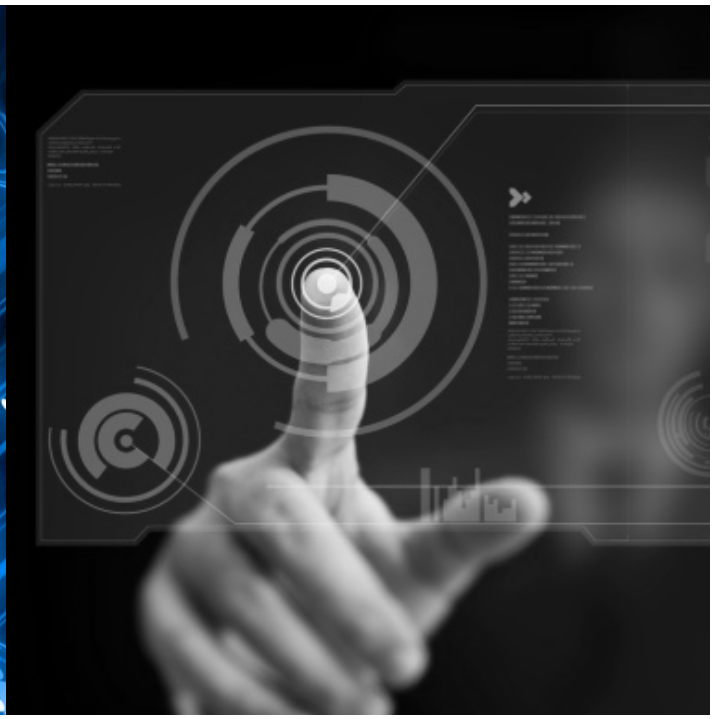
# RSA Incident Response: An APT Case Study

## RSA Security

## 8 April 2015



## Table of Contents


**1. Executive Summary ..... 5**

**2. Security Analytics and ECAT Deployment ..... 7**

**3. Analysis Methodology ..... 8**

**4. Case Study Technical Details ..... 9**

**4.1 Initial Consultation ..... 9**

**4.2 Incident Response ..... 9**

4.2.1 ECAT Analysis – System XX13 ..... 10

4.2.2 ECAT Analysis – System XXDEV3 ..... 12

4.2.3 ECAT Analysis – Trojan.FF-RAT ..... 15

4.2.4 ECAT Analysis – System XXXXNAPP02 ..... 16

4.2.5 ECAT Analysis – Recycler folder ..... 18

4.2.6 ECAT Analysis – System XXME ..... 18

4.2.7 ECAT Analysis – System XX22 ..... 21

4.2.8 ECAT Analysis – Hunting with InstantIOCs. ..... 22

**4.3 SA Analysis – Trojan.Lurker ..... 23**

**4.4 Parallel Detection with Security Analytics ..... 26**

**5. Trojan Families ..... 29**

**5.1 Trojan.Lurker ..... 29**

**5.2 Trojan.SurperhardCorp ..... 30**

**5.3 Trojan.Derusbi ..... 30**

**5.4 Trojan.HiKiT ..... 31**

**5.5 Trojan.FF-RAT ..... 32**

**5.6 Trojan.PlugX ..... 34**

**5.7 Trojan.Gh0st ..... 34**

**5.8 Trojan.PoisonIvy ..... 35**

**6. Conclusion ..... 37**

**7. Appendix I ..... 38**

Table 1: SA beacon detection rules ..... 26

Table 2: Trojan.Lurker files details and C2 channels ..... 29

Table 3: Trojan.SuperhardCorp file details and C2 channels ..... 30

Table 4: Trojan.Derusbi - file details and C2 channels ..... 30

Table 5: Trojan.Hikit - file details and C2 channels ..... 31

Table 6: Trojan.FF-RAT - file details ..... 32

Table 7: Trojan.FF-RAT - file details ..... 33

Table 8: Trojan.FF-RAT - file details ..... 33

Table 10: Trojan.FF-RAT - file details ..... 33

Table 11: Trojan.FF-RAT - file details ..... 33

Table 12: Trojan.PlugX - file details and C2 channels ..... 34

Table 13: Digitally Signed Trojan.Gh0st file details ..... 35

Table 14: Unsigned Trojan.Gh0st file details ..... 35

Table 15: Trojan.PoisonIvy - file details ..... 35

Figure 1: RSA vs traditional analysis comparison ..... 5
Figure 2: Network diagram with capture points ..... 7
Figure 3: IR Competencies ..... 8
Figure 4: ShimCache results from memory analysis ..... 9
Figure 5: ECAT File Properties Window ..... 10
Figure 6: ECAT MFT Viewer - Trojan.Hikit ..... 10
Figure 7: Obfuscated and deobfuscated Trojan.Hikit config file ..... 11
Figure 8: Yara signature for Trojan.Hikit ..... 11
Figure 9: Yara hit on Trojan.Hikit ..... 12
Figure 10: ECAT MFT time analysis - Trojan.Hikit ..... 12
Figure 11: ECAT MFT analysis - At#.job files ..... 12
Figure 12: ECAT MFT time analysis ..... 13
Figure 13: ECAT MFT analysis - Trojan.FF-RAT ..... 13
Figure 14: Digital Signature details of Trojan.FF-RAT ..... 13
Figure 15: Yara signature for Trojan.FF-RAT ..... 14
Figure 16: ECAT MFT time analysis on At3.job file ..... 14
Figure 17: SA analysis - Trojan.FF-RAT beaconing ..... 14
Figure 18: ECAT analysis - Trojan.FF-RAT ..... 15
Figure 19: ECAT analysis - filtering infected hosts ..... 15
Figure 20: ECAT analysis - requesting files enterprise wide ..... 15
Figure 21: ECAT analysis - systems infected with Trojan.FF-RAT ..... 16
Figure 22: ECAT MFT time analysis - Time stomping ..... 16
Figure 23: ECAT analysis - Trojan.Lurker2 ..... 17
Figure 24: Yara Signature - Trojan.Lurker2 ..... 17
Figure 25: Yara signature - Trojan.DerusbiAP32 ..... 18
Figure 26: ECAT analysis - files at root of Recycler folder ..... 18
Figure 27: ECAT analysis - files at root of Recycler folder ..... 19
Figure 28: ECAT analysis - Trojan.Gh0st ..... 19
Figure 30: ECAT analysis - Trojan in cached files ..... 19
Figure 31: Relevant Internet History results ..... 20
Figure 32: Additional relevant Internet History results ..... 20
Figure 33: ECAT analysis - Systems infected with Trojan.Gh0st ..... 20
Figure 34: Trojan.Gh0st - de-obfuscated configuration file ..... 21
Figure 35: ECAT analysis - files at root of Recycler folder ..... 21
Figure 36: ECAT MFT time analysis - Trojan.PlugX ..... 21
Figure 37: ECAT MFT time analysis - Trojan.PoisonIvy ..... 22
Figure 38: ECAT analysis - filtering systems infected with PoisonIvy ..... 22
Figure 39: ECAT analysis - filtering systems infected with PoisonIvy ..... 22
Figure 40: ECAT analysis - systems infected with Trojan.Lurker ..... 23
Figure 41: ECAT analysis - systems infected with Trojan.Superhardcorp ..... 23
Figure 42: SA analysis - Trojan.Lurker ..... 24
Figure 43: SA analysis - Trojan.Lurker HTTP anomalies ..... 24
Figure 44: SA analysis - Trojan.Lurker C2 channel activity ..... 24
Figure 45: SA analysis - Trojan.Lurker C2 channel activity ..... 25
Figure 46: SA analysis - Trojan.Lurker C2 activity ..... 25
Figure 47 Suspicious TCP Beaconing ..... 27
Figure 48 Suspicious TCP Beaconing ..... 27
Figure 49 IP.Alias Resolution for drometic.suroot.com ..... 27
Figure 50 FF-RAT Encoded Beacon ..... 28

Figure 51 FF-RAT Decoded Beacon ..... 28
Figure 52 FF-RAT Detection Parser ..... 28
Figure 53: Trojan.Lurker - DES keys used by each variant ..... 29
Figure 54: Trojan.SuperhardCorp - binary snippet ..... 30
Figure 55: Trojan.DerusbiAP32 - configuration file ..... 31
Figure 56: Trojan.Hikit deobfuscated configuration file ..... 32
Figure 57: Trojan.Gh0st magic string ..... 35
Figure 58: Trojan.PoisonIvy - Memory snippet containing password ..... 35
Figure 59: PosionIvy server side ..... 36
Figure 60: Plaintext file ..... 38
Figure 61: aPACK file structure ..... 38
Figure 62: Trojan.FF-RAT configuration file structure ..... 39
Figure 63: Trojan.FF-RAT RC4 key example ..... 39
Figure 64: Trojan.FF-RAT RC4 decrypted configuration file ..... 39
Figure 65: RC4 decrypted configuration file with manually generated aPACK header ..... 40
Figure 66: appack.exe error message ..... 40
Figure 67: Disassembly of appack.exe ..... 41
Figure 68: Patching appack.exe ..... 41
Figure 69:RC4 decrypted and aPACK decompressed Trojan.FF-RAT configuration file ..... 42

# 1. Executive Summary

This case study contains information from an engagement that the RSA Incident Response (IR) team worked during the September to October 2013 timeframe. It highlights the analysis flow using two of our flagship products, Security Analytics (SA) and the Enterprise Compromise Assessment Tool (ECAT), for an Advance Persistent Threat (APT) intrusion investigation. These key technologies allow RSA analysts to process massive datasets and find forensically interesting artifacts in near real-time and more quickly than using standard incident response processes.

APT actors are typically state sponsored, highly skilled, and have the resources to maintain prolonged campaigns of attacks against their targets. Law Enforcement (LE), security researchers or other 3rd-party entities typically notify victims of APT intrusions, like the one in this case. When analysts initially start working with a customer, the intent is to verify the intelligence of the notification. Too narrow of a focus on specific threat actors and known Indicators of Compromise (IOC) can give the analyst a myopic view of the scope of the incident. This is where the traditional Incident Response process and the process employed by RSA diverge. Utilizing SA and ECAT in parallel, analysts are able to mark up their respective datasets and feed each other actionable intelligence. Given the forensic capabilities of the respective tools, a large majority of the Host Based triage analysis can be completed before ever requesting full disk images. Additionally, with the capability of examining the host in detail remotely, false positives commonly found in traditional IOC sweeps can be eliminated, reducing analytical work load.

Neither technology employed by RSA, ECAT[1] or Security Analytics, rely on static signatures from known IOCs. Instead, the technologies utilize a multi-layer approach to identify known good behavior and related binaries while the unknown and non-standard artifacts stand out. This allows the analyst to broaden their search and discover artifacts beyond the scope of the known. This workflow has been instrumental in many Incident Response engagements led by RSA; oftentimes there are multiple intrusion sets and older campaigns left behind in the environment, not discovered by traditional methods including Anti-Virus, or discovered during previous 3rd party response efforts. Figure 1 shows a timeline of a traditional response as compared to an incident response effort leveraging ECAT and Security Analytics.

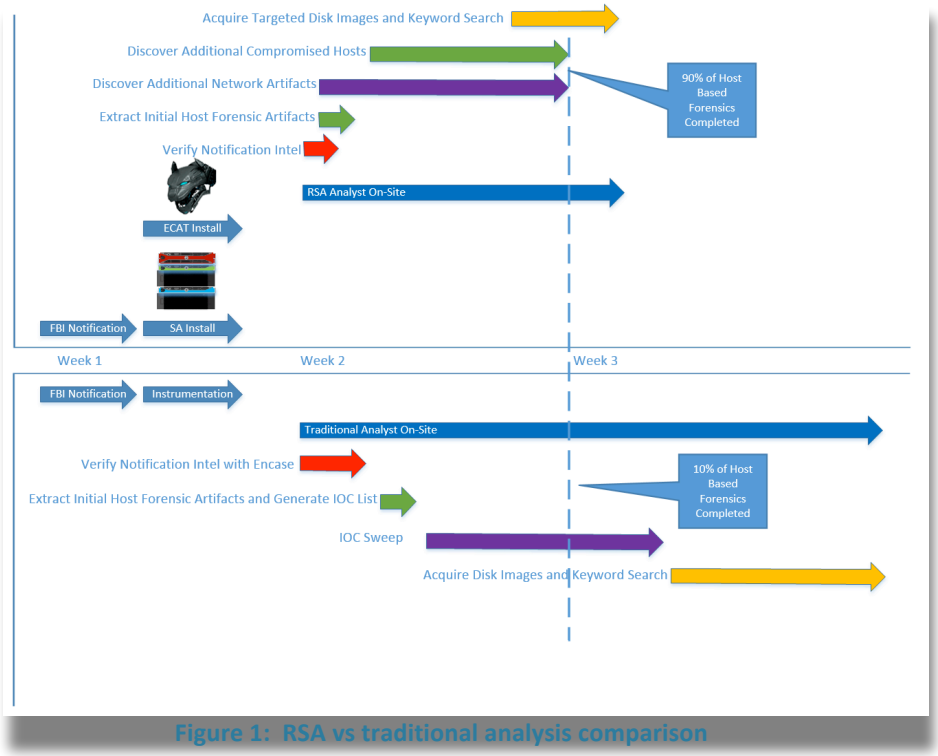



Figure 1: RSA vs traditional analysis comparison

[1] ECAT can be integrated with OPSWAT, which scans files with multiple AV engines, and Yara signatures, which can be created by the end user.

During this response effort RSA IR discovered multiple APT actors in the network, where at least one APT group had been present for over 3 years. At least 18% of the systems in the network had either been infected with Trojans deployed by APT actors, or had clear evidence they had been accessed for the purpose of stealing information. Eight different Trojan families were discovered during the investigation, some of which were capable of capturing keystrokes and providing GUI access to the infected system.

## 2. Security Analytics and ECAT Deployment

RSA utilized the victim's Security Analytics infrastructure capturing all enterprise ingress/egress traffic from 3 US locations, as depicted in Figure 1 below. The ECAT agent was deployed to about 500 Windows systems on the network.

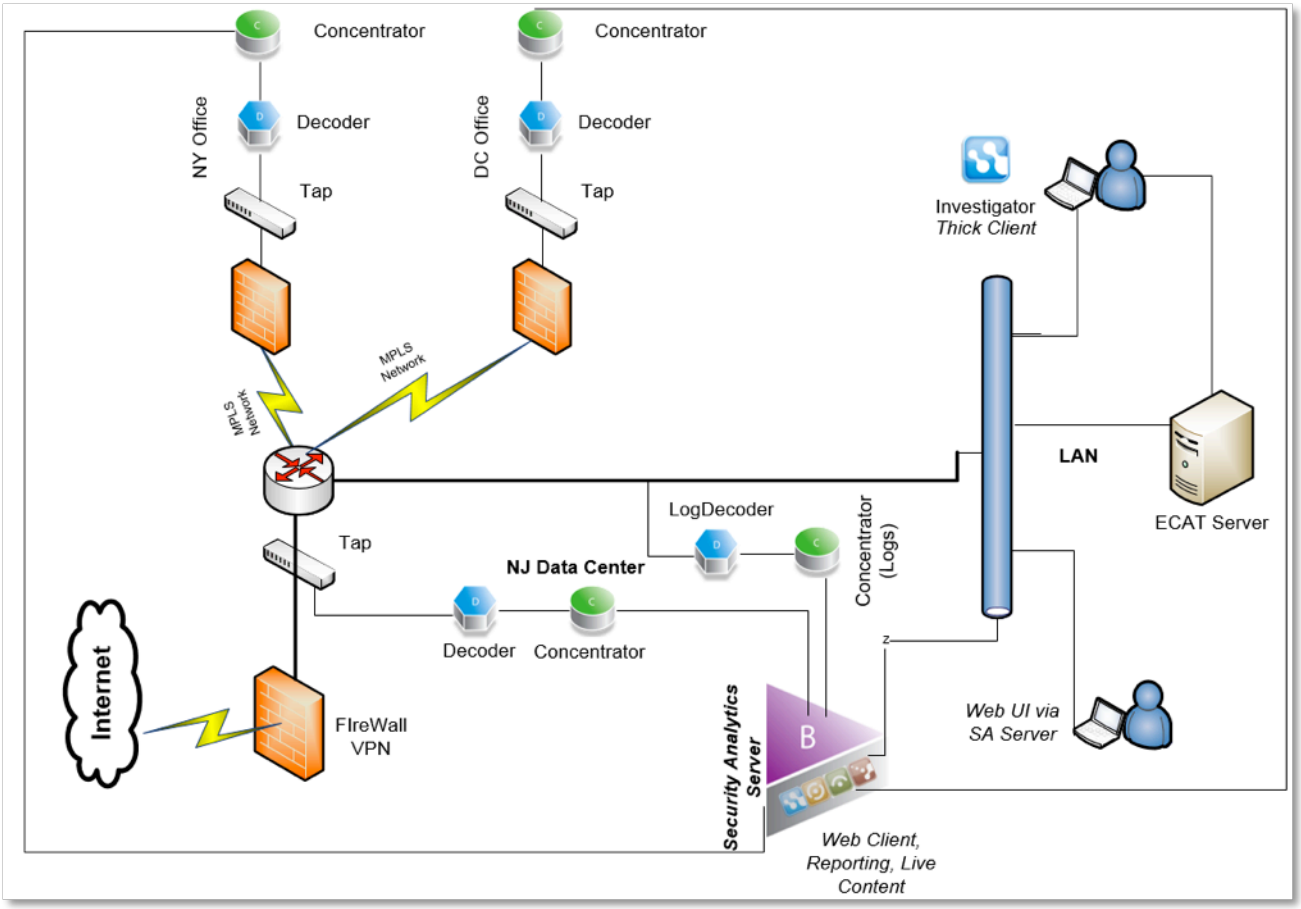



**Figure 2: Network diagram with capture points**

# 3. Analysis Methodology

RSA IR employs a methodology that is founded on industry standards. The holistic approach includes the following four core components:

- Intelligence gathering and research;
- Host-based forensic analysis;
- Network-based forensic analysis; and,
- Malware analysis.

Using an iterative approach, the RSA IR Team employs a repeatable process as needed upon the discovery of additional actionable data. Analysis is executed concurrently and therefore activities are performed simultaneously for maximum efficiency and effectiveness. To complete this work, RSA IR uses several commercial and open source forensic tools to recover artifacts to build a comprehensive understanding of the extent of compromise. In addition, the Team will leverage available tools and technologies in place within the enterprise to effectively utilize the IOCs to identify compromised systems and monitor for continued attacker presence. Using this methodology and associated proactive and reactive techniques, the Team is able to enhance overall situational awareness and ultimately provide answers to questions and actionable information allowing for tactical decision making in near real-time.

