**

**Responsibilities:**

1. Full target penetration alone or with a team depending on available resources;
2. Penetration testing report and recommendations

**Technical requirements:**

1. Knowledge of penetration testing, methods, processes, proficiency in a variety of penetration testing tools;
2. Knowledge of common Web development languages (asp, php, jsp), experience with SQL-injection, XSS, common websecurity exploits and patches;
3. Experience with all kinds of operating systems and databases for common security vulnerabilities;
4. Good verbal and written language skills;
5. Be able to work in a team; individuals who lose trust, do not listen to the teamleader and not accepting the rules will be kicked out;
Work Location: Guangdong (OR Guangzhou Shenzhen)

Baochibaozhu package, Relatively free playing time.
Salary: monthly allocation of the total amount of work and cooperation share more than 1W.
Vacancies: 5 people
For candidates: first contact me (preferably work resume), after my check the resume will be passed to the head coordinator for arranging a personal meeting.
Salary: free meal and apartments, office location is in a senior villa suite of 200 square meters, computers are available but please bring your own hard drive with environment and tools you are familiar with. Even a single completed project will provide you with money for your monthly expenses.
Powerful background. No comments!

Tho who are competent, please contact:
Email: Infosec@cntv.cn QQ: admin@inessus.com”

And Mer4en7y’s replied to this job offer:

![Image of Mer4en7y's comment]

作者：mer4en7y   时间：2012-4-6 08:28

[Mer4en7y’s comment about job offer]

Which can be translated as: “Aren’t you recruiting people for APT? Guangzhou is too far, but anyway I support it”.

There are some interesting comments in the mentioned forum thread regarding reference “Powerful background” in job offer. People in the thread speculated that it could mean the work is supported by the government.

Mer4en7y is publishing an exploit:

"Winnti" More than just a game

Mer4en7y published a modified Perl script for network scan:

http://www.2cto.com/kf/201110/109200.html
Modify a perl scan script

2011-10-27 12:39:47

#! /usr/bin/perl

# #
# By www.hbjj.com www.2oto.com
# Modify: Mer4en7y
# Team: 90Sec
# #
use HTTP::Request;
use LWP::UserAgent;
system('cls');
print \n;
print '====== the directory scanned tools =============== \n';
print '====== modify: Mer4en7y ================ \n';
print '====== Team: 90Sec =============== \n';
print '====== scan after view fuck.txt === \n \n';
print 'Please enter the URL: \n';

Mer4en7y's network scanner on Perl

---

**Jimmycocowell Individual Activity**

Let’s continue with other places where delimiters have been found:

https://www.myspace.com/574064782/blog

---

Another bot control message by Wz
http://www.wuhanbike.net/home.php?mod=space&uid=15845&do=profile

Another bot control message by 奔跑 (Run)

http://jimycocowell.blogspot.ru/

Another bot control message by Jimycocowell
The attacker left two messages. The very first one is labeled as “first home” / “first love” and contains a ciphered C&C domain as described above, i.e. C&C domain is encrypted with RC4 algorithm and its hex binary value is presented in text format between delimiters.

But the next message dubbed “second” contains a ciphered C&C domain too but it is encoded in another way: The initial C&C domain is XORed with fixed byte value and the resulted data is transformed using BASE64 encoding. The resulted text is inserted between the same delimiters. By all appearances this method is used in the next version of the backdoor which is the subject of current research (see c_20100.NLS). It is also possible that programs with support of either this or that encryption could be used simultaneously in the frame of one attack. Between all found messages for the bot the second type of messages (BASE64) is significantly prevalent.

A link to this “Jimycocowell home” is also present at following place of “bitgodgod” user: http://www.blogger.com/profile/06442609461818597659

Bitgodgod and Bitbugbug

We have located one sample of Winnti malware with a hardcoded C&C: mail.7niu.com. Domaintools information about the domain:

- Domain Name : 7niu.com
- PunnyCode : 7niu.com
- Creation Date : 2006-06-11 00:00:00
- Updated Date : 2012-01-27 21:35:57
- Expiration Date : 2016-06-11 00:00:00

