A case study on Stuxnet and Flame Malware

K F Morton
School of Information Technology and Engineering
Israel Institute of Technology, Haifa, Israel
morton@ee.technion.ac.il

David Grace
Institute of Information Science and Technologies
Palmerston North, New Zealand
david.grace@topmail.co.nz

Abstract: - A vast numberof malware is packed by packers. Obfuscation tools are not only cost effective and readily available but also provide an effective camouflage to malware code. Unpacking and analyzing the malicious code may appear aoptimum solution to this problem; but provided with gigantic number of malware being released every single day, this is not a tranquil peace of work for security companies and researchers. In this paper we aim to provide a comprehensive summary of packer problem with practical demonstration of their effectiveness and we will be reviewing various generic techniques to handle this problem.

Key-Words: -malware, packers, obfuscation, reverse engineering, analysis, Stuxnet, Flame

1 Introduction

Type of malware includes malicious programs such as stuxnet, flame, mydoom, bots, worms, spyware, rootkits, viruses, torjan horses, adware etc.

Viruses spread across normally during execution of any program, software or transferring documents. Worms are stand alone software normally affects computer during booting process. Torjan horses are manually attached to the piece of software, which forces user to install unnecessary software. Backdoor allows user to enter server or any main program by passing normal authentication process. Spyware collects and distributes information about users access pattern. Exploit allows users to use secured devices by weakening security attributes. Rootkit is type of backdoors, which hides attacker/hacker traces once user logs into the system.

The main features for malware detection and prevention are security, safety, stealth and sustainability. These are explained briefly in this section.

Security: Since the framework is used for security analysis the platform should not be victim for such security attacks. Components of the framework have to be managed in secure manner.

Safety: With the framework users handle various malicious files on the platform this may cause unexpected incident. Especially for the dynamic analysis process the framework would run the captured malicious program and this process should not cause any unwanted damage to innocent network users. Particularly network access from the host that runs malware should be separated and should be controlled in a way any harmful network traffic does not go out.

Stealth: Once the framework is deployed on the Internet due to its nature of the system attacker may be able to identify the existence or activities of the platform. To minimize unnecessary risks the behaviour of the system should not be very disrupting so the program especially network access from the malware need to be handled with care.

Sustainability: Since counter malware activity is continuous process of keeping acquiring new specimens, storing acquired malware to database, analysis of specimen, generating signature for detection and reporting. As new malwares are always created thus this process cannot be stopped. The framework should be operable in continuous manner.

Section 2 outlines features of stuxnet malware. Section 3 outlines flame malware and MyDoom malware and it'sobfuscation is explained in section 4. Obfuscation can also be applied to stuxnet and flame malwares and conclusion in section 5 gives the future work on this process.

2 Stuxnet Malware

Stuxnet is the first malicious threat targeting industrial control system such as gas pipeline, power plant etc [1]. It has mainly four features: command

and control, multiple propagation methods, stolen VeriSign driver certificate, and a root kit [1].

Stuxnet is primarily designed to corrupt Siemens (S7-315 and S7-417) and predicted that in future it might also corrupt the hard-coded passwords of the Siemens step 7 software. According to Information Technology Council of Iran's Industry and Mines Ministry, Iran had identified that IP addresses of 30000 industrial computer system were infected since September 25th, 2010.

2.1 Basic Characteristics of Stuxnet

Stuxnet includes four main files, such as .LNK file, ~WTR4141.tmp, ~WTR4132.tmp, encoded payload.dll with a selection of different files such as .dll, .exe, .dat, .sys, .tmp [2]. These files are all packed in a .dll file, which is known as UPX packed .dll file, also free, portable, and executable in fraction of time and are available in different extensible formats. The ultimate goal of stuxnet is to interrupt the systems by reprogramming programmable logic controller (PLC) so that attackers can easily take control of PLCs [3]. Stuxnet is also designed to transfer data about production lines from the industrial plants of Iran to the outside location [4].