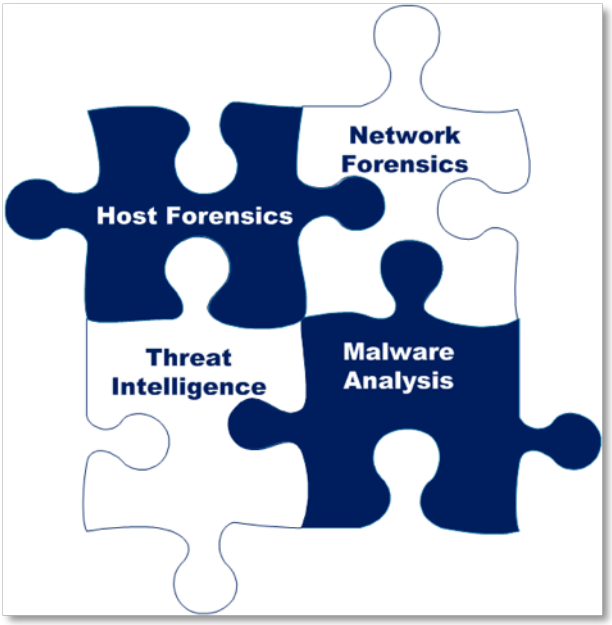



**Figure 3: IR Competencies**

Definitions:

- Threat Intelligence - open source research and real-time fusion of known threat data
- Host Forensics - analysis of file systems, logs, memory, and other volatile data to identify IOCs and/or suspicious activity
- Network Forensics - analysis of network traffic and logs to identify IOCs and/or suspicious activity
- Malware Analysis - analysis of code to identify tactics, IOCs and/or suspicious activity.

# 4. Case Study Technical Details

## 4.1 Initial Consultation

A Law Enforcement agency notified the victim (CompanyA) on July 2013 about potential unauthorized activity emanating from CompanyA's network. This LE notification mentioned that a rather large amount of data had been exfiltrated. CompanyA requested the help of the RSA IR team to determine the extent of the problem. CompanyA sent firewall logs for a 24-hour period encompassing the time in the notification from the LE agency. The RSA IR team was able to rapidly analyze the firewall logs, pinpointing several entries from a machine that had two large transfers. The two file transfer sizes combined matched the approximate amount of data that had been reported as being exfiltrated to an external system in the continental United States. The RSA IR team provided this information back to CompanyA, and requested that CompanyA provide a memory dump of that particular server for analysis.

The RSA IR team used Volatility to analyze the submitted memory dump. Very quickly, while parsing the Application Compatibility cache from the memory image, RSA IR confirmed this server likely had unauthorized activity based on locations and filenames that were executed on the system. Figure 4 below depicts a portion of the suspected tools that were executed on the system. These filenames and location have been previously associated with APT actor tools, techniques and procedures (TTP's).

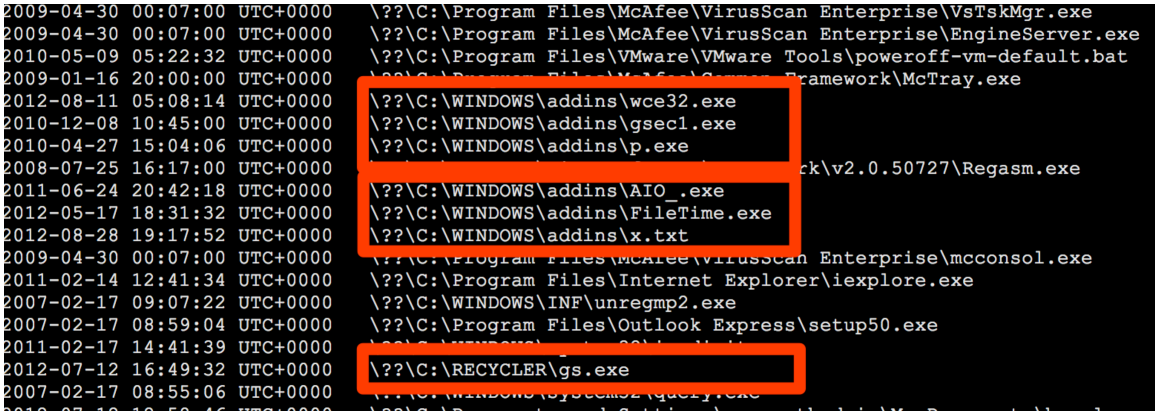



Figure 4: ShimCache results from memory analysis

Based on these findings RSA IR advised CompanyA that this server had evidence of unauthorized activity, and that based on some of the filenames of the tools, the adversary had most likely dumped password hashes. RSA IR advised CompanyA that based on the Last Modified times for the executed files, this adversary had been on this system since at least 2012 and possibly 2010. These findings indicated a high probability that other systems on the network were compromised and/or accessed.

The same LE agency notified the company again in September 2013 of more unauthorized activity.

## 4.2 Incident Response

On 26 September 2013 the RSA IR team was formally engaged to respond to this incident. The following section describes how RSA IR analysts were able to utilize SA and ECAT to investigate this incident. The ECAT agent was initially deployed to a small number of systems on the network primarily due to the victim's belief that this was an isolated incident. CompanyA had taken the first known compromised system offline in June, a few weeks before we performed memory analysis on it. RSA IR chose eight systems to perform host forensics on due to their importance to the victim organization.

### 4.2.1 ECAT Analysis – System XX13[2]

ECAT contains a set of filters and IOCs that highlight files of interest based purely on behavioral characteristics such as how they are loaded or where they are located. One of these filters is the "Reserved Filename", which displays any files that have reserved names and are not in their default location. The list of Reserved Filenames includes both common Windows file system names such as svchost.exe, explorer.exe, etc, as well as the names of common applications such as browser executables. In the example depicted in Figure 5 below, ECAT indicated that a file named svchost.exe (which natively resides under c:\Windows\system32\ folder) is suspicious due to its unexpected location.

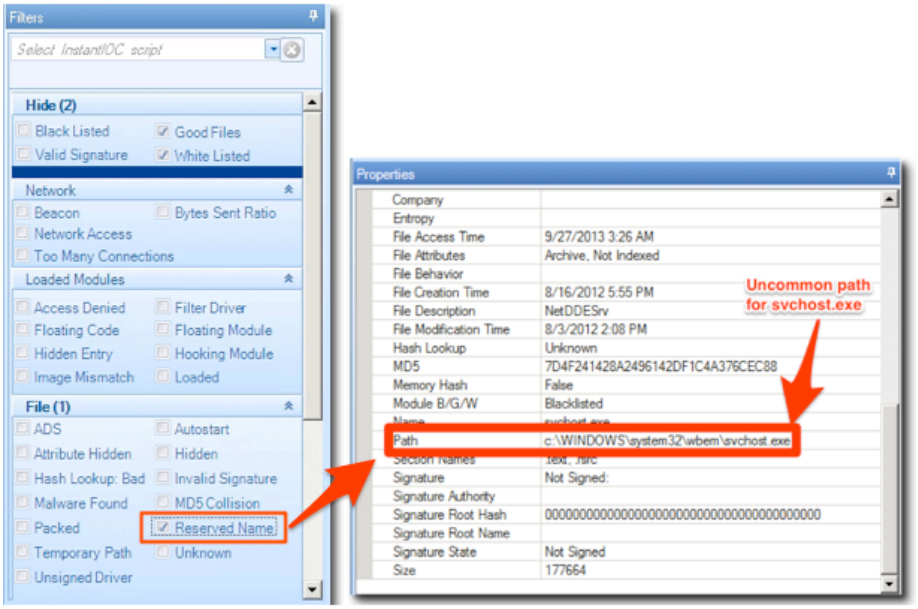



Figure 5: ECAT File Properties Window

Another very useful feature of ECAT is the ability to triage a system by downloading a system's Master File Table (MFT) directly from ECAT's console, swiftly allowing the ECAT analyst to triage a system remotely without interfering with the end user's usage of the system. The built-in ECAT MFT Viewer displays all relevant NTFS attributes including all 8 NTFS time stamps[3]. Frequently, modern APT Trojans time stomp their files to avoid suspicion, so seeing all 8 time stamps is critical in determining when something malicious occurred as well as finding other related events.

Within a few seconds after requesting the MFT the analyst was able to perform time analysis on the system. This process started with events that occurred around the time when the known malicious file named svchost.exe was created. This analysis revealed another related file named svchost.conf, which was determined to be this Trojan's obfuscated configuration file.

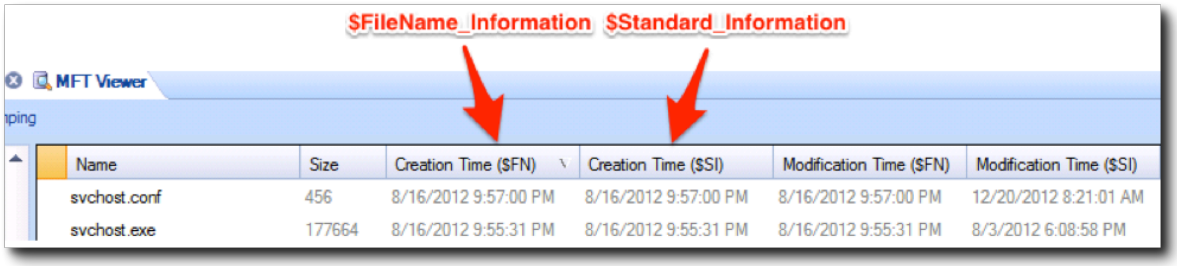



Figure 6: ECAT MFT Viewer - Trojan.Hikit

[2] Throughout this report some or all of the letters in the names of the systems have been obfuscated to protect the identity of our client.

[3] Four time stamps come from the $STANDARD_INFORMATION ($SI) attribute, whereas the other four come from the $FILENAME_INFORMATION ($FN) attribute

The configuration file (svchost.conf) is obfuscated with a 4 byte XOR key (0xFA274BCD) and contains C2 IP address **206.205.82.9**. These two malicious files are components of what RSA IR refers to as Trojan.Hikit:

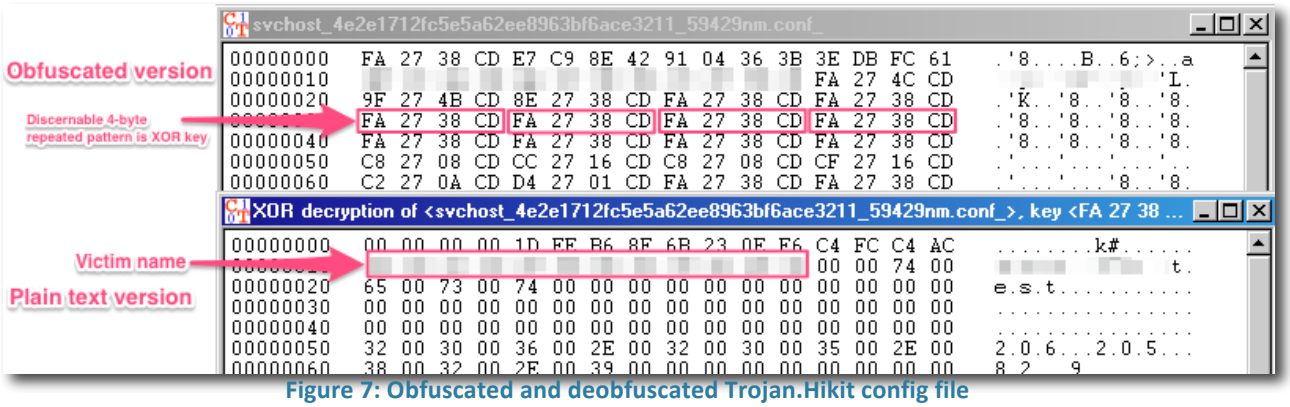



Figure 7: Obfuscated and deobfuscated Trojan.Hikit config file

From the created timestamp of svchost.conf, the analyst deduced that svchost.exe was executed via a scheduled job, and it dropped svchost.conf. Malware analysis on svchost.exe confirmed this behavior. Typically, when two or more files are created in such proximity in time to each other, and at least one of them has "00" for the seconds, this is indicative of this file being executed via a scheduled job. This is because when scheduling an "at" job the user only specifies an hour and minute, thus a job is executed as soon as the specified minute arrives, and the seconds end up being "00".

In this case it is highly likely that file svchost.exe was laterally copied over to this system, followed by remotely scheduling an "at" job to execute that file[4]. The job was executed about 1 minute later, and it resulted in the dropping of file svchost.conf. The fact that another local system was involved to infect system XX13 was important because it showed that at least one other system on the network was compromised. A quick check of the C:\Windows\Tasks folder did not show any leftover At.job files, whereas the entries on SchedLgU.txt file had already rolled into 2013. Furthermore, the Windows Security Event logs, which would typically contain logs on which account and from which system the lateral movement occurred, had also rolled. Lastly, this was a Windows XP system and so their event log Microsoft-Windows-TaskSchedules%4Operational.evtx did not exist, which is typically another great evidence source for lateral movement.

The analyst blacklisted file svchost.exe by its MD5 hash in ECAT so that if it were encountered again it would be marked as malicious. However, as it is very common with many Trojans deployed on a network, at least some of them will vary slightly from others and have a different hash value since at a minimum they are probably configured to beacon to different C2 channels or compiled at a different time. This is where another great feature of ECAT is very useful, namely ECAT's ability to ingest YARA signatures. This feature also helps immediately mark "suspicious" files as "malicious". So, it is common practice for RSA IR analysts to create Yara signatures for newly discovered malicious files. Figure 8 provides a signature for Trojan.Hikit:

```
rule Trojan_HIKIT
{
    meta:
        Author = "HB"
        Date   = "26 Sep 2013"
        Project = "Orion"
    strings:
        $b1 = {63006F006E006E006500630074002000250064002E00250064002E00250064002E00250064002000250064000000000680069006B00690074003E}
        $b2 = {6800690074007800 2E007300790073000000 6D006100740072006900780 05F007000610073007300730077006F007200}
        $b3 = {700072006F007800790000063006F006E006E006500630074000000660069006C006500000000007300680065006C006C}
        $a1 = "Open backdoor error" wide
        $a2 = "data send err.." wide
    condition:
        any of ($b*) or all of ($a*)
}
```

Figure 8: Yara signature for Trojan.Hikit

[4] While we have seen adversaries locally schedule a job to execute a file on the local system, this is not very common although worth keeping in mind.

### 4.2.2 ECAT Analysis – System XXDEV3

On a second host (XXDEV3) of the eight systems where ECAT was deployed, ECAT discovered a second instance of Trojan.Hikit. The Yara rule had already marked the file as malicious (i.e. blacklisted it).

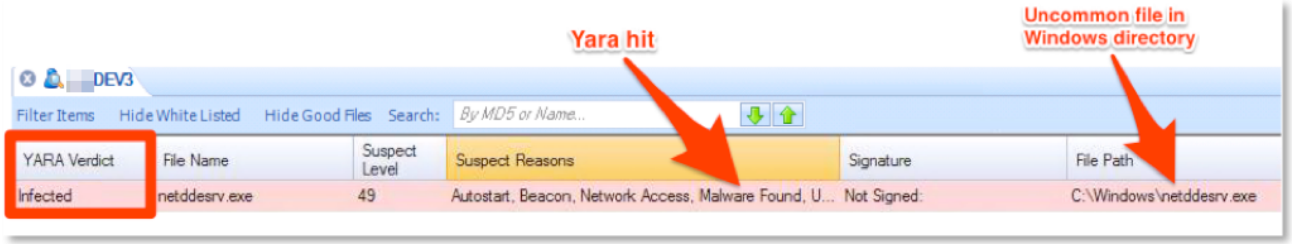



Figure 9: Yara hit on Trojan.Hikit

When the analyst triaged XXDEV3, two Trojan.Hikit configuration files were found. Furthermore, while this example of Trojan.Hitkit appeared to have been on the system since 2012, the two configuration files were created in 2013, as shown below:

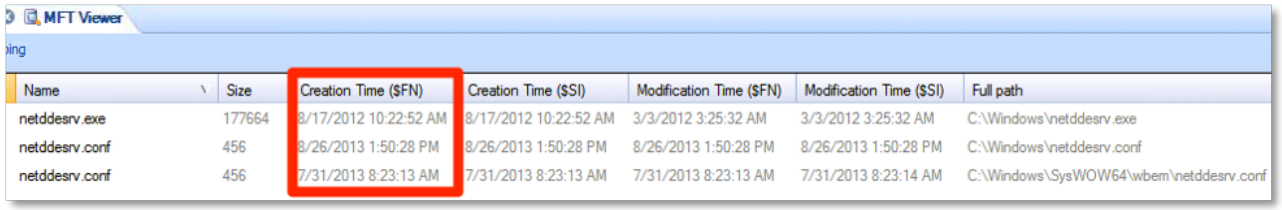



Figure 10: ECAT MFT time analysis - Trojan.Hikit

Both configuration files were obfuscated with a 4-byte XOR key similar to the example from XX13. The same C2 node was found in both configuration files: **drometic.suroot.com** (200.108.192.31). The analyst also found three "at" job files that fortunately weren't deleted after execution, as shown below:

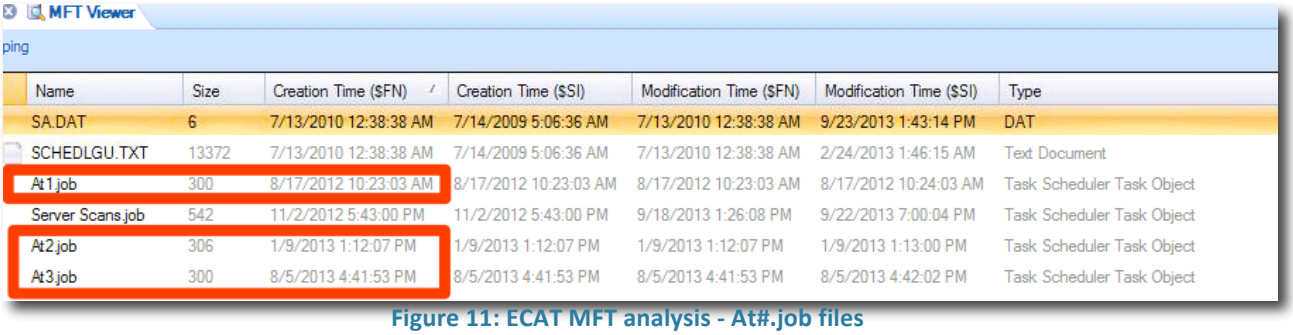



Figure 11: ECAT MFT analysis - At#.job files

At this point the analyst had three relevant timestamps to perform time analysis on. After sorting all the entries in the MFT based on the $FN timestamp, the analyst discovered the following relevant activity:

1. A few seconds after file netddesrv.exe was created on the system a job was scheduled (At1.job). This job file executed a file named **log.bat**, which was no longer present on the system, which executed[5] netddesrv.exe in return. This proved to be a classic example of lateral movement.

[5] This assumption is probably true because throughout this case we saw the adversary execute files via batch files such as in this instance.

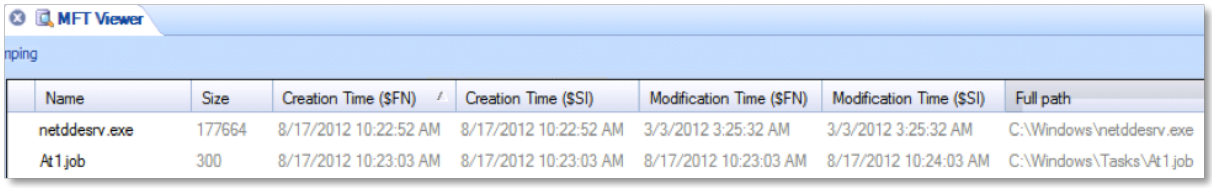



**Figure 12: ECAT MFT time analysis**

2. Looking at the next scheduled job the analyst discovered artifacts that were different from what had been encountered up to that point, namely two new files appeared after a scheduled job was executed.