Registrant:
- Organization : qi tou niu
- Name : xibei jiao
- Address : beijing
- City : beijing
- Province/State : Beijing
- Country : CN
- Postal Code : 100000

http://www.blogger.com/profile/06442609461818597659
You can see how similar “bitbugbug” and “bitgodgod”. Both are directly related to Winnti activity. The email address “bit_bugbug@tom.com” also can be found on Chinese websites about home rentals:

http://oldhouse.0379home.com/RentView-1108.html

<table>
<thead>
<tr>
<th>[Personal rental] the Yawei property rental Yawei International Plaza apartment 411,126 m²</th>
<th>1000 January</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source:</strong></td>
<td>Personal</td>
</tr>
<tr>
<td><strong>County:</strong></td>
<td>Luolong</td>
</tr>
<tr>
<td><strong>Property Name:</strong></td>
<td>Yawei property</td>
</tr>
<tr>
<td><strong>Property Address:</strong></td>
<td>The Yawei international Square</td>
</tr>
<tr>
<td><strong>Floor:</strong></td>
<td>Layer 17, layer 0, layer basement</td>
</tr>
<tr>
<td><strong>Use of the area:</strong></td>
<td>126 m²</td>
</tr>
<tr>
<td><strong>Rent:</strong></td>
<td>1000 yuan / month</td>
</tr>
<tr>
<td><strong>Build time:</strong></td>
<td>In 2009</td>
</tr>
<tr>
<td><strong>Published:</strong></td>
<td>2009-7-18 19:51</td>
</tr>
<tr>
<td><strong>Facilities:</strong></td>
<td>Water and electricity</td>
</tr>
<tr>
<td><strong>Traffic conditions:</strong></td>
<td>Very convenient</td>
</tr>
<tr>
<td><strong>Remarks:</strong></td>
<td>Luoyang City, Luolong Yawei International Plaza apartment (New Area Dennis), 4th Floor, # 411, Pro road north, snugly advertising, 3233985570 Mr. Sun Contact me when instructions see from 0379home, thank you!</td>
</tr>
<tr>
<td><strong>Contact:</strong></td>
<td>Mr Sun</td>
</tr>
<tr>
<td><strong>Gender:</strong></td>
<td>Male</td>
</tr>
<tr>
<td><strong>Tel:</strong></td>
<td>13233985570</td>
</tr>
<tr>
<td><strong>E-mail:</strong></td>
<td><a href="mailto:bit_bugbug@tom.com">bit_bugbug@tom.com</a></td>
</tr>
</tbody>
</table>
Yang Individual Activity

We have located another individual calling himself Yang. He distributed bot control commands and was quite active on the internet as well.

http://yang8559420.blog.163.com/

Yang8559420 blog

Search for “yang8559420” brought some results:

Yang is a distributor of resources (maps or programs) for applications based on ArcGIS Engine (http://www.esri.com/software/arcgis/arcgisengine)

http://shop65775432.taobao.com/?spm=a1z0b.7.2-2442034955.3.rflsIS
Yang offered ArcGIS engine sourcecode for sale

Information about the seller:

http://shop65775432.taobao.com/view_page-74445421.htm

Yang8559420 trader profile (Chinese)
Yang8559420 trader profile (English Google-translation)

Yang is certified at alipay.com (see field “Certification” above):


Alipay real-name authentication introduced

Alipay real-name authentication Alipay (China) Network Technology Co., Ltd. to provide an identification service. Alipay real-name authentication to verify membership information and bank account information. Equivalent owned by Alipay real-name authentication an Internet identity cards; can shop Taobao and many other e-commerce sites, selling goods; increase PayPal account to pay the credit of the owners. (Such as PayPal account has not been through the real-name authentication, you need to modify the name or identity card number, may apply to the real-name authentication PayPal account to pay by name or identity card number can be changed.)

Alipay certification

Yang left some feedback about a coat:

Yang comments on the internet (private life related)

Yang is selling glasses:

http://webcache.googleusercontent.com/search?q=cache:susBSuR_5zoJ:re.taobao.com/search%3Frefpid%3Dmm_16823808_2252954_8791633%26keyword%3D%25D5%25E6%25CB%25BF%2520%25D2%25C2%25B9%252520%25D6%26back%3Dlo1%253D0%2526lo2%253D0%26nt%3D1%26isinner%3D1%26yp4p_page%3D3%26posid%3D7+%22yang8559420%22&cd=14&hl=ru&ct=clnk&gl=ru
"Winnti" | More than just a game