The stuxnet malware can be spread via CD, flash memory (USB) in the PLC of industrial control system(ICS). There are very rare chances that industrial control system are connected directly to the internet and each PLC is configured with the unique properties. The attacker can gain the knowledge about the design documents of ICS with the help of Employee's (insider) of the company or they can gain the knowledge from the earlier version of Stuxnet.

Using zero-day liabilities or multiple zero-day vulnerabilities, stuxnet uses local area network (LAN) to spread across other computers a shown in Fig. 1.

It uses special method to load a .dll file by bypassing behavior blocking and host intrusion protection based technologies that monitor load library calls.

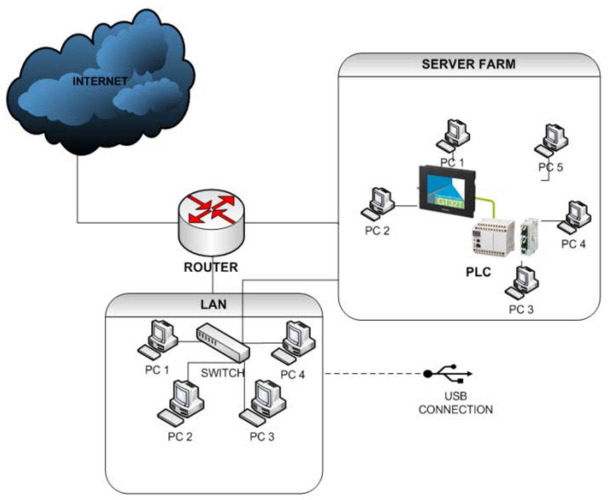


Fig. 1 Spreading of Stuxnet

Once stuxnet enters into intranet, it updates its definition with the use of download server [5]. If there is any previous version of stuxnet presence in intranet, new stuxnet stimulates it and spreads in PLC of ICS [6]. For the installation purpose, stuxnet verifies administrative privileges on the system. If it does not have already, it tries to attain privileges by using one of the two zero vulnerabilities. It verifies the detail configuration of the ICS for an appropriate target. Once stuxnet installs, it will gather information about negotiation, system etc and sends these details to the attacker/hacker via http.

3 Flame Malware

Flame is the most sophisticated computer malware ever seen by the industry[7]. Flame, also known as w32.Flame.skywiper and is designed to steal different databases. A distinct functionality is used by flame malware is "Audio Spying" that can record audio, screenshots and can monitor keyboard activities and network traffic. For example flame has capabilities of keeping the records of Skype conversation by detecting and recognizing a microphone on the infected computer.

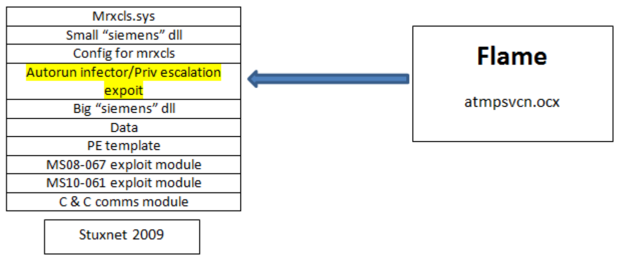
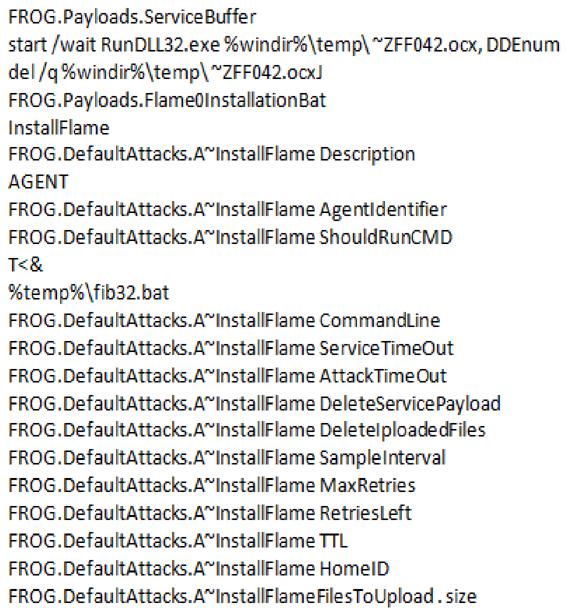