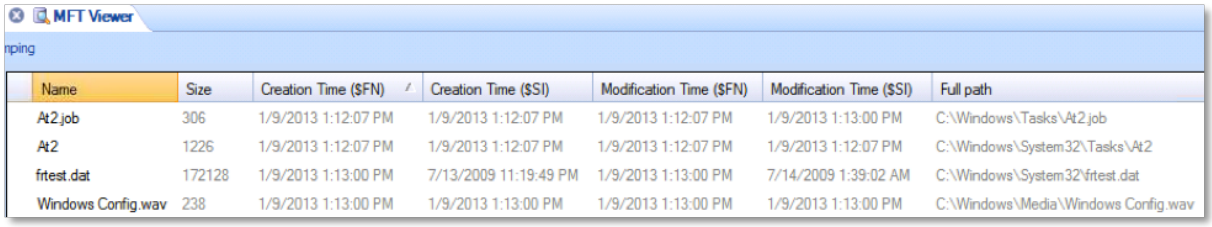



**Figure 13: ECAT MFT analysis - Trojan.FF-RAT**

The scheduled job At2.job executed c:\set.exe (no longer present on the system), which dropped files **frtest.dat** and **Windows Config.wav**[6]. The last two files were components of what RSA IR refers to as Trojan.FF-RAT. It was unclear at that point whether this Trojan was from a different APT group or if the same APT group that entrenched itself in this system in August 2012, decided to drop a second type of Trojan in this system. Another interesting finding about Trojan.FF-RAT was that file frtest.dat was legitimately digitally signed (as of the time of the engagement):

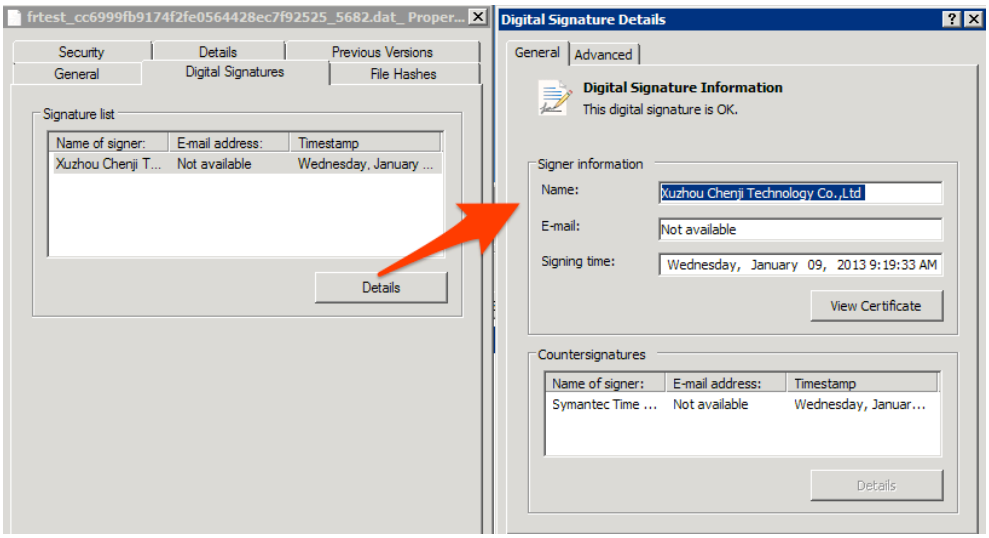



**Figure 14: Digital Signature details of Trojan.FF-RAT**

Digitally signed malware is rare, and implies a higher level of sophistication from an adversary. The file **Windows Config.wav** file was compressed and contained the Trojan's configuration information, including the C2 domain and its project name. Malware analysis showed that the configuration file of this Trojan contained two C2 domain names, which both resolved to the same IP at that time: **mno80.dwy.cc** and **mno995.dwy.cc** (198.55.120.222).

[6] Notice the "00" on the seconds for the creation time. If the At2.job had not existed or the SchedLgU.txt file does not contain any evidence of scheduled "at" jobs, you could infer that the Trojan was dropped via lateral movement, by looking the "00" seconds in the creation time of the malicious file(s).

RSA created a YARA signature for this new Trojan based on a unique decompression algorithm that the Trojan utilized:

```
rule Artifact_ORION_aPLib
{
    meta:
        Author = "HB"
        Date   = "30 Sep 2013"
        Project = "Orion"
    strings:
        $a1 = "aPLib v"
        $a2 = "the smaller the better :)"
        $a3 = "Joergen Ibsen"
    condition:
        all of them
}
```

Figure 15: Yara signature for Trojan.FF-RAT

3. The third job file "At3.job" also executed a file named **log.bat** (no longer present on the system), however there is nothing else relevant around this time:

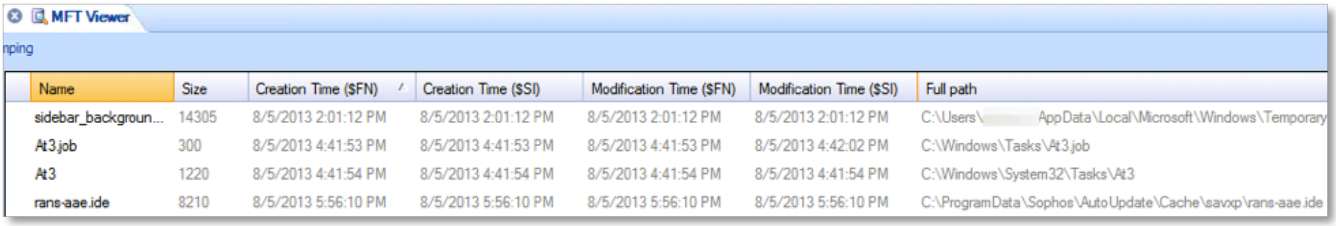



Figure 16: ECAT MFT time analysis on At3.job file

Since ECAT was only deployed to eight systems at this point, Security Analytics complimented this gap in endpoint visibility by providing network visibility. A quick lookup of IP address 198.55.120.222 shows that 14 internal systems are beaconing out to that IP address, and there are several other domain names also involved:

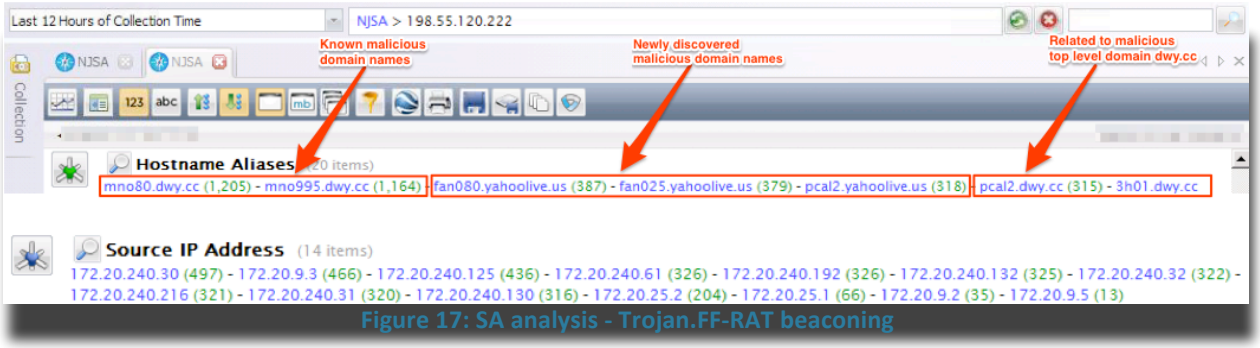



Figure 17: SA analysis - Trojan.FF-RAT beaconing

After presenting these findings to CompanyA, ECAT agents were deployed to every Windows system on the network. This is where ECAT also compliments SA; several of the discovered Trojans were set to sleep longer than others, were not actively running, or the system was previously infected with Trojan.FF-RAT however the Trojan executable files had since been removed.

ECAT uses frequency analysis to give the analyst instant visibility across the environment on any given file. In this case of the Yara signature for Trojan.FF-RAT, ECAT informed the analyst that file frtest.dat (by MD5 hash) also existed on 5 additional systems. ECAT also informed the analyst how this file was loaded/entrenched, namely via a service name **Nwsapagent.**

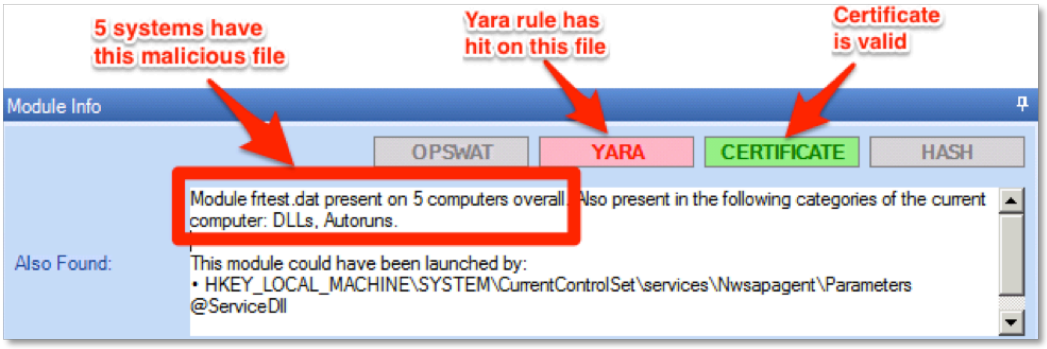



Figure 18: ECAT analysis - Trojan.FF-RAT

The 5 systems in question are shown below:

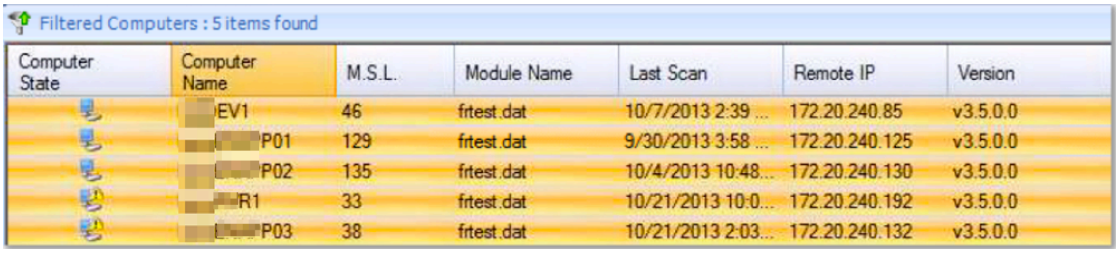



Figure 19: ECAT analysis - filtering infected hosts

### 4.2.3 ECAT Analysis – Trojan.FF-RAT

The analyst knew that Trojan.FF-RAT consisted of at least a DLL file (with a .dat extension) and a configuration file under: C:\Windows\Media**Windows Config.wav**, and the hash values of the DLLs and the configuration files varied from system to system. Since the configuration file (Windows Config.wav) was a unique filename that does not exist on a Windows system by default, and was always found in the same directory, the analyst used this fact to query all systems for evidence of Trojan.FF-RAT. This request would show all systems that were or had been infected with Trojan.FF-RAT, as well as account for systems where the Trojan was present on the system but not actively running. ECAT makes this request very easy:

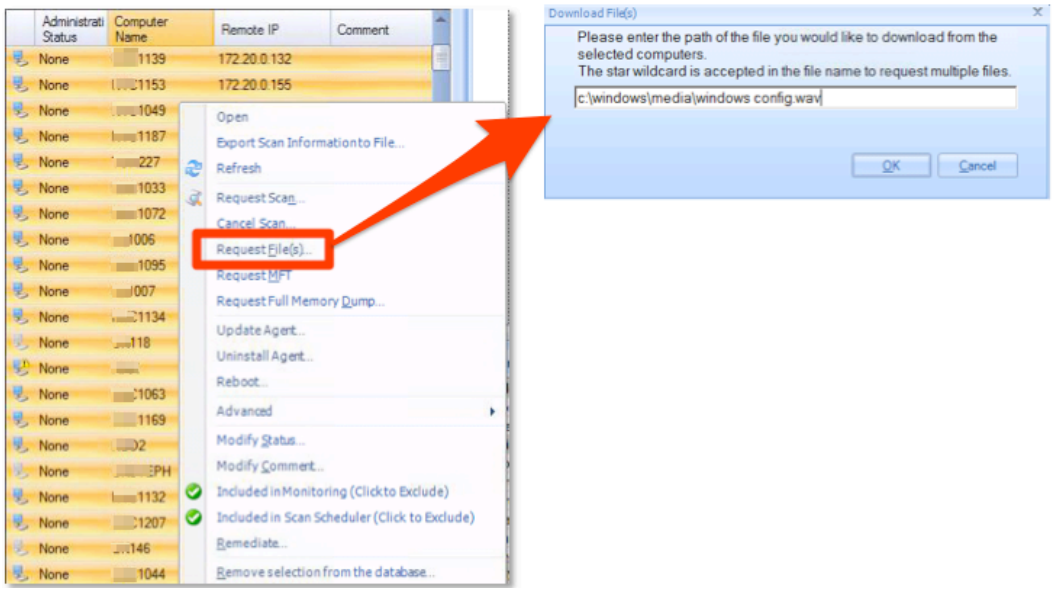



Figure 20: ECAT analysis - requesting files enterprise wide

Within a few seconds ECAT gave the analyst a list of systems that contained file C:\Windows\Media**Windows Config.wav**[7], which had been infected with Trojan.FF-RAT. A total of 31 systems contained a Trojan.FF-RAT configuration file, as shown below (some system names have been blurred to protect the name of the victim).

| Machine Name | Type | Message |
|---|---|---|
| AS4 | Completed, Download | Saved received file to: .\Files\W\windows config_b975d6cad0323adb3ead29a353d99de9_464nm.wav_ |
| BX9 | Completed, Download | Saved received file to: .\Files\W\windows config_1326ef179ad955226d7179e6078ffc30_44161nm.wav_ |
| BX8 | Completed, Download | Saved received file to: .\Files\W\windows config_59c5f25343a013fbc2f0f6331bc0f9f5_44146nm.wav_ |
| YNC1 | Completed, Download | Saved received file to: .\Files\W\windows config_99cf329a43475fb73ec4eb36c6e8c263_43178nm.wav_ |
| AS3 | Completed, Download | Saved received file to: .\Files\W\windows config_5b0f277e7375d93d33be3e482b2e899e_38727nm.wav_ |
| EMO | Completed, Download | Saved received file to: .\Files\W\windows config_7eff3346e4536468808e41e99b1cbc9e_36322nm.wav_ |
| | Completed, Download | Saved received file to: .\Files\W\windows config_b261687c13a0ec463f34a5fc143e1fcb_33541nm.wav_ |
| | Completed, Download | Saved received file to: .\Files\W\windows config_679f781775009db56ce3a5b06195267d_31419nm.wav_ |
| BX5 | Completed, Download | Saved received file to: .\Files\W\windows config_18ce7a31528e9439c21fd382e351004f_3191nm.wav_ |
| APP01 | Completed, Download | Saved received file to: .\Files\W\windows config_5f34af7b902811a3ea147f541d9260f2_30591nm.wav_ |
| 32 | Completed, Download | Saved received file to: .\Files\W\windows config_6ba872ea5606f41967edd7b6799387b7_30170nm.wav_ |
| AS6 | Completed, Download | Saved received file to: .\Files\W\windows config_4fd1ab5c97cf7c4104f3cbb926f16c23_30154nm.wav_ |
| AS5 | Completed, Download | Saved received file to: .\Files\W\windows config_5e84cca88aded1281e1c916379a7b7a3_27201nm.wav_ |
| BX4 | Completed, Download | Saved received file to: .\Files\W\windows config_e18baf8934876e5a75a42a635e9a87d8_25748nm.wav_ |
| | Completed, Download | Saved received file to: .\Files\W\windows config_2a8e45e1163473ef8e7fd71a6b542c43_25215nm.wav_ |
| 1 | Completed, Download | Saved received file to: .\Files\W\windows config_936bf93721588523bf30d647ec35addb_24576nm.wav_ |
| BBIMSS | Completed, Download | Saved received file to: .\Files\W\windows config_0f304e9cc4660e3f86e61430b5ce15d4_24513nm.wav_ |
| BX7 | Completed, Download | Saved received file to: .\Files\W\windows config_f6b1af17a6ccb0e8de904487079bc075_24185nm.wav_ |
| PORT | Completed, Download | Saved received file to: .\Files\W\windows config_7a033c5fd033b46327c49b29aba1cde7_23779nm.wav_ |
| ACKUP | Completed, Download | Saved received file to: .\Files\W\windows config_dcacd283b24fac04b27cee13c0c52463_23122nm.wav_ |
| RVR1 | Completed, Download | Saved received file to: .\Files\W\windows config_bf617827060ce2852a328e83717ce3f1_18282nm.wav_ |
| NAME | Completed, Download | Saved received file to: .\Files\W\windows config_69349f7d7724b78ab0a0c445fe5568b7_14721nm.wav_ |
| 1026 | Completed, Download | Saved received file to: .\Files\W\windows config_8d556b44d94cb1284f5bdc4bc67606a8_14158nm.wav_ |
| NAPP01 | Completed, Download | Saved received file to: .\Files\W\windows config_6fe3b357a4dd73475799237015827347_1496nm.wav_ |
| ES1 | Completed, Download | Saved received file to: .\Files\W\windows config_503197ce5fd48d14f08cee005c563816_13550nm.wav_ |
| DEV1 | Completed, Download | Saved received file to: .\Files\W\windows config_1de4d3afc365c84460f4f3c3182a3a66_9990nm.wav_ |
| DEV3 | Completed, Download | Saved received file to: .\Files\W\windows config_2421463acff2fcecbf69664f268a0c45_7289nm.wav_ |
| 23 | Completed, Download | Saved received file to: .\Files\W\windows config_44c41be46341a0de78a35707cc51eebb_59905nm.wav_ |
| 1164 | Completed, Download | Saved received file to: .\Files\W\windows config_b9ae3659a268e5c8b3897f6e4634ff67_58591nm.wav_ |
| NAPP02 | Completed, Download | Saved received file to: .\Files\W\windows config_3b98e1e8f83ec61adf23e034eafcb3c1_57233nm.wav_ |
| NAPP03 | Completed, Download | Saved received file to: .\Files\W\windows config_8bb0662b176649e68eca16ec351cf2ad_54392nm.wav_ |

Figure 21: ECAT analysis - systems infected with Trojan.FF-RAT

### 4.2.4 ECAT Analysis – System XXXXNAPP02

The analyst triaged system XXXXAPP02 next, and focused on the C:\Windows\system32\ folder where frtest.dat (Trojan.FF-RAT) was located. The analyst noticed several additional suspicious files in this folder, which were considered suspicious for the following reasons:

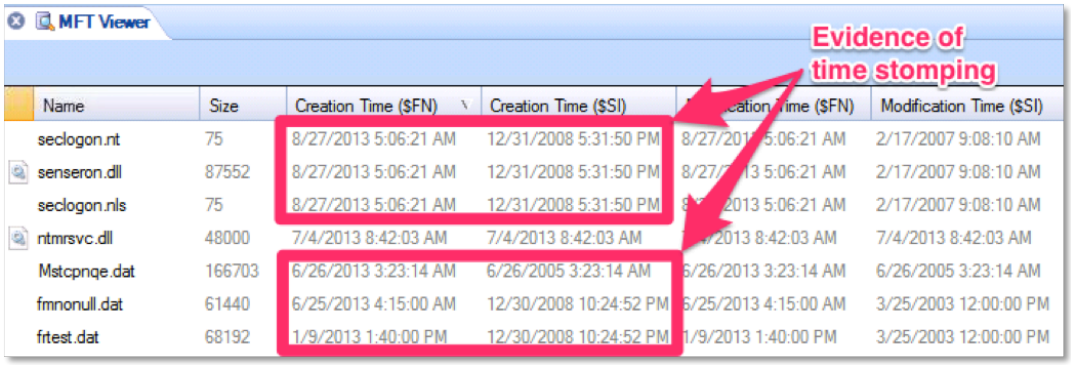



Figure 22: ECAT MFT time analysis - Time stomping

[7] A detailed technical description of how these configuration files were obfuscated can be found on Appendix I.