User: lovemeyang (probably related to Yang). Signature is a message for a bot:


Another message for bot by lovemeyang

So, both Yang8559420 and Lovemeyang messages go with signature:
Same signature used by Yang8559420 and Lovemeyang


Signature by Lovemeyang

Search for “lovemeyang” returned too much data, making it difficult to filter out those identifying possible attackers – false positives are highly-probable. However, it’s worth mentioning that the following link refers to an account
with the “lovemeyang” username and the user has earlier posted blogs relating to IT-security, so possibly the user is that Yang who is involved in the attack:

http://lovemeyang.blog.51cto.com/659880/195451

Yang and relation to a malware
Conclusions

Our research revealed long-term oriented large scale cyber-espionage campaign of a criminal group with Chinese origins. These attacks are not new, many other security researchers have published details of various cybercriminal groups coming from China. However, the current hacking group has distinguishable features that make it stand out among others:

- Massive abuse of digital signatures; the attackers used digital signatures of one victim company to attack other companies and steal more digital certificates;
- Usage of kernel level 64-bit signed rootkit;
- Abusing great variety of public Internet resources to store control commands for the malware in an encrypted form;
- Sharing/selling stolen certificates to other groups that had different objectives (attacks against Uyghur and Tibetan activists);
- Stealing source code and other intellectual property of software developers in online gaming industry.

The Winnti hacking group is not the first and not the last. By making our research paper available to the public, we hope that it will not only spread the knowledge among security researchers but also will help system administrators and security officials in all type of organizations around the world to learn the tactics and tools of the perpetrators. We hope that our shared knowledge will help to better protect IT infrastructure. We also hope that our message will reach Chinese law enforcement agencies. If the current research is not enough to initiate criminal investigation, we hope that it will be enough at least to make some checks and probably prevent other malicious activity from reaching out foreign countries and business within China.
Appendix

Winnti MD5s:

Winnti 1.0

Win32 samples

006c456149da562a4e337e2c146cf1a
024cc9872d9f413292d0f952920547ca
0613d67070679fb97ddefc5973c4d604
0630a443bd0102647ca1707cdf7f8c35
0751ca6f8b652cae6f2b650f0cf9036a
095a6a3b6eba996d2786b5ec919b1a7e
0af3761919bffaa0019e7899333846e2b7
0f3c15de074f934499f5bcb095d5557f
11ed89f0ab17c3f973e2bf970879661a
128cb2a5de0d422d69ab6d23ebb0aa
17c72e0cde2e4019a6885f8188ac410
1881363417608b4ad14abebcaefcb57
1a5da85093681e685893547d1a2eaf
1ab7360a9438fb816f01ac00c179d9a4
1d688ca3148df378a15796f43242b77c
212bb6c7ec7848b73ae6bf211cef7615
296220a85742a8722b1335977dd98251
379251974ebcd5c397f92ca45bb9620d
38fb6993c394ea6df0123544be4e77
3c722f0bea82e5bb89587f7fab012c911
3ecbc145dd593ee431145dd84e1e50cb
403f8fb208d4b50e1f5f765811fda174
41ff77ea7d4960c75d272a6a6fc31e7c
4402db68df6682bfe3e1e855a2474444
4722c665196fb6c7450980eafde6ac86
4e8f1c053dbe449c93f04e11d4afa352
4f213f9f187a65ce437157a3e7d253c0
50635147a579a8c8859a49c6099f9d32
50678adefc49735a4f236e06e83c089d
5156bc9f1d8ef1c1055933bb9c89c91
516fe9d2fe8b047fa8ba93692f44482
5171b030750f364a3459d5de22bc875d
5a93c03ddfe3e6eb2573b72d12ebe0e5
5db7ba6e771ce48c623a48fb4740b
629c0a9d3d0f471005c87d06ae45113
64d225a757686db6263e5df919e9dfd6
6db0e62dad6407f666aa0e4b995e7f
7460f35e3b4db92bc4cbb6c3f3ac
7529e41a101170eadb83cbb77bf29e65
814001293e4a50d12c5563e0b95ff6e
81b27822a6619a7c78eebbd6dc4bb89d
9251ff253c38c437bad4926378981ad0
9a575f37ffa684d56df15febc24b8f3
Winnti 1.1