Fig. 2 Flame Auto Run

Flame malware can then transferred recorded information to the server which could be beginning point of the malware to respond [8]. It does have lot of similarities with other malware such as stuxnet and Duqu as shown in Fig. 2. Flame is 20 times bigger that stuxnet in terms of code and complexity. Like stuxnet, flame also uses local area network (LAN) or intranet or USB stick to spread in different systems. Unlike stuxnet, flame is not only used to affect Industrial Controller System, but also to individuals, educational institutes and businesses. Although stuxnet and flame are using different programming languages and application architectures, but features are common in terms of spreading, using similar securities, vulnerabilities, affecting system and also the use of hacking techniques/algorithms that are not used anywhere else.



```
FROG.Payloads.ServiceBuffer
start /wait RunDLL32.exe %windir%\temp\~ZFF042.ocx, DDEnum
del /q %windir%\temp\~ZFF042.ocxJ
FROG.Payloads.Flame0InstallationBat
InstallFlame
FROG.DefaultAttacks.A~InstallFlame Description
AGENT
FROG.DefaultAttacks.A~InstallFlame AgentIdentifier
FROG.DefaultAttacks.A~InstallFlame ShouldRunCMD
T<&
%temp%\fib32.bat
FROG.DefaultAttacks.A~InstallFlame CommandLine
FROG.DefaultAttacks.A~InstallFlame ServiceTimeOut
FROG.DefaultAttacks.A~InstallFlame AttackTimeOut
FROG.DefaultAttacks.A~InstallFlame DeleteServicePayload
FROG.DefaultAttacks.A~InstallFlame DeleteIploadedFiles
FROG.DefaultAttacks.A~InstallFlame SampleInterval
FROG.DefaultAttacks.A~InstallFlame MaxRetries
FROG.DefaultAttacks.A~InstallFlame RetriesLeft
FROG.DefaultAttacks.A~InstallFlame TTL
FROG.DefaultAttacks.A~InstallFlame HomeID
FROG.DefaultAttacks.A~InstallFlameFilesToUpload . size
```

Fig. 3 Flame Code

After analyses in detail, researchers identified similarities in functions of the flame and a version of stuxnet released in 2009 as shown in Fig. 3. Flame has capabilities to clean all the traces of viruses from the system using "kill" command. It has capacity to steal or alter electronic documents as well.

4 Obfuscation of MyDoom Malware

Obfuscation is a ubiquitous feature of present day malware. Because obfuscation hardens the complexity of a program for reverse engineering analysis and changes the signatures of the program it is a prevalent choice of malware authors. There are three common obfuscation techniques junk insertion, code reordering and packing. Packing is the dominant method. Malware authors today rely heavily on packers.

The percentage of new malware that is packed is on the rise; from 29% in 2003 to 35% in 2005 to almost 80% of current day malware is packed [9]. Packers work by compressing a program and then wrapping it with a decompression utility as a single executable code. To make it more complex, encryption is used. 1st solution that comes to mind for packer problem is to unpack the packed code, reverse engineer it and then do the malicious code analysis. However, it is hard to know and implement the defense for all packers and unpackers. As per a major security and antivirus company there exists at least 2000 variants of packers in more than 200 families of which they could identify the unpacker code in nearly 1200 packers spread among 150 families. Effectively, packer code was available for 800 members in 110 families only. Hence, there is still a backlog of 1200 members among 90 families [10]. This number increases day by day, as new packers are released and existing packer's code is modified to reuse.