1. File **frtest.dat** was already known to be malicious.

2. Files **fmnonull.dat** (Trojan.FF-RAT) and **Mstcpnqe.dat** (Trojan.Derusbi) looked suspicious for the following reasons:
    a. There are only a handful of .dat files in the system32 folder by default, and their timestamps including the $FN are much older (i.e. during the OS installation time). These two files were time stomped to look older (compare $FN Creation time vs $SI Creation time) but in fact were created on 25 June and 26 June 2013 respectively.
    b. Both files were created (looking at $FN timestamp) very early in the morning (a substantial amount of malicious activity in this case occurred between 12:00AM and 6:00AM EST).

3. File **ntmrsvc.dll** (Trojan.Lurker2) may look like a legitimate filename (in fact it is only one character different from the legitimate filename ntmssvc.dll). The most suspicious aspect of this file was that it was created recently with no other activity around it in the system32 folder. When looking at all the files in the system and sorting by $FN, another suspicious file in the C:\Windows\Temp folder was identified.

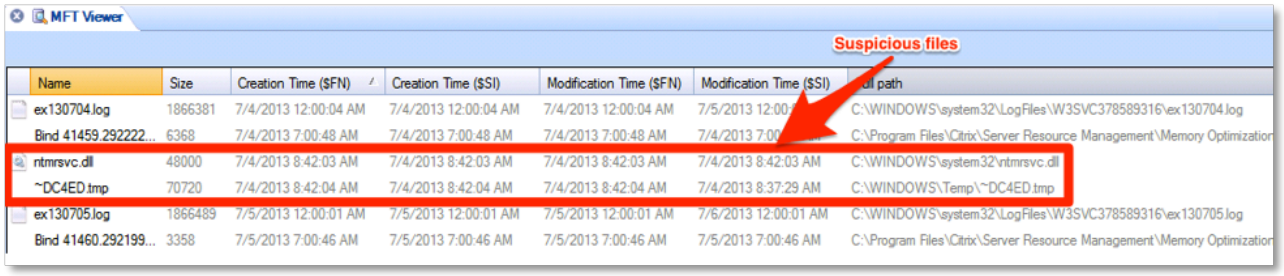



Figure 23: ECAT analysis - Trojan.Lurker2

4. Files **seclogon.nls**, **senseron.dll**, and **seclogon.nt** are suspicious for the following reasons:
    a. Time stomped.
    b. Created during the early hours in the morning.

On this system the analyst discovered two new Trojan families Trojan.Derusbi and Trojan.Lurker2. The analyst created Yara signatures for each of them.

```
rule Trojan_Lurker2_ORION
{
    meta:
        Author = "HB"
        Date   = "30 Sep 2013"
        Project = "Orion"
    strings:
        $b1 = {636D642E657865004C55524B}
        $b2 = {45525F52414353004C55524B25735F534D5F2573}
        $b3 = {4C55524B45525F524143535F524D5F2573}
        $a1 = "0123456789012345678 9eric0123456789012345678karen"
    condition:
        any of them
}
```

Figure 24: Yara Signature - Trojan.Lurker2

```
rule Trojan_Derusbi_AP32_Orion
{
    meta:
        Author = "HB"
        Date   = "30 Sep 2013"
        Project = "Orion"
    strings:
        $http1  = {00000000485454502F312E312032303000000000485454502F312E3020323030}
        $http2  = {00000000434F4E4E4543542025733A256420485454502F312E300D0A0D0A0000}
        $file1  = "%s\\seclogon.nls"
        $file2  = "%s\\seclogon.nt"
        $file3  = "%swindows.exe"
        $o1     = "\\wsedrf\\qazxsw"
        $o2     = "\\shell\\open\\command"
        $b1     = {4C4F47494E494E464F3A2025640A0000}
        $b2     = {436F6465506167653A2025730A000000}
        $b3     = {5C636D642E657865}
    condition:
        all of ($http*) or all of ($file*) or all of ($o*) or all of ($b*)
}
```

Figure 25: Yara signature - Trojan.DerusbiAP32

### 4.2.5 ECAT Analysis – Recycler folder

At this stage of the investigation the analyst had discovered four different Trojan families and several infected hosts. Very often APT adversaries are not careful enough to cleanup after themselves, and relevant artifacts can be found by exploiting the tendencies of the adversary. For example, when RSA performed memory analysis on the early stages of this case, it discovered (via ShimCache analysis) that the adversary had a preference for storing relevant files under C:\Recycler\ and C:\Windows\addins\.

Since certain directories do not typically contain certain types of files, the analyst used ECAT to query these directories for files that are out of place. For example, the root of the Recycler folder should not contain any files at all, so the analyst requested that ECAT download **C:\Recycler*.*.** Here are the files that were retrieved from the C:\Recycler\ directory from various systems:

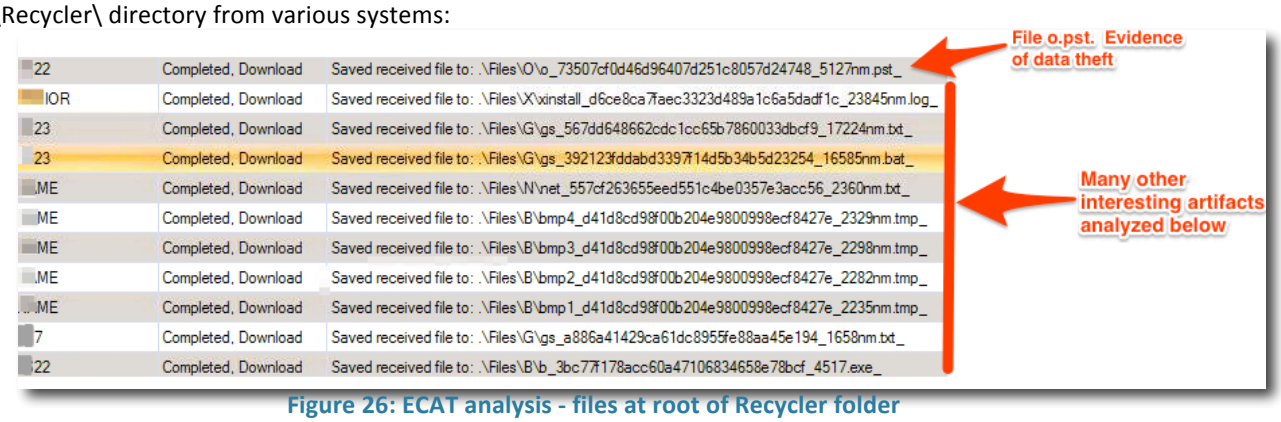



Figure 26: ECAT analysis - files at root of Recycler folder

### 4.2.6 ECAT Analysis – System XXME

The triage process of system named XXME led to the discovery of some interesting artifacts. The first relevant fact was that file **C:\Recycler\net.txt** was created in 2010. This event confirms initial suspicions from the earliest stages of this case, that the earliest evidence of unauthorized activity goes back to 2010:

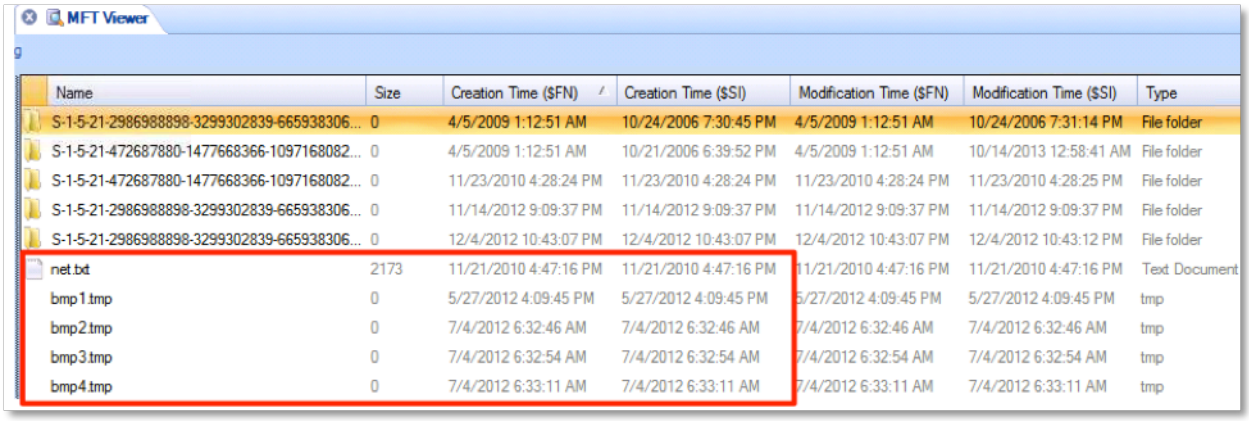



**Figure 27: ECAT analysis - files at root of Recycler folder**

Time analysis did not show anything else relevant from 2010. When the analyst performed time analysis around the **bmp#.tmp** files the following relevant activity was discovered:

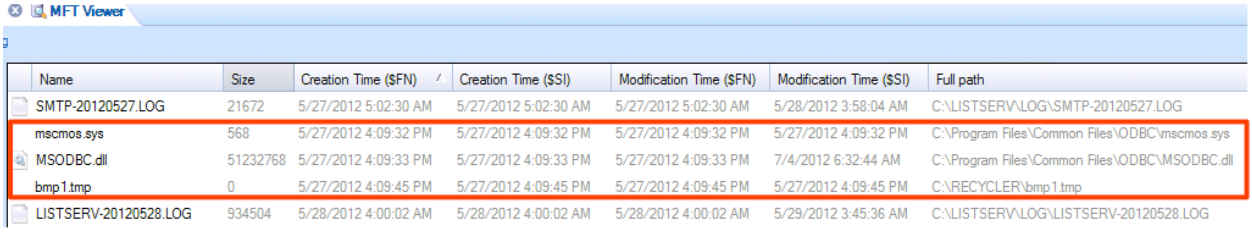



**Figure 28: ECAT analysis - Trojan.Gh0st**

Despite the unusual size of file MSODBC.dll at about 50MB[8], this file along with file mscmos.sys were found to be components of Trojan.Gh0st. The adversary artificially increased the size of file MSODBC.dll by appending junk data to the end of it, presumably to avoid suspicion. A quick look at mscmos.sys reveals that it was an obfuscated (XOR with 0x99) configuration file that contained domain name **ru.pad62.com,** and the victim's company name appeared at the beginning of the file.

When looking at the July 4th 2012 activity the analyst observed another interesting file:

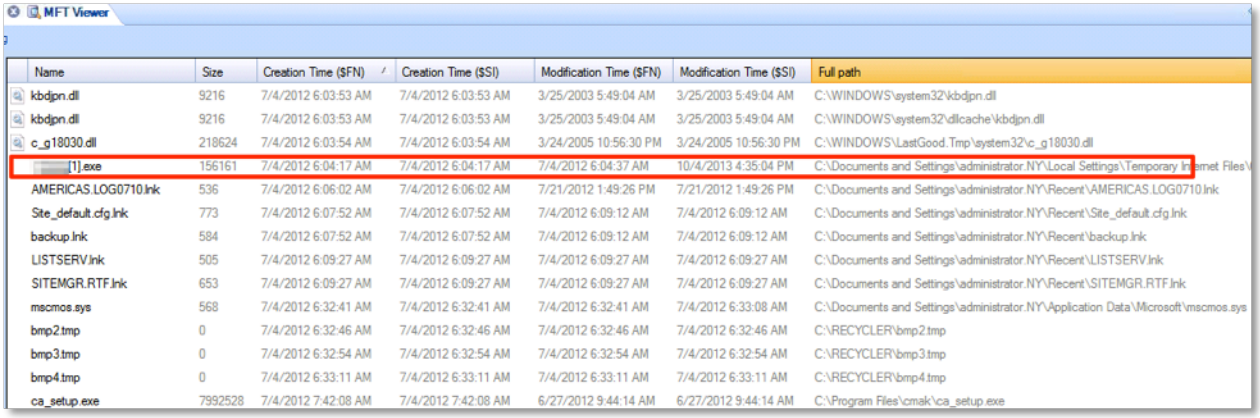



**Figure 29: ECAT analysis - Trojan in cached files**

Due to the time proximity to the **bmp#.tmp** file, the analyst noticed that an executable file was cached under the administrator.NY account. This file was also found to be highly suspicious because its filename contained the

[8] The size of a typical Trojan is less than 1MB; with a large percentage of Trojan sizes falling between (10KB – 350KB)

company name in it (blurred for this reason). The fact that this file is cached implies that the adversary had GUI access to this system and used Internet Explorer to download this file. Whenever relevant cached files are discovered on a system, the analyst investigates the Internet History of that user's profile to determine from where this file was downloaded. Internet history showed that the malicious file was downloaded from: **www.haircollalife.com**.

| FileName | SourceURL | FullHttpHeader | LastModifiedDate | BytesDownloa | MaxBytes |
|---|---|---|---|---|---|
| 1].exe | https://www.haircollalife.com/img/category/im-temp.php82a678a1 | HTTP/1.1 200... | 2012-07-04T06:04:45Z | 333782 | 333782 |

**Figure 30: Relevant Internet History results**

The file in question also appeared to have been digitally signed; however this certificate was no longer valid. This downloaded file was found to be a dropper for Trojan.Gh0st. When executed it drops three files: 6to4adv.dll, BusMgr.sys, and SvcPack.dat.

Digging through the rest of the Internet history, another suspicious file was downloaded on 26 August 2013 named x8.txt from **http://198.27.112.7:666**. The downloaded file was also a legitimately digitally signed executable file (i.e. another Trojan.FF-RAT variant).

| FileName | SourceURL | FullHttpHeader | LastAccessedDate | LastModifiedDate | BytesDownloa | MaxBytes |
|---|---|---|---|---|---|---|
| x8[1].txt | http://198.27.112.7:666/x8 | HTTP/1.1 200... | 2013-08-26T19:15:07Z | 2013-08-26T19:11:08Z | 127592 | 127592 |

**Figure 31: Additional relevant Internet History results**

Evidence found in this system shows that the APT group that is using digitally signed Trojans deployed a Trojan.Gh0St variant (MSODBC.dll) in May 2012. It then deployed a digitally signed Trojan.Gh0st variant in July 2012. Internet research on the digitally signed Trojan.Gh0st showed that Sophos started identifying this variant as Troj/PWS-BYU on 17 July 2012[9]. CompanyA runs Sophos AV in their environment, so the signed Trojan.Gh0st only worked for about 13 days before it was quarantined. The unsigned version of Trojan.Gh0st was also no longer active because malware analysis showed that MSODBC.dll attempts to load a file named JET.dll, which was no longer present on any system. So, here we are dealing with the remnants of a Trojan that is no longer active in this network. The analyst used ECAT to find all systems that still contained Trojan.Gh0st artifacts, namely the presence of file **svcpack.dat**, and found several such systems:

| | | |
|---|---|---|
| IOR | Completed, Download | Saved received file to: .\Files\S\svcpack_6466ac2b6070e9161a937e1d0bebc698_11771nm.dat_ |
| DEV1 | Completed, Download | Saved received file to: .\Files\S\svcpack_80431d60ceca87336536f66c740fb77_3278nm.dat_ |
| 2 | Completed, Download | Saved received file to: .\Files\S\svcpack_d318f3748c377919cd37eb8f00e2446e_0452nm.dat_ |
| CKUP | Completed, Download | Saved received file to: .\Files\S\svcpack_cab1a38e462c082bc060fb42fe237ff8_58533nm.dat_ |
| 22 | Completed, Download | Saved received file to: .\Files\S\svcpack_b4dbacc42fcfedea1c458f4b80a982b5_57548nm.dat_ |
| | Completed, Download | Saved received file to: .\Files\S\svcpack_bf673e55640e084e6eb877d61321a2db_55408nm.dat_ |
| 1 | Completed, Download | Saved received file to: .\Files\S\svcpack_7ea39738bfa1f3639b06cbd21d20670c_52738nm.dat_ |
| 7 | Completed, Download | Saved received file to: .\Files\S\svcpack_a452d945c6631b92e85cd3cc4c8ce389_42635nm.dat_ |

**Figure 32: ECAT analysis - Systems infected with Trojan.Gh0st**

File svcpack.dat contains Trojan configuration information at the beginning of the file, which is obfuscated via a single byte XOR (0x11). Here is an example of a de-obfuscated configuration data from a SvcPack.dat file:

[9] https://secure2.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~PWS-BYU/detailed-analysis.aspx

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 72 | 75 | 2E | 70 | 61 | 64 | 36 | 32 | 2E | 63 | 6F | 6D | 00 | 00 | 00 | 00 | ru.pad62.com |
| 00000010 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 72 | r |
| 00000020 | 75 | 2E | 70 | 61 | 64 | 36 | 32 | 2E | 63 | 6F | 6D | 00 | 00 | 00 | 00 | 00 | u.pad62.com |
| 00000030 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 50 | 00 | P |
| 00000040 | BB | 01 | 30 | 2E | 30 | 2E | 30 | 2E | 30 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | » 0.0.0.0 |
| 00000050 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000060 | 00 | 50 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | P |
| 00000070 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000080 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000090 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 000000A0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 32 | 30 | 31 | 32 | 30 | 33 | 32 | 2012032 |
| 000000B0 | 33 | 31 | 36 | 31 | 36 | 31 | 36 | 41 | 42 | 43 | 44 | 45 | 46 | 00 | [redacted] | | 3161616ABCDEF |
| 000000C0 | [redacted] | | | | | | | | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 000000D0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 000000E0 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 1C | 0C | 00 | 00 | 00 | 00 | 00 | 00 | |

Company name

Figure 33: Trojan.Gh0st - de-obfuscated configuration file

### 4.2.7 ECAT Analysis – System XX22

The analyst started to triage system XX22 by performing time analysis around the files discovered in the C:\Recycler folder of this system. MFT analysis also revealed some deleted text files that used to exist in this folder. These text files were recovered and were found to contain recursive directory listings of drives of other systems in the network. Typically APT adversaries get recursive directory listings to determine which filenames look interesting for exfiltration.