Win32 samples

Drivers
### Droppers

<table>
<thead>
<tr>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>509c562db69f8332b9fc3298236e8ffa</td>
</tr>
<tr>
<td>130a799edeb0753164cdb76ccf8fd64c</td>
</tr>
<tr>
<td>5654424ea88de69d5c6031f7009f0428</td>
</tr>
</tbody>
</table>

### Winnti 1.2

<table>
<thead>
<tr>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>0393eebedbde6e5ee868f81ac024b401</td>
</tr>
<tr>
<td>36711896cfeb67f599305b590f195aec</td>
</tr>
<tr>
<td>43da75e78e7e1893dce276bd5b2e680</td>
</tr>
<tr>
<td>535ede2d69a7e07a097ef6648b12e417</td>
</tr>
<tr>
<td>8acb42de94427141f7cafeed74f9fc43</td>
</tr>
<tr>
<td>a0a96138b57ee24eed31b652ddf60d4e</td>
</tr>
<tr>
<td>d350ae5dc15bcc18fde382b84f4bb3d0</td>
</tr>
<tr>
<td>e252d9ec4bca3d261f5acdd33bfd1cb</td>
</tr>
<tr>
<td>f454ba447eef28f96dafa3398df82a7e</td>
</tr>
<tr>
<td>011815cb37f49a1d143db995a5e705f</td>
</tr>
<tr>
<td>115dc2627483aba7119ad4ceab1e042a</td>
</tr>
<tr>
<td>18677c3a2af1476aa8cbc73cf74d8c1</td>
</tr>
<tr>
<td>1b0753f717d7a33defc389e399b20d57</td>
</tr>
<tr>
<td>29525be71ba4846739e553a0835ab460</td>
</tr>
<tr>
<td>2989b78ac3a752bf792ac9ac066df0</td>
</tr>
<tr>
<td>2ff739a927b62d4b7096e636951b77d</td>
</tr>
<tr>
<td>3047ed57acacc30c2327e74070b3864b7</td>
</tr>
<tr>
<td>3d107d5bdf554c6ae8d05c886080a18d</td>
</tr>
<tr>
<td>4197499923ab6125e2ee5e950b21ec91</td>
</tr>
<tr>
<td>453021b8cc10f9077fa80d09c631d</td>
</tr>
<tr>
<td>4732d2056060c66f46caded82954836e</td>
</tr>
<tr>
<td>4d028c7a47c1cb0d00e894ad351a61996</td>
</tr>
<tr>
<td>6e9b47f2ae1f9e7260b8793f35fbbd3a</td>
</tr>
<tr>
<td>8a1d1965b2d8501e692394bb801f58ca</td>
</tr>
<tr>
<td>a0629962c34ed9594b18493f459560a7</td>
</tr>
<tr>
<td>ada515709be09e495bc9c1206069e796</td>
</tr>
<tr>
<td>bfcfd3417b513a6c3fed4b5466055d939</td>
</tr>
</tbody>
</table>

### Samples

<table>
<thead>
<tr>
<th>Hash</th>
</tr>
</thead>
<tbody>
<tr>
<td>60bd5a9ab78f6c614b824ddcb47dfd7c</td>
</tr>
<tr>
<td>8f54c08ee45a8d5eb31d05dbab4b561</td>
</tr>
<tr>
<td>15d6249e0e7e03b3e00cc3917431cf64</td>
</tr>
<tr>
<td>4fbb502ba8c7e8d81ec98a5974b9001a</td>
</tr>
<tr>
<td>5618bc41af50c790c8e8680ba30030ed</td>
</tr>
<tr>
<td>7d51ea0230d4692eeedc2d5a4cd66d2d</td>
</tr>
<tr>
<td>961954bbc411d4eaf7d2efad94a6e160</td>
</tr>
<tr>
<td>c206992f7c6836ec6a227a6e29ae7609</td>
</tr>
</tbody>
</table>
# Winnti 2.0