MFT Viewer

| Name | Size | Creation Time ($FN) | Creation Time ($SI) | Modification Time ($FN) | Modification Time ($SI) | Type |
|---|---|---|---|---|---|---|
| S-1-5-21-16415116... | 0 | 10/8/2013 12:06:59 PM | 10/8/2013 12:06:59 PM | 10/8/2013 12:06:59 PM | 10/10/2013 10:09:25 AM | File folder |
| S-1-5-21-16415116... | 0 | 6/9/2011 9:32:38 PM | 6/9/2011 9:32:38 PM | 6/9/2011 9:32:38 PM | 6/9/2011 9:32:47 PM | File folder |
| S-1-5-21-16415116... | 0 | 6/2/2011 1:52:44 AM | 6/2/2011 1:52:44 AM | 6/2/2011 1:52:44 AM | 6/2/2011 1:52:44 AM | File folder |
| 11.txt | 4758392 | 10/11/2013 4:35:47 AM | 10/11/2013 4:35:47 AM | 10/11/2013 4:35:47 AM | 10/11/2013 4:33:21 AM | Text Document |
| b.exe | 366734 | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:37 AM | Application |
| 10.txt | 7874623 | 10/11/2013 4:21:26 AM | 10/11/2013 4:21:26 AM | 10/11/2013 4:21:26 AM | 10/11/2013 4:16:20 AM | Text Document |
| 7d.txt | 11588166 | 10/11/2013 3:58:54 AM | 10/11/2013 3:58:54 AM | 10/11/2013 3:57:14 AM | 10/11/2013 3:57:14 AM | Text Document |
| o.pst | 1920051... | 10/14/2013 2:19:13 PM | 10/14/2013 2:19:13 PM | 10/14/2013 2:19:13 PM | 10/11/2013 10:26:51 AM | pst |

Figure 34: ECAT analysis - files at root of Recycler folder

Looking at the activity around file b.exe the analyst discovered that it was responsible for dropping files **sbiedll.dll** and **helper.url** as shown below:

MFT Viewer

| Name | Size | Creation Time ($FN) | Creation Time ($SI) | Modification Time ($FN) | Modification Time ($SI) | Full path |
|---|---|---|---|---|---|---|
| b.exe | 366734 | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:29 AM | 6/9/2013 6:57:37 AM | C:\RECYCLER\b.exe |
| sbiedll.dll | 181760 | 6/9/2013 6:57:42 AM | 6/9/2013 6:57:42 AM | 6/9/2013 6:57:42 AM | 5/18/2013 12:38:58 AM | C:\WINDOWS\Temp\RarSFX0\sbiedll.dll |
| helper.url | 113874 | 6/9/2013 6:57:43 AM | 6/9/2013 6:57:43 AM | 6/9/2013 6:57:43 AM | 4/28/2012 9:02:10 PM | C:\WINDOWS\Temp\RarSFX0\helper.url |

Figure 35: ECAT MFT time analysis - Trojan.PlugX

These three malicious files are components of what RSA IR refers to as Trojan.PlugX. While looking at the c:\windows\system32 folder the analyst noticed a highly suspicious file named **svchost**. This file was very suspicious

because it contains the name of svchost.exe, which is a critical Windows file, but it has no extension. When the analyst pivoted on file svchost the following relevant files were discovered.

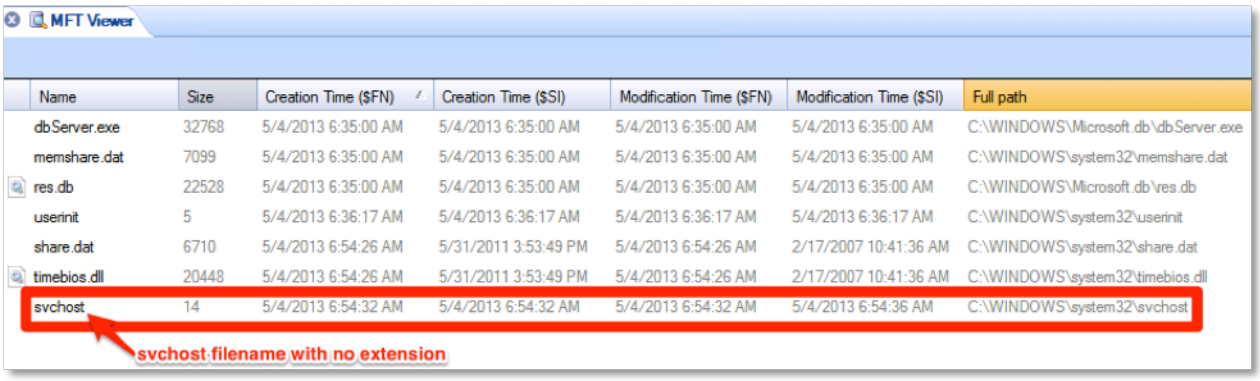



Figure 36: ECAT MFT time analysis - Trojan.PoisonIvy

Some quick malware analysis on dbServer.exe revealed that it was a variant of Trojan.PoisonIvy and it consisted of file dbServer.exe, which de-obfuscated and loaded res.db, which then loaded memshare.dat. File svchost is the keystroke logger file of Trojan.PoisonIvy. File timebios.dll and share.dat were also found to be components of PoisonIvy. The PoisonIvy password that was configured in these samples was: **15911117665**

ECAT identified 6 systems that contain file timebios.dll:

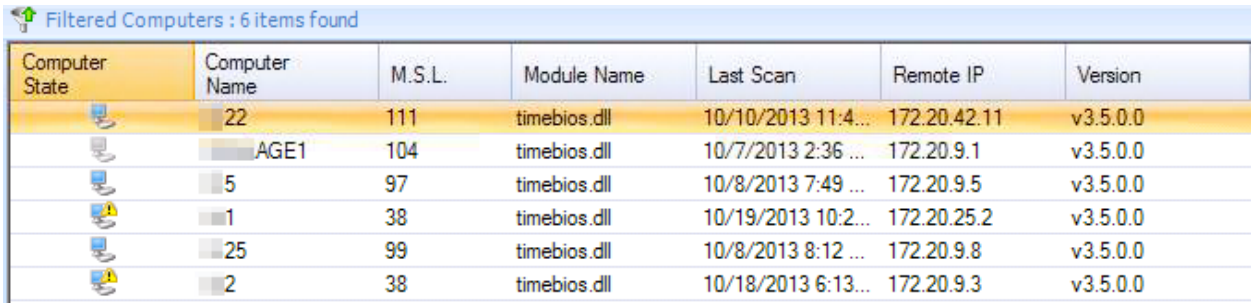



Figure 37: ECAT analysis - filtering systems infected with PoisonIvy

Four of these six systems also contained file dbServer.exe:

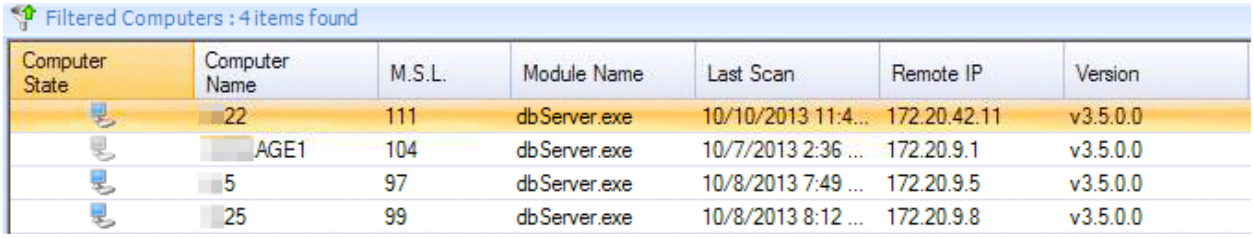
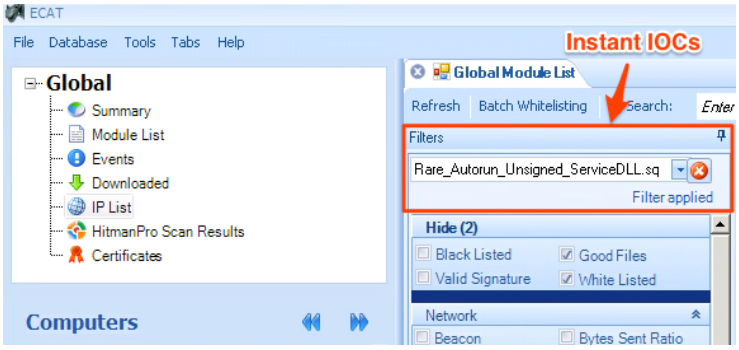



Figure 38: ECAT analysis - filtering systems infected with PoisonIvy

### 4.2.8 ECAT Analysis – Hunting with InstantIOCs.

Another ECAT filter that is a good APT malware identifier is the "Unsigned_ServiceDLL" filter. This InstantIOC filter will point out DLL files that are loaded as a service, but which are not signed. When running this filter against the GlobalModule List, ECAT points two other types of Trojans that RSA refers to as Trojan.Lurker and Trojan.Superhardcore.




ECAT pointed out four DLL files (tpoaed.dll, powms.dll, lpest.dll, asesed.dll) that are all variants of Trojan.Lurker:

Filtered Computers : 10 items found

| Computer State | Computer Name | M.S.L. | Module Name | Last Scan | Remote IP | Version |
|---|---|---|---|---|---|---|
| | 209 | 52 | tpoaed.dll | 9/27/2013 3:25 ... | 172.20.25.209 | v3.5.0.0 |
| | 1112 | 51 | tpoaed.dll | 9/27/2013 6:14 ... | 172.20.0.100 | v3.5.0.0 |
| | 1160 | 51 | tpoaed.dll | 9/27/2013 5:23 ... | 172.20.0.192 | v3.5.0.0 |
| | 1172 | 49 | powms.dll | 9/27/2013 5:04 ... | 172.20.0.223 | v3.5.0.0 |
| | 1199 | 73 | lpest.dll | 9/27/2013 6:10 ... | 172.20.0.233 | v3.5.0.0 |
| | 203 | 56 | lpest.dll | 9/27/2013 3:27 ... | 172.20.25.203 | v3.5.0.0 |
| | 1175 | 51 | lpest.dll | 9/27/2013 5:26 ... | 172.20.0.232 | v3.5.0.0 |
| | 1135 | 67 | asesed.dll | 9/27/2013 6:16 ... | 172.20.0.194 | v3.5.0.0 |
| | 1158 | 75 | asesed.dll | 9/27/2013 5:41 ... | 172.20.0.15 | v3.5.0.0 |
| | 1142 | 51 | asesed.dll | 9/27/2013 4:56 ... | 172.20.0.184 | v3.5.0.0 |

Figure 39: ECAT analysis - systems infected with Trojan.Lurker

Also, ECAT pointed out a file named AppMgmt32.dll, which is also an unsigned DLL that was loaded as a service named IRMON. RSA refers to this Trojan as Trojan.Superhardcore. Overall, the following systems contained Trojan.Superhardcore:

Filtered Computers : 3 items found

| Computer State | Computer Name | M.S.L. | Module Name | Last Scan | Remote IP | Version |
|---|---|---|---|---|---|---|
| | 1119 | 53 | AppMgmt32.dll | 9/27/2013 5:27 ... | 172.20.0.197 | v3.5.0.0 |
| | 1166 | 49 | AppMgmt32.dll | 9/27/2013 4:16 ... | 172.20.0.216 | v3.5.0.0 |
| | 1027 | 103 | AppMgmt32.dll | 9/27/2013 4:20 ... | 172.20.0.56 | v3.5.0.0 |

Figure 40: ECAT analysis - systems infected with Trojan.Superhardcorp

## 4.3 SA Analysis – Trojan.Lurker

One of the approaches that the RSA IR team uses to identify malicious network traffic relies on identifying anomalies in network protocols. Most Trojans do not follow protocol standards and thus can be detected based on these anomalies. For example, Trojan.Lurker is a classic HTTP Trojan. Detection depends on knowledge of the HTTP protocol and detecting anomalies and non-standard traffic. In this case the Trojan stood out for several reasons:

1. First, the Trojan had to use the POST method to send data to the server. There is no RFC mandated maximum HTTP Header size, although the default limits on Apache are 8KB and on IIS, 16KB. Trojans generally attempt to follow standards to allow the requests to be handled by proxies, if they exist in the environment. For that reason, most HTTP Trojans utilize the GET and POST method to facilitate a command shell that appears to be standard HTTP traffic. The POST Method traffic in most environments is generally 10% of overall HTTP traffic, making it easier for the analyst to find malicious sessions.
2. Second, the Trojan uses a short MSIE 7 User-Agent that identifies itself as Windows XP 64 bit Operating System.

3. Third, the Trojan is using only 3 HTTP headers, a very low amount, and doesn't follow best practices for HTTP/1.1; the Content-Type header is not present for the POST method.


Figure 41: SA analysis - Trojan.Lurker

We now highlight these items in the actual payload of the traffic:

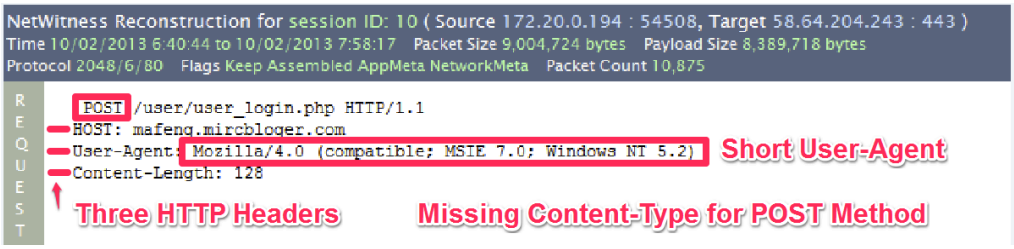



Figure 42: SA analysis - Trojan.Lurker HTTP anomalies

RSA discovered HTTP sessions to these IP's containing both Windows PE's as well as encrypted header RAR's. The command shell was encrypted, but files sent via the C2 channel were in the clear. The first PE to be sent was rar.exe.


Figure 43: SA analysis - Trojan.Lurker C2 channel activity

Next, a RAR archive with additional tools was sent to the infected host.

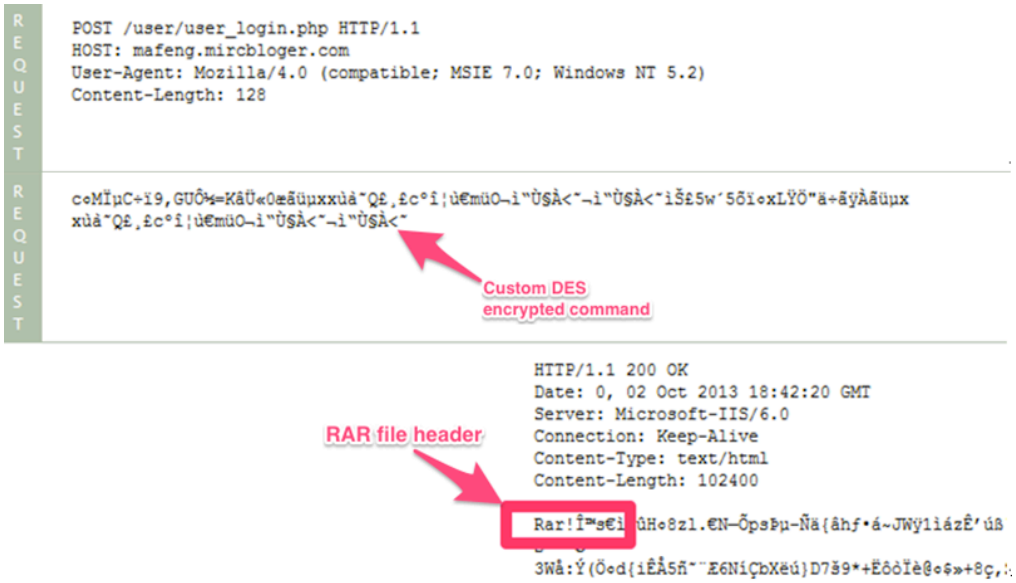



Figure 44: SA analysis - Trojan.Lurker C2 channel activity

A batch script was then uploaded and executed by subsequent commands. As can be seen from the previous to examples as well, this Trojan download files in plaintext.


Figure 45: SA analysis - Trojan.Lurker C2 activity

RSA IR decrypted the commands from the Trojan.Lurker traffic and identified the following relevant activity:

1. RS cmd.exe
2. net use \\XX16\ipc$ "*password*" /u:LOCAL*username*
3. UL C:\Windows\IME\IMEJP\ntfre.exe 332800
4. UL C:\Windows\IME\IMEJP\~WRD0208.tmp 134052
5. UL C:\Windows\IME\IMEJP\~WRD0219.tmp 183148
6. UL C:\Windows\IME\IMEJP\p8.bat 291
7. ntfre e -p"**&uej2&2^@!Ejd3wUDHFsw21"** ~WRD0219.tmp
8. r local*user1*::XXXXXX8F348F93FAD30C70304DXXXXXX:XXXXXX9F20421885C88B11C388XXXXXX::: "m -s:172.20.240.21 -u:*user1* -t:2013-10-15-00 -o:c:\windows\ime\imejp\mail"
9. r local*user2*::XXXXXX8F348F93FAD30C70304DXXXXXX:XXXXXXCCA445FCCD44E6BD66D8XXXXXX::: "m -s:172.20.240.21 -u:*user2* -t:2013-09-15-00 -o:c:\windows\ime\imejp\mail"

```
10. r local\user3::XXXXXX8F348F93FAD30C70304DXXXXXX:XXXXXX64AB0C641B0FC741B8C2XXXXXX::: "m
    -s:172.20.240.21 -u:user3 -t:2013-09-15-00 -o:c:\windows\ime\imejp\mail"
11. p8.bat
12. rd mail /s/q
13. del m.exe
14. del r.exe
```

The APT operator starts this session (1) with the infected system by launching a remote shell (RS). The operator then establishes a network connection (2) using stolen credentials. Several files are then uploaded (UL) to the infected system (3, 4, 5, 6). It should be noted that ntfre.exe is the RAR command line utility, whereas the .tmp files are actually RAR archive files. A very strong RAR password is used to extract (7) the contents of ~WRD0219.tmp RAR archive file. The operator then runs r.exe which is a pass-the-hash tool (8, 9, 10) on three different users, and also executes m.exe, which is an email harvesting tool. The email harvesting tool is passed arguments to only grab a delta of emails, i.e. the actors already have taken all previous emails, and are now only interested on the latest emails. The batch file p8.bat (11) is executed. This file contains the commands shown on Figure 46. The batch file does a couple of things: it extracts the files in archive ~WRD0208.tmp using RAR password: **64740629**. This archive file contained the password hash dump utility (PW.exe), which is executed against server XX16, hence the reason for establishing a network connection (2) to that system. The RAR utility (ntfre.exe) is then used to archive the collected emails. The content of the RAR archive (~WRD00h.tmp) is hidden password protected with password **happyday** (-hphappyday). The rest of the commands are to cleanup (also 12, 13, 14). The RAR file was uploaded successfully to the C2 node.

## 4.4 Parallel Detection with Security Analytics

During the early stages of an intrusion the adversary is typically very busy performing tasks such as moving laterally to additional systems on the network, dumping password hashes, mapping out the network, and stealing data. After they accomplish this part of the mission, the C2 channels will go quiet for the most part, by which we mean there maybe outbound connection attempts but there is nothing on the destination node listening on that port. Another symptom of this "quiet time" is that the adversary may choose to park their domain names, by which we mean, that the domain names resolve to a legitimate IP address such as a Google IP, or resolve to the loopback address, or some other non-malicious IP. The adversary may occasionally interact with a system on the network to ensure that they still have access to the network, but the activity is typically minimal. The only exception to this behavior is if the adversary comes in for another round of data theft. This current case is a perfect example of this scenario, where at least one of the APT groups has had access to the network for at least 3 years (since 2010).