<table>
<thead>
<tr>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>06d8b1468f09d10aa5c4b115544ccc6e</td>
</tr>
<tr>
<td>0cd07490fc02e2a602781bb939d0bc3d</td>
</tr>
<tr>
<td>2d0950f69e206486c5272f2b0fc3aa22</td>
</tr>
<tr>
<td>3358c54a22d186ec9de0f15bc4bb2698</td>
</tr>
<tr>
<td>35bdc5a2acbf35bd9f9b9169e1a47d3e7</td>
</tr>
<tr>
<td>5778178a1b5259c3127b678a49cd23e53</td>
</tr>
<tr>
<td>6dfcc4c8edc77642f15592143f34569</td>
</tr>
<tr>
<td>9a83cd3f8e619c8b1b38b0b5ceeeea357</td>
</tr>
<tr>
<td>afe4ec9a88f84f9b9c1eb0f3ff47a12b</td>
</tr>
<tr>
<td>B0BD6C2157C20B23FD23D77FA26F3BC</td>
</tr>
<tr>
<td>bbb9b5c7a59b98f18b06344ac8980ef</td>
</tr>
<tr>
<td>d23237edbdcc4118b538454b45c00021</td>
</tr>
<tr>
<td>d4a2060a5086c56f7ff65ea65de81ff</td>
</tr>
<tr>
<td>dc22d742a15f8d6d8edf49d1c8cc8be9</td>
</tr>
<tr>
<td>e7e5c5c991e6d66fca16c988c891e10f</td>
</tr>
<tr>
<td>f4c9bc4f045b90c496d4b75398dfa5c</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>04f3fbaaaaf5026df29e0d7d317194043</td>
</tr>
<tr>
<td>07e40089cdf338e8d1423bd97332a4d</td>
</tr>
<tr>
<td>0b105c6ecedfe572ac7db52135aa47ef</td>
</tr>
<tr>
<td>7024ea8285c8e098829ac8f2b1de4455</td>
</tr>
</tbody>
</table>

## Compromised certificates

<table>
<thead>
<tr>
<th>Company</th>
<th>Serial number</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTsoft Corp</td>
<td>30 d3 fe 26 59 1d 8e ac 8c 30 66 7a c4 99 9b d7</td>
</tr>
<tr>
<td>Kog Co., Ltd.</td>
<td>66 e3 f0 b4 45 9f 15 ac 7f 2a 2b 44 99 0d d7 09</td>
</tr>
<tr>
<td>LivePlex Corp</td>
<td>1c aa 0d 0d ad f3 2a 24 04 a7 51 95 ae 47 82 0a</td>
</tr>
<tr>
<td>MGAME Corp</td>
<td>4e eb 08 05 55 f1 ab f7 09 bb a9 ca e3 2f 13 cd</td>
</tr>
<tr>
<td>Rosso Index KK</td>
<td>01 00 00 00 00 01 29 7d ba 69 dd</td>
</tr>
<tr>
<td>Sesisoft</td>
<td>61 3e 2f a1 4e 32 3c 69 ee 3e 72 0c 27 af e4 ce</td>
</tr>
<tr>
<td>Wemade</td>
<td>61 00 39 d6 34 9e e5 31 e4 ca a3 a6 5d 10 0c 7d</td>
</tr>
<tr>
<td>YNK Japan</td>
<td>67 24 34 0d db c7 25 2f 7f b7 14 b8 12 a5 c0 4d</td>
</tr>
<tr>
<td>Guangzhou YuanLuo</td>
<td>0b 72 79 06 8b eb 15 ff e8 06 0d 2c 56 15 3c 35</td>
</tr>
<tr>
<td>Fantasy Technology Corp</td>
<td>75 82 f3 34 85 aa 26 4d e0 3b 2b df 74 e0 bf 32</td>
</tr>
<tr>
<td>Neowiz</td>
<td>5c 2f 97 a3 1a bc 32 b0 8c ac 01 00 59 8f 32 f6</td>
</tr>
</tbody>
</table>
## Winnti C&Cs