This "quiet time" presents a challenge to signature based network devices, since there are no payloads to hit on. RSA IR uses a simple application rule within SA to identify suspected sessions in the form of TCP beaconing. TCP beaconing is periodic attempts to create a TCP session with the Command and Control infrastructure.

| Condition 1 | Condition 2 |
|---|---|
| ip.proto=6 | ip.proto=6 |
| streams=1 | streams=2 |
| risk.info='flags_syn' | risk.info='flags_syn' |
| risk.info!='flags_ack' | risk.info!='flags_psh' |
| risk.info!='flags_psh' | risk.info!='flags_fin' |
| risk.info!='flags_fin' | alert!='rfc1918_dst' |
| risk.info!='flags_rst' | payload=0 |
| alert!='rfc1918_dst' | |

Table 1: SA beacon detection rules

The first condition would appear when the remote listener is not listening on that port. There could be any number of reasons for this such as an HTRAN[10] listener has been shut down or the Trojan server component itself has been shut down. The packets generally follow the 3, 6, N periodicity. This means that the first SYN packet is sent, 3 seconds later the second SYN packet is sent and 6 seconds later the 3rd SYN packet is sent. N stands for the Trojan's internal timer for attempting another connection.

| | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 172.20.25.2 | 85.13.234.200 | TCP | 74 | S+, cppdp > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 |
| 2 | 3.027974 | 172.20.25.2 | 85.13.234.200 | TCP | 74 | [TCP Retransmission] S+, cppdp > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 |
| 3 | 9.043538 | 172.20.25.2 | 85.13.234.200 | TCP | 62 | [TCP Retransmission] cppdp > http [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 |

**Figure 46 Suspicious TCP Beaconing**

The second condition expects a response from the server, but usually in the form of RST/ACK packets. This would indicate a host configuration that resets TCP connections when a service isn't bound to that TCP port.

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 0.000000 | 172.20.10.62 | 122.10.91.18 | TCP | 62 | ttc-etap-ns > domain [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 |
| 2 | 0.000489 | 122.10.91.18 | 172.20.10.62 | TCP | 60 | domain > ttc-etap-ns [RST, ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 3 | 0.515978 | 172.20.10.62 | 122.10.91.18 | TCP | 62 | [TCP Retransmission] ttc-etap-ns > domain [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 |
| 4 | 0.516398 | 122.10.91.18 | 172.20.10.62 | TCP | 60 | domain > ttc-etap-ns [RST, ACK] Seq=1 Ack=1 Win=65535 Len=0 |
| 5 | 1.119518 | 172.20.10.62 | 122.10.91.18 | TCP | 62 | [TCP Retransmission] ttc-etap-ns > domain [SYN] Seq=0 Win=65535 Len=0 MSS=1460 SACK_PERM=1 |
| 6 | 1.119799 | 122.10.91.18 | 172.20.10.62 | TCP | 60 | domain > ttc-etap-ns [RST, ACK] Seq=1 Ack=1 Win=65535 Len=0 |

**Figure 47 Suspicious TCP Beaconing**

The domains for these TCP beaconing sessions can be determined by taking the destination IP address and looking for DNS sessions in alias.ip. Alias.ip is populated by the Advanced DNS parser available through the Security Analytics Live content distribution system. This metadata is generated when a domain lookup results in a valid IP address. A partial view of the metadata on a DNS lookup for drometic.suroot.com is shown below:

service = 53
streams = 2
packets = 2
lifetime = 0
alert = dynamic_dns_query
alias.ip = 200.108.192.31
alias.host = drometic.suroot.com

**Figure 48 IP.Alias Resolution for drometic.suroot.com**

The network traffic for Trojan.FF-RAT was initially discovered with the following rule.

- service = 0
- lifetime = 50-u
- tcp.dstport = 80,81,8000,8080,8443,443,53,21,22,23,10443,1080
- risk.info = 'flags_syn'

This rule looks for traffic that has not been identified by the Security Analytics Service parsers, has been established for more than 50 seconds on well-known ports as well as containing the TCP Flag SYN. Security Analytics attempts to identify sessions based on the layer 3 and layer 4 information such as IP address and source/destination ports. Sessions that grow to over 32MB or over 60 seconds in duration are declared a complete session and sent to the parser logic and written to disk. Long TCP sessions or large downloads will leave session fragments in Security Analytics, so keying off of the TCP Flag SYN, we can find the beginning of such sessions and reduce the amount of data the analyst has to inspect.

The payload discovered in these sessions contains encoded binary data with a single byte XOR key. The XOR key was easy to derive from the traffic as it is exposed when XOR-ing NULL bytes. Applying this key to the payload data yielded a Global Unique Identifier (GUID).

[10] http://www.secureworks.com/cyber-threat-intelligence/threats/htran/

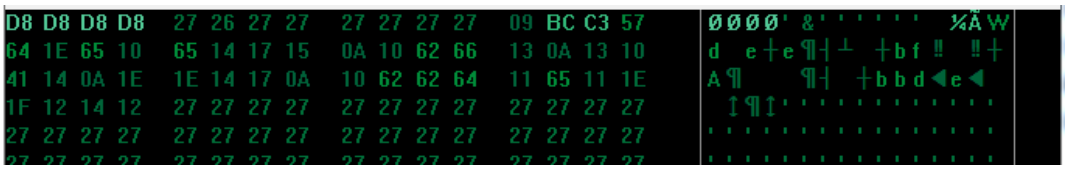



Figure 49 FF-RAT Encoded Beacon

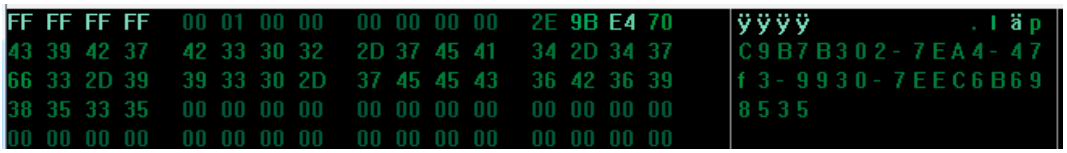



Figure 50 FF-RAT Decoded Beacon

Further analysis revealed more details about the structure of the payload and a FLEX XML parser was developed to aid in discovering more hosts infected with this variant of FF-RAT.

```
<?xml version="1.0" encoding="utf-8"?>
<parsers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="parsers.xsd">
<!--
Written by RSA IR EH
-->
    <parser name="Trojan.FF-RAT" desc="Parses Trojan Beaconing">

    <declaration>
        <session name="SessionBegin" value="begin"/>
        <stream name="StreamBegin" value="begin"/>

        <number name="Request" scope="session"/>
        <number name="Response" scope="session"/>
        <number name="StreamOffset" scope="stream"/>

        <token name="TrojanMagic" value="&#xD8;&#xD8;&#xD8;&#xD8;" options="linestart" />
        <string name="FirstBlock" scope="stream"/>
        <string name="SecondBlock" scope="stream"/>

        <meta name="meta" key="risk.warning" format="Text" />

    </declaration>

    <match name="SessionBegin">
        <assign name="Request" value="0"/>
        <assign name="Response" value="0"/>
    </match>

    <match name="StreamBegin">
        <if name="Request" equal="1">
            <if name="Response" equal="0">
                <assign name="Response" value="1"/>
                <assign name="Request" value="0"/>
            </if>
        </if>
        <if name="Request" equal="0">
            <if name="Response" equal="0">
                <assign name="Request" value="1"/>
            </if>
        </if>
    </match>

    <match name="TrojanMagic">
        <if name="Request" equal="1">
            <payload-position name="StreamOffset" />
            <if name="StreamOffset" equal="4">
                <move direction="forward" value="48"/>
                <read length="4" name="FirstBlock"/>
                <read length="4" name="SecondBlock"/>
                <if name="FirstBlock" equal="$SecondBlock">
                    <register name="meta" value="Trojan.FF-RAT_Beaconing" />
                </if>
            </if>

        </if>
    </match>

    </parser>
</parsers>
```

Figure 51 FF-RAT Detection Parser

# 5. Trojan Families

RSA IR identified eight Trojan families that were being used by one or more APT groups in this engagement. This is a large number of variants for such a small network (~1500 Windows systems), however it shows what a big target this particular network was to the various APT groups that had infiltrated it.

## 5.1 Trojan.Lurker

RSA IR identified several malicious files that it refers to as **Trojan.Lurker** and **Trojan.Lurker2**. This Trojan allowed the adversary to perform the following actions on the infected systems:

1. Execute commands via cmd.exe
2. Execute Files
3. Upload/download files
4. Traverse the file system
5. Enumerate/terminate running processes.

One variant of this Trojan was UPX packed at 18,304 bytes whereas the second variant was not packed and was 48,000 bytes. The primary method of entrenchment was by hijacking the NTMSSVC service by replacing the legitimate DLL name there (ntmssvc.dll) with one of the malicious files. The characteristics of the discovered Trojans are shown below:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes |
|---|---|---|---|---|
| **tpoaed.dll** | 18304 | 127D4ED81A3B107FC20A5B7F951D834B | Sep 07 2009 03:15:30 UTC | price.nspok.com<br>avail.nspok.com<br>202.181.133.97 |
| **lpest.dll** | 18304 | 67595C3D126DFF2FEF1281D4EA0E8F45 | Sep 07 2009 03:15:30 UTC | 220.232.228.11<br>avail.nspok.com<br>202.181.133.97 |
| **asesed.dll**<br>**vdeedd.dll** | 18304 | 836910D7E9CA82AA28123293D2509935 | Sep 07 2009 03:15:30 UTC | mafeng.mircbloger.com<br>avail.nspok.com<br>202.181.133.97 |
| **powms.dll** | 18304 | 1FA362F7611AA30E7DFF1997E3067184 | Sep 07 2009 03:15:30 UTC | rolling.mircbloger.com<br>avail.nspok.com<br>202.181.133.97 |
| **ntmcsvc.dll** | 18304 | 3B8134528C6B9655639B55708A899CDB | Sep 07 2009 03:15:30 UTC | No C2 channels.<br>Misconfigured Trojan. |
| **ntmrsvc.dll** | 48000 | F96D9B121ECCD2C5EBDCD69DCDD6D8D3 | Dec 11 2011 05:12:56 UTC | update01.microsoft-centre.com |
| **ntmrsvc.dll** | 48000 | DE0B3E40B369E025822817F0D54D811E | Dec 11 2011 05:12:56 UTC | update01.microsoft-centre.com |

**Table 2: Trojan.Lurker files details and C2 channels**

Both variants of this Trojan contained configuration data encrypted at the end of the file. Analysis shows that the data is encrypted using a modified version of the DES (ECB) algorithm. The same key is also used to encrypt all network communication. Between the two variants that were discovered two DES keys were found:

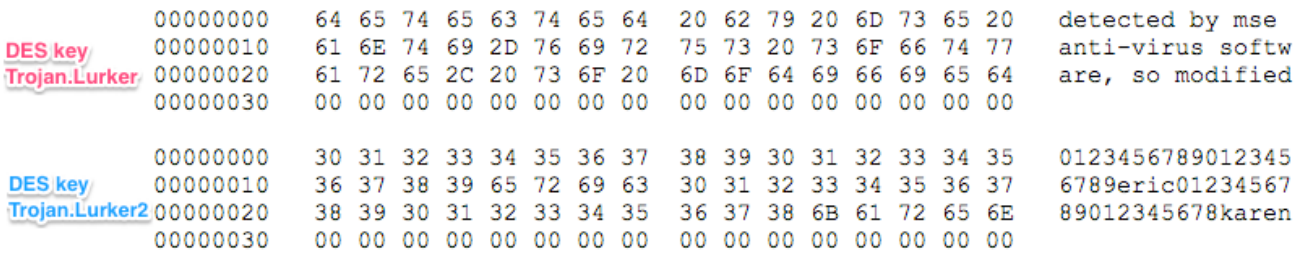



**Figure 52: Trojan.Lurker - DES keys used by each variant**

## 5.2 Trojan.SurperhardCorp

RSA IR identified a Trojan family that it refers to as Trojan.SuperhardCorp. This Trojan allows the adversary to perform the following actions on an infected system:

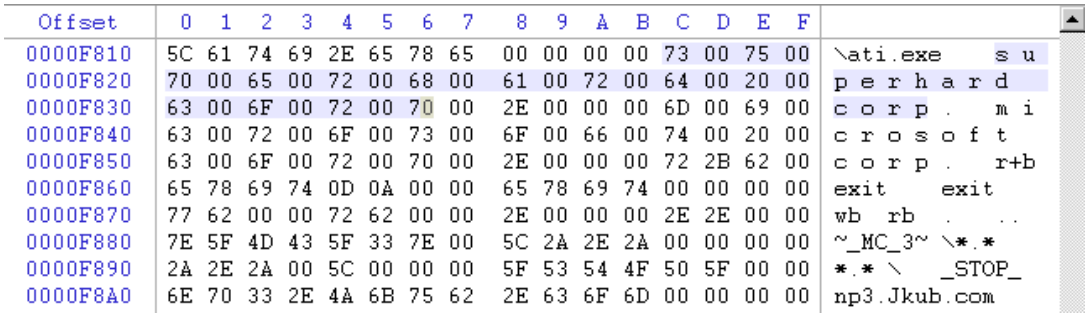
1. Execute commands/files
2. Upload/download files
3. Traverse the file system
4. Enumerate/terminate running processes.

The primary method of entrenchment for this Trojan was a service named IRMON. The Trojan communicates over TCP port 443. The characteristics of the discovered Trojans are shown below:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes |
|---|---|---|---|---|
| **irmon.dll** | 788892 | BE87882D1F306FB9E834FE683EE1A99A | Oct 25 2010 07:31:08 UTC | appear.weibo03.com<br>docume.sysbloger.com |
| **irmon32.dll** | 788992 | 16B2F029BC7BDE4C2EE69B65B323B86E | Oct 25 2010 07:31:08 UTC | ohio.sysbloger.com<br>specs.dnsrd.com |
| **AppMgmt32.dll** | 81408 | 928A2D849047FE1B733A473CFF2EC66C | Jan 20 2010 05:20:33 UTC | np3.Jkub.com<br>ns8.ddns1.com |
| **AppMgmt32.dll** | 769040 | 71AF8D680158C737ACF8304275F4CB2F | Oct 24 2010 13:19:49 UTC | books.mrface.com<br>kieti.ipsecsl.net |

Table 3: Trojan.SuperhardCorp file details and C2 channels

The name of this Trojan comes from a string that is hardcoded in all of these samples. Here is an example:



| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000F810 | 5C | 61 | 74 | 69 | 2E | 65 | 78 | 65 | 00 | 00 | 00 | 00 | 73 | 00 | 75 | 00 | \ati.exe s u |
| 0000F820 | 70 | 00 | 65 | 00 | 72 | 00 | 68 | 00 | 61 | 00 | 72 | 00 | 64 | 00 | 20 | 00 | p e r h a r d |
| 0000F830 | 63 | 00 | 6F | 00 | 72 | 00 | 70 | 00 | 2E | 00 | 00 | 00 | 6D | 00 | 69 | 00 | c o r p . m i |
| 0000F840 | 63 | 00 | 72 | 00 | 6F | 00 | 73 | 00 | 6F | 00 | 66 | 00 | 74 | 00 | 20 | 00 | c r o s o f t |
| 0000F850 | 63 | 00 | 6F | 00 | 72 | 00 | 70 | 00 | 2E | 00 | 00 | 00 | 72 | 2B | 62 | 00 | c o r p . r+b |
| 0000F860 | 65 | 78 | 69 | 74 | 0D | 0A | 00 | 00 | 65 | 78 | 69 | 74 | 00 | 00 | 00 | 00 | exit exit |
| 0000F870 | 77 | 62 | 00 | 00 | 72 | 62 | 00 | 00 | 2E | 00 | 00 | 00 | 2E | 2E | 00 | 00 | wb rb . .. |
| 0000F880 | 7E | 5F | 4D | 43 | 5F | 33 | 7E | 00 | 5C | 2A | 2E | 2A | 00 | 00 | 00 | 00 | ~_MC_3~ *.* |
| 0000F890 | 2A | 2E | 2A | 00 | 5C | 00 | 00 | 00 | 5F | 53 | 54 | 4F | 50 | 5F | 00 | 00 | *.* \ _STOP_ |
| 0000F8A0 | 6E | 70 | 33 | 2E | 4A | 6B | 75 | 62 | 2E | 63 | 6F | 6D | 00 | 00 | 00 | 00 | np3.Jkub.com |

Figure 53: Trojan.SuperhardCorp - binary snippet

## 5.3 Trojan.Derusbi

RSA IR identified two variants of another Trojan family that it refers to as Trojan.Derusbi. This Trojan allows the adversary to perform the following actions on an infected system:

1. Execute commands/files
2. Upload/download files
3. Traverse the file system
4. Enumerate/terminate running processes.

Both variants used the same C2 channel. The characteristics of the discovered Trojans are shown below:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes |
|---|---|---|---|---|
| **mstcpday.dat** | 166703 | AF1746DD9985FE9B19D5036CF45C93F0 | Jan 19 2013 12:31:39 UTC | had-one-job.com |
| **senseron.dll** | 87552 | 0E91F700DF34A2C3633CD49818FA3A61 | Aug 14 2012 16:38:55 UTC | had-one-job.com |

Table 4: Trojan.Derusbi - file details and C2 channels

The first variant (mstscpday.dat) exports function DllRegisterServer, so it can be installed by simply calling it with the regsvr32.exe. During it installation, the Trojan entrenched itself as a service named: **stisvc**, and created a driver which it

loaded in memory and deleted from the file system. The driver name is: C:\WINDOWS\system32\Drivers\{BC87739C-6024-412c-B489-B951C2F17000}.sys.

The second variant of this Trojan consist of two files: an executable file and a compressed configuration file. This variant is entrenched as a service named SENS. The configuration file named seclogon.nls is compressed with the aPACK[11] algorithm:

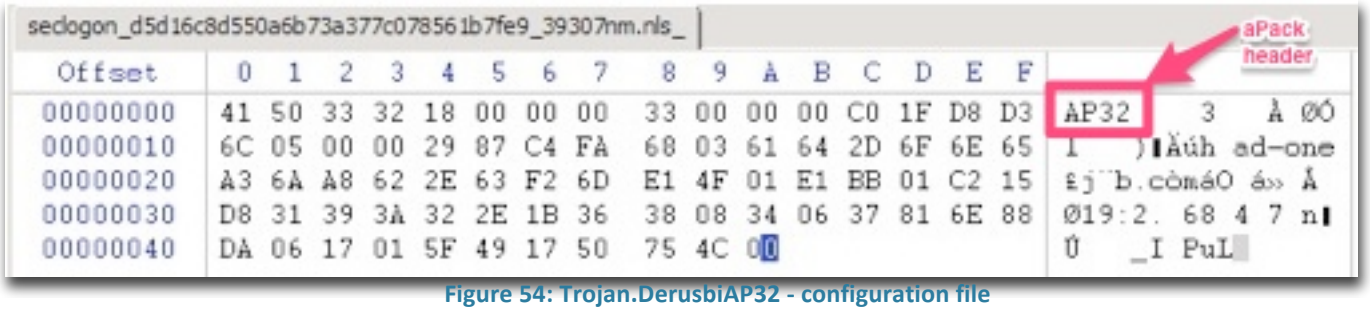



Figure 54: Trojan.DerusbiAP32 - configuration file

We can use decompress this file using utility appack.exe (390A7337B163B819CB99EABE0E8825A4) available at http://www.ibsensoftware.com/index.html. The decompressed data is 1388 bytes (mostly NULLs). The relevant data is shown below:


## 5.4 Trojan.HiKiT

RSA IR identified two malicious files that belong to a Trojan family that RSA IR refers to as Trojan.HiKit. This Trojan consists of two files: an executable file, and a configuration file. The characteristics of the discovered Trojans are shown below:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes |
|---|---|---|---|---|
| **svchost.exe** | 177664 | 7D4F241428A2496142DF1C4A376CEC88 | Feb 27 2012 07:13:30 UTC | 206.205.82.9 |
| **netddesrv.exe** | 177664 | A5F07E00D3EEF7A16ECFEC03E94677E3 | Feb 27 2012 07:13:30 UTC | drometic.suroot.com |

Table 5: Trojan.Hikit - file details and C2 channels

The configuration file named svchost.conf is obfuscated via a 4-byte XOR key. The first four bytes of the file reveal the key. In this case the key was: 0xFA2738CD. The plaintext configuration data is shown below, the data in red was removed as it referenced the victim.