<table>
<thead>
<tr>
<th>Winnti 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>newpic.dyndns.tv</td>
</tr>
<tr>
<td>update.ddns.net</td>
</tr>
<tr>
<td>nd.jcrsoft.com</td>
</tr>
<tr>
<td>cc.nexoncorp.us</td>
</tr>
<tr>
<td>98.126.36.202</td>
</tr>
<tr>
<td>kr.zzsoft.info</td>
</tr>
<tr>
<td>as.cjinternet.us</td>
</tr>
<tr>
<td>ca.zzsoft.info</td>
</tr>
<tr>
<td>sn.jcrsoft.com</td>
</tr>
<tr>
<td>lp.apanku.com</td>
</tr>
<tr>
<td>sshd.8866.org</td>
</tr>
<tr>
<td>ftppd.6600.org</td>
</tr>
<tr>
<td>tcpiah.googleclick.net</td>
</tr>
<tr>
<td>rss.6600.org</td>
</tr>
<tr>
<td>lp.zzsoft.info</td>
</tr>
<tr>
<td>lp.gasoft.us</td>
</tr>
<tr>
<td>eya.jcrsoft.com</td>
</tr>
<tr>
<td>ftppd.9966.org</td>
</tr>
<tr>
<td>kr.xxoo.co</td>
</tr>
<tr>
<td>wi.gcgame.info</td>
</tr>
<tr>
<td>tcp.nhn.tech.com</td>
</tr>
<tr>
<td>ka.jcrsoft.com</td>
</tr>
<tr>
<td>my.zzsoft.info</td>
</tr>
<tr>
<td>jp.jcrsoft.com</td>
</tr>
<tr>
<td>su.cjinternet.us</td>
</tr>
<tr>
<td>vn.gcgame.info</td>
</tr>
<tr>
<td>ap.nhn.tech.com</td>
</tr>
<tr>
<td>ru.gcgame.info</td>
</tr>
<tr>
<td>kr.jcrsoft.com</td>
</tr>
<tr>
<td>wm.ibm-support.net</td>
</tr>
<tr>
<td>fs.nhn.tech.com</td>
</tr>
<tr>
<td>docs.nhnclass.com</td>
</tr>
<tr>
<td>rh.jcrsoft.com</td>
</tr>
<tr>
<td>wm.nhn.tech.com</td>
</tr>
<tr>
<td>wm.myxxoo.com</td>
</tr>
<tr>
<td>ka.zzsoft.info</td>
</tr>
<tr>
<td>ad.jcrsoft.com</td>
</tr>
<tr>
<td>my.gasoft.us</td>
</tr>
<tr>
<td>Domain Name</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>gunz.gcgame.info</td>
</tr>
<tr>
<td>dell-support.org</td>
</tr>
<tr>
<td>t3.jcrsoft.com</td>
</tr>
<tr>
<td>kr.hja63.com</td>
</tr>
<tr>
<td>dbo.gcgame.info</td>
</tr>
<tr>
<td>2m.reegame.net</td>
</tr>
<tr>
<td>ns1.msftncsl.com</td>
</tr>
<tr>
<td>update.reegame.net</td>
</tr>
<tr>
<td>pop.hja63.com</td>
</tr>
<tr>
<td>imap.gasoft.us</td>
</tr>
<tr>
<td>dns.naverpulic.com</td>
</tr>
<tr>
<td>pda.zzsoft.info</td>
</tr>
<tr>
<td>pop.cjinternet.us</td>
</tr>
<tr>
<td>bar.gasoft.us</td>
</tr>
<tr>
<td>hja63.com</td>
</tr>
<tr>
<td>god.zzsoft.info</td>
</tr>
<tr>
<td>goqc.xxoo.co</td>
</tr>
<tr>
<td>apps.mynetav.net</td>
</tr>
<tr>
<td>ns3.nhnclass.com</td>
</tr>
<tr>
<td>tug.mynetav.net</td>
</tr>
<tr>
<td>vip-webmail.com</td>
</tr>
<tr>
<td>mail.7niu.com</td>
</tr>
<tr>
<td>game.joymax.in</td>
</tr>
<tr>
<td>tho.hja63.com</td>
</tr>
<tr>
<td>zb.mynetav.net</td>
</tr>
<tr>
<td>vtc.gasoft.us</td>
</tr>
<tr>
<td>tv3.mynetav.net</td>
</tr>
<tr>
<td>hk.hja63.com</td>
</tr>
<tr>
<td>ad.gasoft.us</td>
</tr>
<tr>
<td>ns5.msftncsl.com</td>
</tr>
<tr>
<td>ftp.zzsoft.info</td>
</tr>
<tr>
<td>sm.gcgame.info</td>
</tr>
<tr>
<td>eudb.reegame.net</td>