[11] http://www.ibsensoftware.com/index.html


Figure 55: Trojan.Hikit deobfuscated configuration file

The second configuration file was obfuscated with XOR key: 0xCE6C2B25. The file had these characteristics:

```
File Name:  netddesrv.conf
File Size:  456 bytes
MD5:        d7367b3216856cef704e271034e237b5
SHA1:       d9ccbcab076e68a9f0f9a25697a07539397f8c95
```

## 5.5 Trojan.FF-RAT

RSA IR discovered several files that belong to a Trojan family that RSA IR refers to as Trojan.FF-RAT. Many of the discovered files have been legitimately digitally signed. On certain systems, Trojan.FF-RAT also contained a keystroke logger and a driver file. Here are some artifacts regarding this Trojan:

1. This Trojan consisted of at least a DLL file (which always had a .dat extension) and a configuration file, which was always found under: C:\Windows\Media\Windows Config.wav. Both 32bit and 64bit versions were deployed.
2. The configuration file is obfuscated, and then decompressed. See Apendix I for details on how to decrypt this file.
3. On certain systems a driver named fstab.sys existed in memory only.
4. The keystroke loggers were always named [RANDOM]_kl.dll, and a new file is created after each reboot. The old files are not deleted. The keystrokes are stored on a file named iismgr.dat, which was located in the same directory as the DLL. The data in the iismgr.dat is obfuscated via a single byte XOR key: 0xC2.

All digital certificates were issued by: Thawte Code Signing CA – G2. The characteristics of the malicious files belonging to this Trojan family are shown below:

Trojans with signer name: **Xuzhou Chenji Technology Co.,Ltd**
Certificate Serial Number: **19ce1672107145e06fdc45fa2b753f0b**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| **whwbedqu_kl.dll** | 116320 | DB4A2052658836096270 3145C32E743E | Dec 03 2012 08:20:05 UTC | May 13 2013 1:25:22 AM |
| **rmwpnwad_kl.dll** | 108128 | 5E287819699278CEFB490B0D7E768CED | Dec 03 2012 08:18:42 UTC | May 13 2013 1:25:24 AM |
| **nullods.dat** | 65464 | 8C3A13CFF4797A4E74988D05FDD8C287 | Dec 28 2012 07:59:48 UTC | Not available |
| **nullods.dat** | 169400 | 0CEB4CC3665E1190E0FA00FB7153AC22 | Dec 28 2012 07:59:18 UTC | Not available |
| **frtest.dat** | 172128 | CC6999FB9174F2FE0564428EC7F92525 | Dec 28 2012 07:59:18 UTC | Jan 09 2013 4:19:33 AM |
| **frtest.dat** | 68192 | C41A3CB0E7ACCA1AC434F65FB518E58B | Dec 28 2012 07:59:48 UTC | Jan 09 2013 4:19:15 AM |
| **Kqizbmwzopzbqvag.kqi** | 204384 | 41ED24E665759992130BF4C08B5F532E | Jul 25 2012 06:41:30 UTC | Sep 24 2012 10:10:57 AM |

Table 6: Trojan.FF-RAT - file details

Trojans with signer name: **Binzhou XinPin Technology Co.,Ltd.**
Certificate Serial Number: **391e363ec82ad7613db478c178180e8b**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| rvtest.dat | 181352 | 9985668A2F401A4EDE85918A5D417409 | Jul 10 2013 05:36:44 UTC | Aug 28 2013 9:43:57 PM |
| frkeser.dat | 181352 | B76A3595523E6050C4034294257323CA | Jul 10 2013 05:36:44 UTC | Jul 28 2013 1:38:56 AM |
| frkeser.dat | 72296 | 939587C6CEB084273B424D982C52AC5A | Jul 10 2013 05:36:27 UTC | Jul 28 2013 1:38:01 AM |
| fmconull.dat | 181352 | DB35A3A80BD62EFF91EAD4A2046D26A5 | Jul 10 2013 05:36:44 UTC | Jul 15 2013 2:31:46 AM |
| fmconull.dat | 72296 | 92E9F1FB37EE75415235C4E567DE0F1B | Jul 10 2013 05:36:27 UTC | Jul 15 2013 2:31:25 AM |
| x8.txt | 127592 | 838B97B916CA2A8A9855D8257A6826E7 | Jul 10 2013 05:36:31 UTC | Jul 15 2013 2:31:40 AM |

**Table 7: Trojan.FF-RAT - file details**

Trojans with signer name: **Hangzhou Degou Information Technology Co.,Ltd.**
Certificate Serial Number: **64477c85f26c2ca67d76468434263e0e**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| **bxevkwcb_kl** | 86192 | 90bfea7038a8a25e1e70ba76291b2016 | Jan 11 2012 09:51:40 UTC | Jan 16 2012 2:54:16 AM |

**Table 8: Trojan.FF-RAT - file details**

Trojans with signer name: **Henan Lvcheng Tianxia Information Technology Co.,Ltd**
Certificate Serial Number: **06b587cdb256cd4224baa55eb3ff2a98**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| frtest.dat | 191640 | B8DF0D1A8EC15C40692D507E62F9EE80 | Mar 20 2012 04:05:37 UTC | Aug 7 2012 5:31:40 AM |
| frtest.dat | 80096 | 705EBCFCE803D3FB69F409BABAF1376E | Mar 20 2012 04:05:05 UTC | May 31 2012 6:02:20 AM |

**Table 9: Trojan.FF-RAT - file details**

**Unsigned Trojan.FF-RAT files**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| ngpqdasi_kl.dll | 101376 | 071B2A2CF343A62EC7C75592362593BC | Dec 03 2012 08:18:42 UTC | N/A |
| lcruhypy_kl.dll | 109568 | E36DA01D2C47C308CDA5AF49272F3FBD | Dec 03 2012 08:20:05 UTC | N/A |
| fmnonull.dat | 61440 | B7F87AF5AFF0A68DE408B112A5A95049 | Dec 28 2012 07:59:48 UTC | N/A |
| fmnonull.dat | 165376 | 21C5FC01CED8B327A6AC1F31B90C525B | Dec 28 2012 07:59:18 UTC | N/A |

**Table 10: Trojan.FF-RAT - file details**

All the C2 domains related to this Trojan resolved to the same IP address 198.55.120.222 at the time of the engagement. Overall the following domain names were used by Trojan.FF-RAT:

**mno80.dwy.cc** **fan025.yahoolive.us** **pcal2.dwy.cc** **pcal2.yahoolive.us**
**mno995.dwy.cc** **fan080.yahoolive.us** **3h01.dwy.cc**

## 5.6 Trojan.PlugX

RSA IR identified another malicious file that it refers to as Trojan.PlugX. This Trojan is well documented in the security community and has several capabilities including uploading/downloading files, executing commands, and listing processes. Here are the characteristics of the files related to this Trojan family:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes/Notes |
|---|---|---|---|---|
| **b.exe** | 366734 | 3BC77F178ACC60A47106834658E78BCF | Feb 20 2011 08:19:48 UTC | Self-extracting executable |
| **iehelper.exe** | 14608 | 288B1C32B3B951C79E78F764DD1B08F8 | Jun 17 2012 07:51:16 UTC | Sandboxie L.T.D loader file. |
| **sbiedll.dll** | 181760 | 86D7F18C89CEFE4C43DB9F38755CC33D | May 17 2013 09:38:58 UTC | Sandboxie L.T.D file. |
| **helper.url** | 113874 | 1F8F685815648E3308EA096C1367BA27 | N/A | Obfuscated and compressed code |
| ***[final.dll]*** | 154112 | 35958c670840819889f18a69db72ac3b | Oct 17 2012 08:33:02 UTC | dns2.ipv6do.com up.outhmail.com |

**Table 11: Trojan.PlugX - file details and C2 channels**

File b.exe, which is a self-extracting file, is the dropper for this variant of Trojan.PlugX. The archive contains three files iehelper.exe, sbiedll.dll, and helper.url. The first two files are loaders for the third file, which is obfuscated shellcode. The two loader files iehelper.exe (which is signed by SANDBOXIE L.T.D) and sbiedll.dll are files used by Sandboxie, an isolated operating environment, which attempts to protect users from malicious programs. The iehelper.exe file will load and start the shiedll.dll, which injects the code contained in helper.url into a running process.

After the iehelper.exe and sbiedll.dll files have been executed, the data from the helper.url file will be injected into a running process. The helper.url file contains raw shell code, which has two layers of obfuscation. The first layer of deobfuscation starts with XORing 113842 bytes of data in helper.url with the 0xC0. The second stage of deobfuscation is more complex, involving a series of mathematical operations on the data, as well as decompressing a portion of the data of helper.url using RTLDecompressBuffer API call. Once these steps are performed a DLL file is produced (given the name final.dll) which is the actual Trojan.PlugX file. This Trojan has been configured to beacon out to **dns2.ipv6do.com** and **up.outhmail.com**. Here is a sample beacon from this Trojan:

```
POST /update?id=000f7578 HTTP/1.1
Accept: */*
X-Session: 0
X-Status: 0
X-Size: 61456
X-Sn: 1
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; .NET4.0C; .NET4.0
E; .NET CLR 2.0.50727; .NET CLR 3.0.04506.648; .NET CLR 3.5.21022)
Host: dns2.ipv6do.com
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache
```

## 5.7 Trojan.Gh0st

RSA IR identified remnants of what it refers to as Trojan.Gh0st. One of the variants was digitally signed, but the certificate is no longer valid. Both variants consisted of a separate configuration file that contained the C2 information. All recovered configuration files contained the same C2 node: **ru.pad62.com**. Malware analysis shows the Gh0st magic string Drag0n in its initial beacon in network traffic.

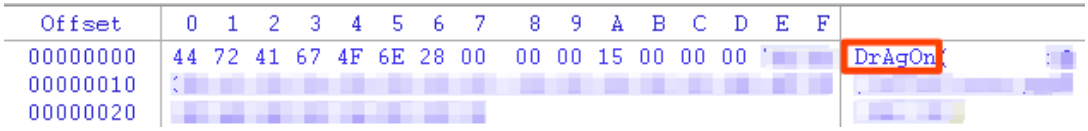



Figure 56: Trojan.Gh0st magic string

Trojans with signer name: **Shanghai Qiangwang Technology Co., Ltd**
Certificate Serial Number: **027c0d1cecf1e7e82eb89fc3d5512613**

| File Name | Size | MD5 Hash | Compile Time | Signing Time |
|---|---|---|---|---|
| *[Omitted]*x.exe | 156161 | C2E664463269D9A4E5E1F201DA867E0F | Apr 03 2012 09:59:50 UTC | Jun 4 2012 1:17:57 AM |
| 6to4adv.dll | 113264 | 4E5C58E519AF4DB9CD444350A4241D5A | Jul 07 2012 05:01:37 UTC | Jul 07 2012 12:23:03 AM |
| BusMgr.sys | 30192 | 284295406F74C7831AA58EF46F3AD10B | Jun 27 2012 14:43:21 UTC | Jul 07 2012 8:59:31 AM |

Table 12: Digitally Signed Trojan.Gh0st file details

**Unsigned Trojan.Gh0st files**

| File Name | Size | MD5 Hash | Compile Time | Notes |
|---|---|---|---|---|
| MSODBC.dll | 51232768 | 2F08BFF22FD8F3D264AE72BBC4EF7AD9 | May 15 2012 06:24:15 UTC | Artificially increased size |
| msodbc.dll | 32768 | 1F206932514C3ADDC94160F27170AC7F | May 15 2012 06:24:15 UTC | Actual PE size |

Table 13: Unsigned Trojan.Gh0st file details

## 5.8 Trojan.PoisonIvy

RSA IR discovered two active variants of PoisonIvy. This Trojan is well documented in the security community and has several capabilities including uploading/downloading files, executing commands, and listing processes, observe/control the user's GUI, keystroke logging, etc. Here are the characteristics of the files related to this Trojan family:

| File Name | Size | MD5 Hash | Compile Time | C2 Nodes/Notes |
|---|---|---|---|---|
| dbServer.exe | 32768 | 8adcbec6614fdcb297311e7dd5dc3de3 | Apr 27 2013 13:50:00 UTC | Loader |
| res.db | 22528 | 981ebda6cf315af63ed46e2a367c0b2b | N/A | Obfuscated DLL |
| Decryp_res.db | 22538 | bd864c39cb8118356b061f4843a39add | Apr 27 2013 13:37:59 UTC | Decrypted version of res.db |
| memshare.dat | 7099 | 4aefaac9f96c01398ad96ebe8ad5c5f3 | N/A | Obfuscated code |
| timebios.dll | 20448 | 18f55f3533101f8c0dce96c070d22736 | Jan 28 2013 10:14:25 UTC | Loader |
| share.dat | 6710 | 561130a9d3e483b397ff12e8dd3a1a32 | N/A | Obfuscated code |

Table 14: Trojan.PoisonIvy - file details

The PoisonIvy password chosen for both of these samples of PoinsoIvy was: **15911117665**. Both samples beacon out to: **2012jg.sony36.com**. The communication protocol did not deviate from the standard PoisonIvy protocol that is also publicly available. In fact the publicly available PoisonIvy server will interact with a system infected with this Trojan. Below is the configuration information in memory that shows the PoisonIvy C2 node and the password:

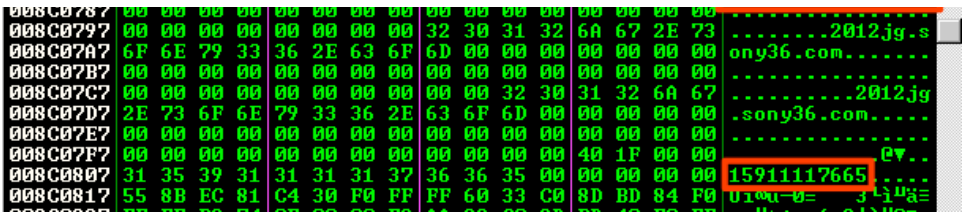



Figure 57: Trojan.PoisonIvy - Memory snippet containing password

Here is a screenshot from the PoisonIvy server after a system infected with this Trojan.PoisonIvy connected to it:

| **Server Settings:** | |
|---|---|
| ID | |
| Group | |
| DNS/Port | Direct: 2012jg.sony36.com |
| Proxy DNS/Port | |
| Proxy Hijack | No |
| ActiveX Startup Key | |
| HKLM Startup Entry | |
| File Name | |
| Install Path | C:\WINDOWS\System32\svchost.exe |
| Keylog Path | C:\WINDOWS\System32\svchost |
| Inject | No |
| Process Mutex | 20130130 |
| Key Logger Mutex | |
| ActiveX Startup | No |
| HKLM Startup | No |
| Copy To | No |
| Melt | No |
| Persistence | No |
| Keylogger | No |

**Figure 58: PosionIvy server side**

# 6. Conclusion

This case study shows the typical flow of an investigation using ECAT and Security Analytics, which give RSA IR the visibility needed to successfully and efficiently investigate intrusions with the intent of successful remediation. Only with full network and endpoint visibility can investigators ensure they've identified all malware deployed or C2 channels used by an adversary. Additionally, this visibility is critical after remediation of the intrusion, as APT adversaries will try to reenter the environment. Most APT groups are politically or economically motivated, state sponsored, highly skilled, and therefore capable of sustaining long-term campaigns against their intended targets. By having proper visibility over the network you will be able to proactively identify new infections and more rapidly remediate them, reducing your exposure and the adversary's opportunity to steal or manipulate more data. Traditional forensics, a.k.a dead-box forensics, is not suitable for the nature of today's intrusions because it is too slow and a very reactive process. An RSA IR analyst can triage a remote system in 10 – 20 minutes and without affecting the endpoint. A traditional forensic process will not even have an image acquired in that time, not to mention the user disruption.

While other defense mechanisms such as perimeter controls and education of users are extremely important at preventing an intrusion, the next line of defense is quick detection of malicious activity once prevention fails. It is this proactive approach at reviewing both network traffic and endpoints for signs of malicious activity that gives companies the best chance at quickly identifying malicious activity.

ECAT cuts down the analysis time by allowing analysts to whitelist files that have already been analyzed or are trusted, focusing the analysis on only new files that appear on the endpoints. Furthermore, ECAT uses a variety of techniques to distinguish between suspicious and normal activity in both memory and on disk, enabling the analyst to focus on the most suspicious activity. ECAT's ability to process Yara signatures is also an extremely useful feature that not only allows a company to incorporate their own intelligence into the product, but also import signatures from other intelligence groups that share these signatures. Lastly, ECAT allows the analyst to quickly triage endpoints by analyzing their MFTs. As has been illustrated in many examples in this report, whenever a malicious artifact is found in a system, a quick triage is necessary because it can reveal many other artifacts such as signs of data exfiltration and remnants of older Trojans.

Security Analytics complements ECAT by allowing the analyst to look for anomalies in the communication protocols used by malware, or if the C2 channels are dormant, by identifying beaconing behavior. This is a very powerful approach that nets many of the C2 channels in an incident.

# 7. Appendix I

Trojan.FF-RAT consists of a configuration file that in this case was always found under: C:\Windows\Media\Windows Config.wav. This configuration file is RC4 encrypted and aPACK compressed. This section will demonstrate how an analyst can decrypt and decompress this file to reveal the configuration information.

First we need to understand the structure of an aPACK compressed file. So, we start with a test file that we compress using the appack.exe[12] utility.

test.txt

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 54 | 68 | 69 | 73 | 20 | 69 | 73 | 20 | 61 | 20 | 74 | 65 | 73 | 74 | 2E | | This is a test. |

Figure 59: Plaintext file

Next we use the appack.exe utility to compress this file by running: **appack.exe c test.txt test.ap32**. The aPACK compressed file consists of the following structure:

- 1st DWORD – aPACK magic header
- 2nd DWORD – Total Header length (i.e. the first 6 DWORDs)
- 3rd DWORD – Length of compressed data
- 4th DWORD – CRC32 hash of compressed data
- 5th DWORD – Length of decompressed data
- 6th DWORD – CRC32 hash of decompressed data
- The rest of the bytes are the compressed data.

This structure is depicted below:

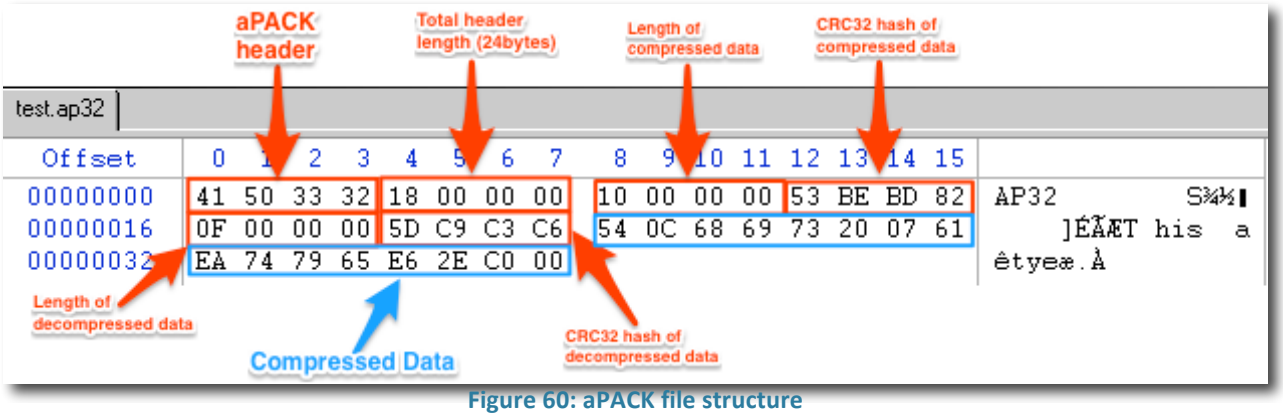



Figure 60: aPACK file structure

Now we go back to a sample Windows Config.wav file. This file has the following structure:

- 1st DWORD – Hardcoded value (0x19860609), which may represent a date, that is, YYYYMMDD or YYYYDDMM.
- 2nd DWORD – Obfuscated RC4 key. De-obfuscated by XOR-ing with 1st DWORD.
- 3rd DWORD – NULLs
- 4th DWORD – Same hardcoded value as 1st DWORD.
- 5th DWORD – Length of compressed data
- 6th DWORD – Length of decompressed data.

[12] http://www.ibsensoftware.com/files/aPLib-1.01.zip

- The rest of the data is aPACK compressed and RC4 encrypted (offset 0x18 – end)

This structure is demonstrated below:

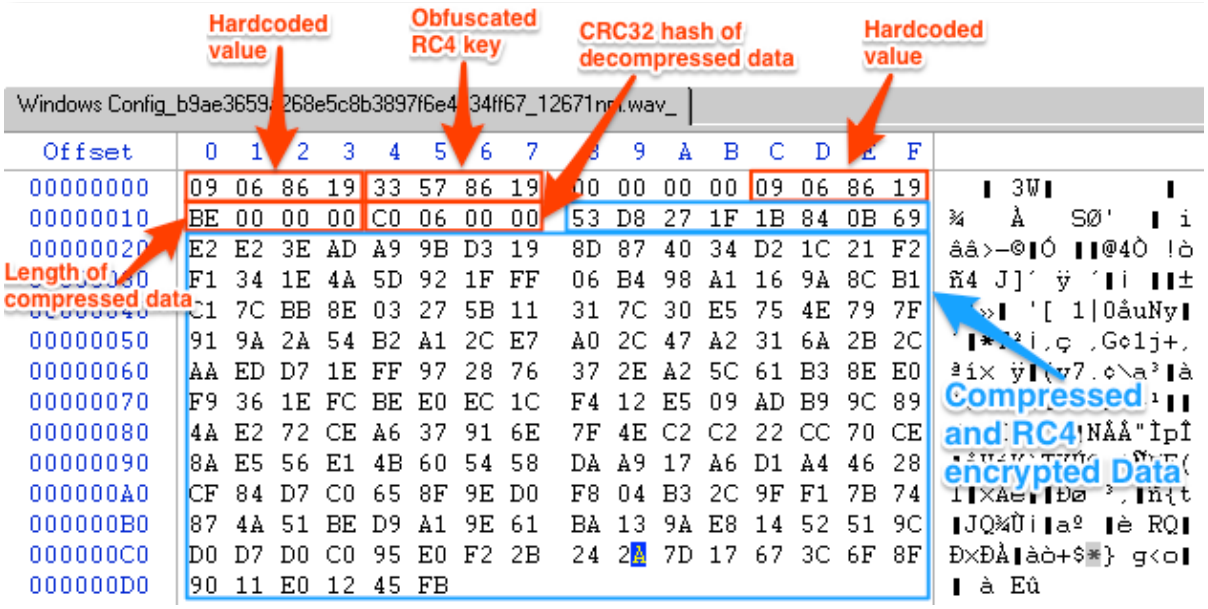



Figure 61: Trojan.FF-RAT configuration file structure

The RC4 key is derived by XOR-ing the first two DWORDs. In this case: **0x19860609** XOR **0x19865733 = 0x0000513A**. The Trojan then prints the ASCII version of 0x0000513A, so essentially our RC4 key is 64-bits long as shown below:

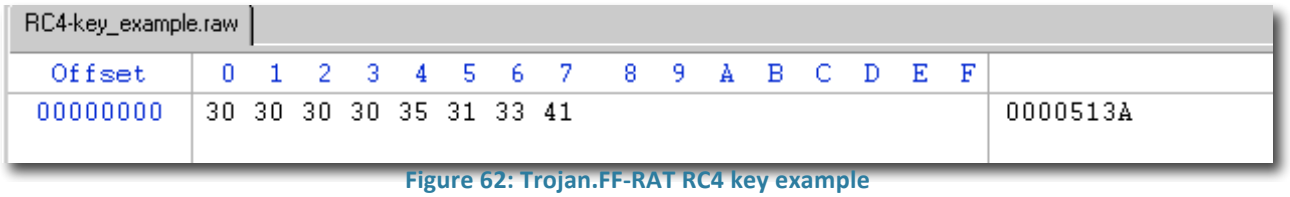



Figure 62: Trojan.FF-RAT RC4 key example

So using RC4 key 0x3030303035313341 we now decrypt the data in the Windows Config.wav file starting at file offset 0x18 until the end. The decryption operation results in this data:

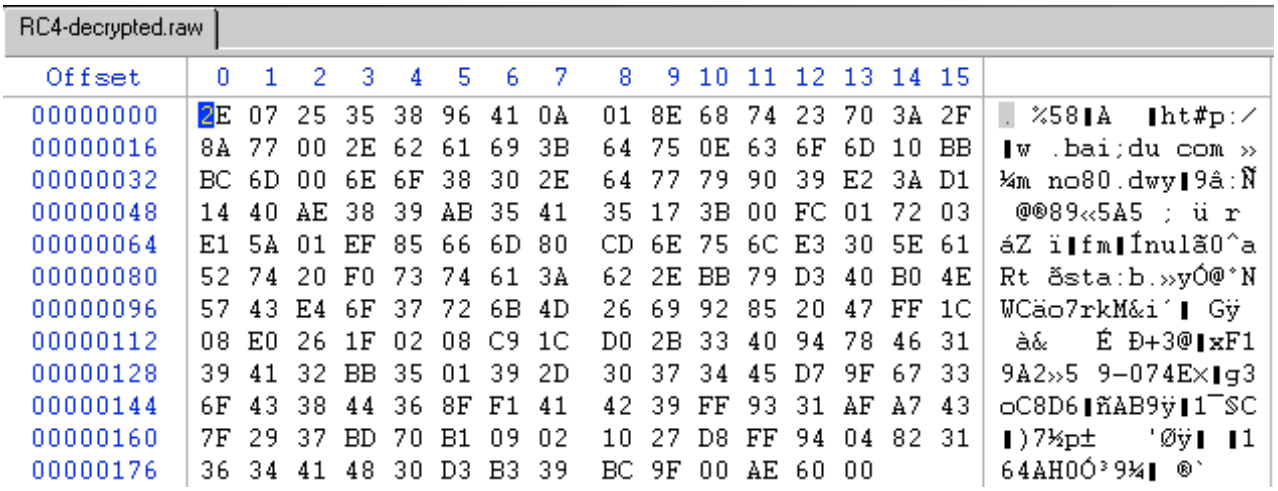



Figure 63: Trojan.FF-RAT RC4 decrypted configuration file

The RC4 decrypted data is aPACK compressed but without the header. We actually do have all the pieces of the header except one: the CRC32 of the decompressed data. Lets list the structure of the aPACK compressed file to demonstrate by referencing figure 61:

- 1st DWORD – AP32 (we can create this ourselves)
- 2nd DWORD – Length of the header (we can create this ourselves, i.e. 0x18000000)
- 3rd DWORD – Length of compressed data (we have this from the configuration file figure 62 (i.e. 0xBE000000))
- 4th DWORD – CRC32 of compressed data (we can calculate this ourselves since we have the data)
- 5th DWORD – Length of decompressed data (we have this from the configuration file figure 62 (i.e. 0xC0060000))
- 6th DWORD – CRC32 of decompressed data (we have no way of knowing or calculating this since we do not have the decompressed data)

So, we are missing one critical piece of information, namely the CRC32 hash value of the decompressed data, and there is no way of generating or knowing this in advance since we are trying to decompress the data. Lets put an aPACK header together with the information we have along with the data we decrypted (figure 64), and add NULLs for the 6th DWORD since we do not have this information:

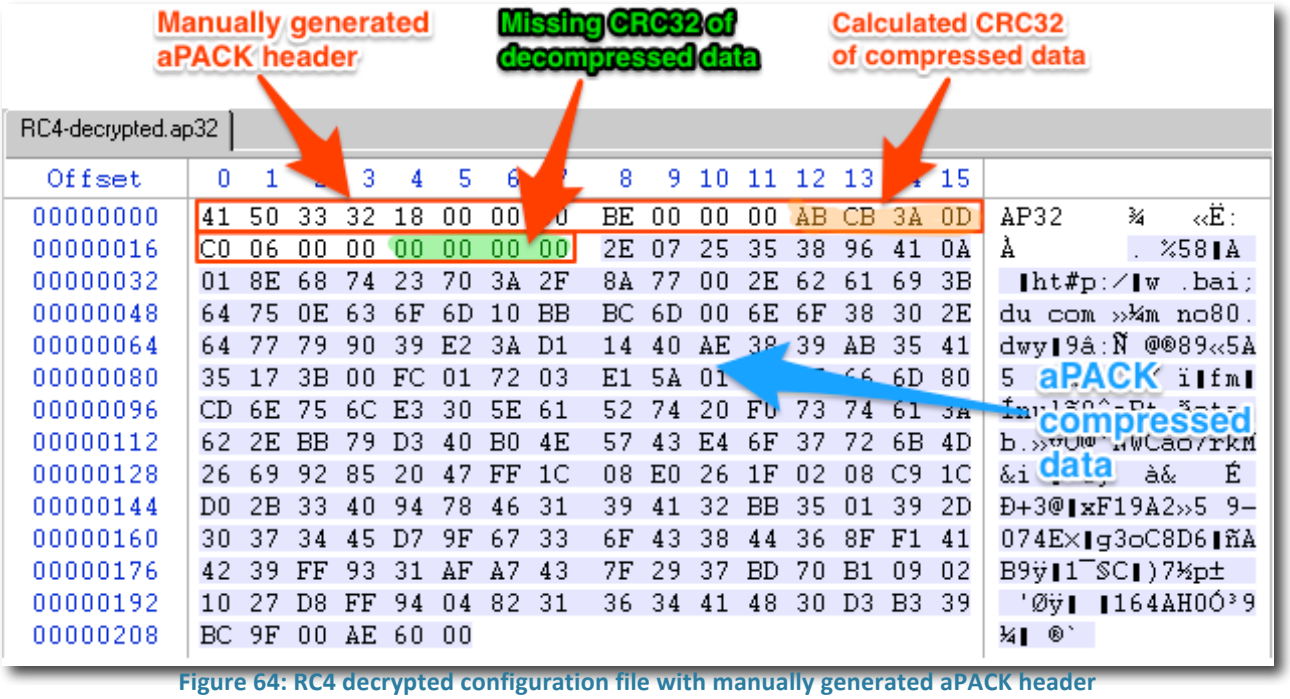



Figure 64: RC4 decrypted configuration file with manually generated aPACK header

When we execute the appack.exe tool to decompress this file we get an error message:

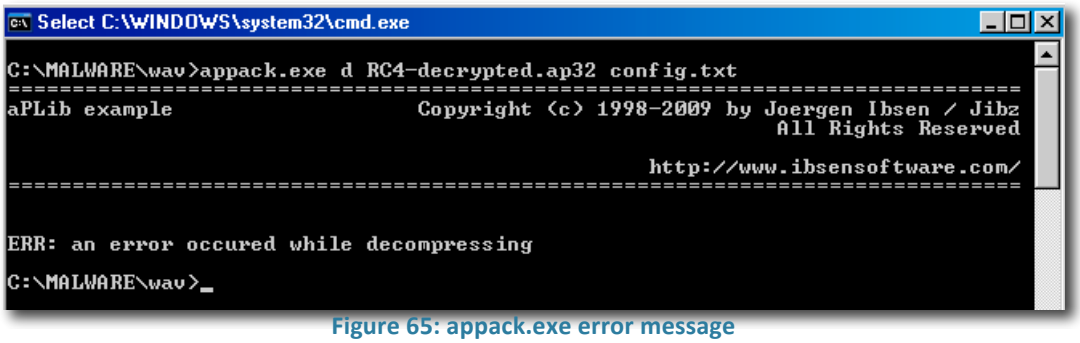



Figure 65: appack.exe error message

It is obvious that the appack.exe utility is throwing this error because the CRC32 of the decompressed data does not match what it calculates after it is done decompressing the file. So, we need to get around this error by identifying and modifying the code in appack.exe where this CRC32 hash check is made in order to make the utility continue executing regardless of whether the CRC32 hash of the decompressed data matches what is on the header.

A little debugging of this tool leads us to the code responsible for this CRC32 check. The code is shown below:

```
00402C01  .  83C4 08      ADD ESP, 8
00402C04  .  3B46 14      CMP EAX, DWORD PTR DS:[ESI+14]
00402C07  .  89D8         MOV EAX, EBX
00402C09  .  74 03        JE SHORT appack.00402C0E          Jump if CRC32 hash matches
00402C0B  >  83C8 FF      OR EAX, FFFFFFFF
00402C0E  >  894424 1C    MOV DWORD PTR SS:[ESP+1C], EAX
00402C12  .  61           POPAD
00402C13  L. C3           RETN
```

Figure 66: Disassembly of appack.exe

At address 0x00402C09 we see a conditional jump-if-equal (JE), which means that if the CRC32 hash of the decompressed data matches the value of the 6th DWORD in the header, the jump will be taken. We can modify the code of this utility to make the jump here unconditional (JMP) thus make the utility think that the CRC32 check is always successful.

```
00402C01  .  83C4 08      ADD ESP, 8
00402C04  .  3B46 14      CMP EAX, DWORD PTR DS:[ESI+14]
00402C07  .  89D8         MOV EAX, EBX
00402C09     EB 03        JMP SHORT appack.00402C0E         Unconditional jump
00402C0B  >  83C8 FF      OR EAX, FFFFFFFF
00402C0E  >  894424 1C    MOV DWORD PTR SS:[ESP+1C], EAX
00402C12  .  61           POPAD
00402C13  L. C3           RETN
```

Makes jump unconditional

We can permanently patch the appack.exe file by using a Hex editor and changing byte 0x74 with 0xEB. This particular instruction is located at file-offset 0x2009 as shown below:

appack.exe — Change byte to 0xEB

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00002000 | 00 | 83 | C4 | 08 | 3B | 46 | 14 | 89 | D8 | 74 | 03 | 83 | C8 | FF | 89 | 44 | ▮Ä ;F ▮Øt ▮Èÿ▮D |
| 00002010 | 24 | 1C | 61 | C3 | 60 | 8B | 74 | 24 | 24 | 8B | 1E | 83 | C8 | FF | 81 | FB | $ aÃ` ▮t$$▮ ▮Èÿ▮û |
| 00002020 | 41 | 50 | 33 | 32 | 75 | 0B | 8B | 5E | 04 | 83 | FB | 18 | 72 | 03 | 8B | 46 | AP32u ▮^ ▮û r ▮F |

Figure 67: Patching appack.exe

Now, when we execute this patched version of appack.exe we successfully decompress any Trojan.FF-RAT configuration file:

```
C:\WINDOWS\system32\cmd.exe
C:\MALWARE\wav>appack.exe d RC4-decrypted.ap32 config.txt
==============================================================================
aPLib example                  Copyright (c) 1998-2009 by Joergen Ibsen / Jibz
                                                        All Rights Reserved

                                                  http://www.ibsensoftware.com/
==============================================================================

decompressed 214 -> 1728 bytes in 0.00 seconds

C:\MALWARE\wav>
```

Successfully decompressed

The relevant parts of the decrypted configuration file are shown below:

config.txt

| Offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00000000 | 2E | 25 | 35 | 38 | 96 | 41 | 00 | 00 | 00 | 00 | 00 | 00 | 68 | 74 | 74 | 70 | .%58▮A http |
| 00000016 | 3A | 2F | 2F | 77 | 77 | 77 | 2E | 62 | 61 | 69 | 64 | 75 | 2E | 63 | 6F | 6D | ://www.baidu.com |
| 00000032 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000304 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000320 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6D | 6E | 6F | 38 | mno8 |
| 00000336 | 30 | 2E | 64 | 77 | 79 | 2E | 63 | 63 | 3A | 38 | 30 | 00 | 00 | 00 | 00 | 00 | 0.dwy.cc:80 |
| 00000352 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000368 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000384 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6D | 6E | 6F | 39 | mno9 |
| 00000400 | 39 | 35 | 2E | 64 | 77 | 79 | 2E | 63 | 63 | 3A | 39 | 39 | 35 | 00 | 00 | 00 | 95.dwy.cc:995 |
| 00000416 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000976 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00000992 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 66 | 6D | 63 | 6F | 6E | 75 | 6C | 6C | fmconull |
| 00001008 | 2E | 64 | 61 | 74 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | .dat |
| 00001024 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 66 | 73 | 74 | 61 | 62 | 2E | 73 | 79 | fstab.sy |
| 00001040 | 73 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | s |
| 00001056 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 4E | 57 | 43 | 57 | 6F | 72 | 6B | 73 | NWCWorks |
| 00001072 | 74 | 61 | 74 | 69 | 6F | 6E | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | tation |
| 00001088 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | FF | FF | 00 | 00 | E0 | 1F | 02 | 00 | ÿÿ à |
| 00001104 | 00 | 00 | 00 | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 6D | 6E | 6F | 33 | mno3 |
| 00001120 | 40 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | @ |
| 00001136 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 46 | 31 | 41 | 32 | F1A2 |
| 00001152 | 31 | 35 | 46 | 39 | 2D | 30 | 37 | 34 | 45 | 2D | 34 | 30 | 33 | 34 | 2D | 38 | 15F9-074E-4034-8 |
| 00001168 | 44 | 36 | 36 | 2D | 41 | 42 | 39 | 45 | 39 | 31 | 42 | 36 | 43 | 39 | 42 | 37 | D66-AB9E91B6C9B7 |
| 00001184 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00001456 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |
| 00001472 | 01 | 00 | 00 | 00 | 31 | 36 | 34 | 32 | 31 | 30 | 34 | 36 | 39 | 34 | 00 | 00 | 1642104694 |
| 00001488 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | |

**Figure 68:RC4 decrypted and aPACK decompressed Trojan.FF-RAT configuration file**