</tr>
<tr>
<td>tech.ibm-support.net</td>
</tr>
<tr>
<td>gm.gcgame.info</td>
</tr>
<tr>
<td>winlogon.net</td>
</tr>
<tr>
<td>iyy.conimes.com</td>
</tr>
<tr>
<td>ru.gcgame.info</td>
</tr>
<tr>
<td>oa.nexoncorp.us</td>
</tr>
<tr>
<td>cjinternet.us</td>
</tr>
<tr>
<td>wm.ibm-support.net</td>
</tr>
<tr>
<td>hp-supports.com</td>
</tr>
<tr>
<td>pass1.hangame.co.uk</td>
</tr>
<tr>
<td>mail.cjinternet.us</td>
</tr>
<tr>
<td>tt.xxoo.co</td>
</tr>
<tr>
<td>e.jcrsoft.com</td>
</tr>
<tr>
<td>gamenow.8800.org</td>
</tr>
<tr>
<td>googlefiles.net</td>
</tr>
<tr>
<td>ns4.msftncsl.com</td>
</tr>
</tbody>
</table>
gf.jcrsoft.com
sg.xxoo.co
ns3.nhnclub.com
wog.zzsoft.info
ssl.msftncsl.com
ns7.msftncsl.com
udp.nhntech.com
ad.jcrsoft.com
ns6.msftncsl.com
ibm-support.net
gf.zzsoft.info
kerberos.dnsalias.com
ns1.nhnclub.com
imap.zzsoft.info
gongyi.co
jcrsoft.com
uni.vip-webmail.com
smtp.jcrsoft.com
cc.nexoncorp.us
imm.conimes.com
mail.hja63.com
pass2.googletrait.com
club.cjinternet.us
mail.nexoncorp.us
as.cjinternet.us
service.dell-support.org
service.googlefiles.net
ftp.nexoncorp.us
e.gcgame.info
hansoft.sunsb.net
www.jcrsoft.com
ftpd.6600.org
sshd.8866.org
cpu.4pu.com
nx2.joymax.in
av.gcgame.info
dl-adobe.com
cj.jcrsoft.com
ro.myxxoo.com
rh.gcgame.info
cc.xxoo.co
swordwind.net
lp.xxoo.co
brqc.xxoo.co
ava.apanku.com
wi.gcgame.info
zm.gasoft.us
as.xxoo.co
gh.gasoft.us
baesystems.conimes.com
ns2.nhnclub.com
intercpu.com
e.hja63.com
pda.gasoft.us
wsafelogin.com
mail.nexongame.net
smtp.cjinternet.us
wm.nhntech.com
www.gcgame.info
ix.xxoo.co
support.dell-support.org
han.zzsoft.info
imap.hja63.com
nhntech.com
qc.xxoo.co
ip.xxoo.co
sl.myxxoo.com
mail.joymax.in
help.googleclick.net
www.nexoncorp.us
conimes.com
usa.xxoo.co
my.reegame.net
login.joymax.in
hsb.mynetav.net
docs.naverpulic.com
fax.nexoncorp.us
mail.jcrsoft.com
guys.mynetav.net
google.x3322.org
jc.nhn.tech.com
roqc.xxoo.co
ws.gcgame.info
xss.gongyi.co
new.java-ssl.com
ava.zzsoft.info
eya.jcrsoft.com
gn.xxoo.co
crl.nhn.tech.com
tah.xxoo.co
dns.nhnclass.com
zzsoft.info
nx.xxoo.co
ns2.naverpulic.com
pop.zzsoft.info
on.xxoo.co
pwd.nhn.tech.com
ftp.gcgame.info
nx2.hangame.co.uk
he.xxoo.co
hk.zzsoft.info
nhnclass.com
sellsads.sells-it.net
wapqq.3322.org
kr.reegame.net
nt.nexoncorp.us
tcp.nhnentech.com
www.hja63.com
aion.reegame.net
su.cjinternet.us
get.java-ssl.com
eudb.nexongame.net
nsqc.xxoo.co
mail.gasoft.us
kr.jcrsoft.com
ads01.mynetav.net
gm.gasoft.us
a1.reegame.net
smtp.gcgame.info
pda.jcrsoft.com
kor.xxoo.co
ns9.msftncsl.com
nx.jcrsoft.com
nexon.hangame.co.uk
smtp.gasoft.us
ns2.java-ssl.com
alta.apanku.com
nexon.joymax.in
my.gasoft.us
dns2.msftncsl.com
ckts.mynetav.net
pass1.googletrait.com
dns.nhnclub.com
kr.zzsoft.info
mir.reegame.net
jrun.hja63.com
wm.googleclick.net
bot.dongevil.info
mail.zzsoft.info
nexononline.com
tv.mynetav.net
e.gasoft.us
xy.hja63.com
www.apanku.com
usa.nexongame.net
ftp.gasoft.us
ogp.reegame.net
kog.jcrsoft.com
www.joymax.in
br.xxoo.co
ftp.cjinternet.us
qc.zzsoft.info
pay.gcgame.info
pp.ibm-support.net
blog.mynetav.net
ijj.conimes.com
tank.hja63.com
lp.gasoft.us
nx3.googlefiles.net
pass1.nexongame.net
gqc.xxoo.co
br.reegame.net
ftpd.9966.org
kr.xxoo.co
offices.dyndns-office.com
hansoft.does-it.net
gasoft.us
docs.nhnclub.com
sf.cjinternet.us
pass2.nexongame.net
update-microsoft.com
ka.jcrsoft.com
us.xxoo.co
myav.mynetav.net
w53.myxxoo.com
isatap.dyndns.org
tt.conimes.com
vn.gcgamexxoo.com
ap.nhnitech.com
bot.jgame.in
l.xxoo.co
ftp.hja63.com
mail.msftncsl.com
dns01.dyndns-work.com
service.hp-supports.com
ns2.nhnclass.com
fax.cjinternet.us
nx2.intercpu.com
windows.doomdns.com
btg.mynetav.net
xxoo.co
mynetav.net
mini.msftncsl.com
pass2.hangame.co.uk
webadmin.dnsdojo.net
imap.gcgame.info
joymax.in
udp.jjevil.com
www.reegame.net
myxxoo.com
iss.conimes.com
ads01.dyndns-web.com
www.mynetav.net
dns.msftncsl.com
pop.jcrsoft.com
ball.reegame.net
lyto.zzsoft.info
rw.nhntech.com
els.jcrsoft.com
a1.googletrait.com
googletrait.com
w80.xxoo.co
scvhosts.com
nexon.nexongame.net
pic.4pu.com
q.gcgame.info
dbo.jcrsoft.com
ns2.msftncsl.com
ynk.xxoo.co
tw.hja63.com
pass1.reegame.net
my.zzsoft.info
www2.mynetav.net
www.nexongame.net
id.naverpublic.com
roap.myxxoo.com
openhost.webhop.net
mir2.nexongame.net
imap.jcrsoft.com
pop.gasoft.us
bar.zzsoft.info
game.nexongame.net
fs.nhntech.com
osk.zzsoft.info
docs.nhnclass.com
t3.nhntech.com
ahn.gasoft.us
officess.dyndns-office.com
new.nexoncorp.us
dbo.zzsoft.info
w.zzsoft.info
lp.gcgame.info
ro.hja63.com
gcgame.info
xl.apanku.com
web-games.us
sl.xxoo.co
login.hangame.co.uk
ro.xxoo.co
dbo.gasoft.us
moon.reegame.net
egi.mynetav.net
vn.jcrsoft.com
ftp.mynetav.net
us.nhntech.com
masternow.webhop.net
file.googlefiles.net
holleword.3322.org
est.zzsoft.info
apanku.com
help.ibm-support.net
tw.reegame.net
est.gasoft.us
mg.jcrsoft.com
lp.apanku.com
smtp.hja63.com
xnews.myPicture.info
lp.zzsoft.info
nx3.interdriver.net
rss.6600.org
fn.hja63.com
usp.xxoo.co
ads01.dyndns-pics.com
oky.mynetav.net
pop.nexoncorp.us
naverpulic.com
pop.gcgame.info
ap.googleclick.net
haj.mynetav.net
ac.xxoo.co
mini.nexongame.net
udp.googleclick.net
nd.xxoo.co
new.myxxoo.com
rh.jcrsoft.com
wm.xxoo.co
dns.java-ssl.com
wyqc.xxoo.co
q.zzsoft.info
pass1.joymax.in
item.ItemDB.com
reegame.net
mailes.dyndns-mail.com
nd.gasoft.us
a.zzsoft.info
w53.xxoo.co