Several variants of a malware are distributed using different packers resulting in different signatures every time a malware is out in the field. In following sections we will further explore this with use of **MyDoom** worm, also known as W32.MyDoom@mm, Novarg and Mimail.R. This is a mailing worm family spread via email and P2P networks. Once entered in to a computer system it opens the backdoors for other malicious codes allowing the attacker access to infected system. Some variants were also used for DDOS attacks. However, scope of this paper is not to explore the functionalities of MyDoom worm, rather being a very successful malware we have used three of its variants packed with different packers. When analyzed all three resulted into different signatures and analysis. MyDoom malware is analysed using three different signature techniques: Armadillo, InstallShield and UPX.

4.1 Malware packing and effectiveness

Given a malware variant M such as MyDoom processed through a packer P such as Armadillo, generates an obfuscated malicious code PM; this makes reverse engineering analysis and signature based detection more difficult.

$M \pm P = PM$

Where,

M	=	Malware
P	=	Packer
PM	=	Packed Malware

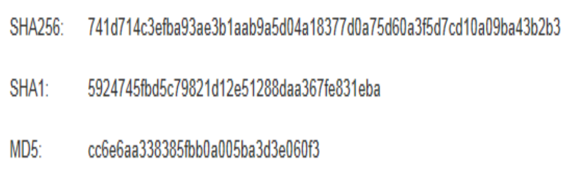
With this approach M gives a signature ‘A’ while PM Gives a Signature ‘B’.

If there are *n* variants of a malware M such as M*1*, M*2*, M*3*……….M*n* and each generates a signature *S* Then *n*remains constant and there will be *n*number of signatures such as *S*1, *S2*, *S3*……*Sn*. The main problem with signature based detection method is that it requires strict code analysis and manual intervention. These signatures can easily be bypassed as and when new signatures are created plus the increasing size of signature repository becomes an issue with growing number of malware being released in the wild everyday [9].

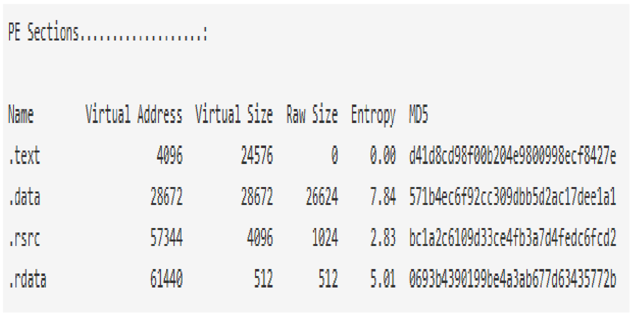
Before we proceed to test the effectiveness of packers we need to understand how packers work. Packers at a high level observation have four functional mechanisms compression, protection, encryption and bundling. Different packers use one or all of these methods where either a file is simply compressed to reduce the size with no concentration to avoid unpackingor it is both encrypted and obfuscated to prevent access to original file. Some use protection by combining both of the above mentioned methods and there exists bundles as well which are self-contained and package multiple files together as a single executable file.

Three examples we have used in this paper are well known. They are UPX, InstallShield and Armadillo. To better understand the functionality of a PE file and a packer we refer to figures from Scott and Mian [12] comparing a normal executable vs. a packed executable. A packer packs and secures the original code with unpacking code that is used at the time of execution. This is called stub code. Original code remains hidden and hence hiding its signatures and identity as the packed code will generate new signature. Analysis with My Doom Worm is as follows.

For further analysis our first MyDoom variant “Email-Worm.Win32.Mydoom.b” is packed with UPXFreak V0.1 -> HMX0101. When analyzed, we found following results.

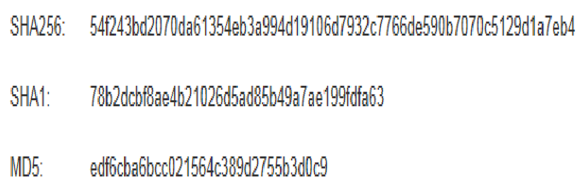


```
SHA256:    741d714c3efba93ae3b1aab9a5d04a18377d0a75d60a3f5d7cd10a09ba43b2b3

SHA1:      5924745fbd5c79821d12e51288daa367fe831eba

MD5:       cc6e6aa338385fbb0a005ba3d3e060f3
```

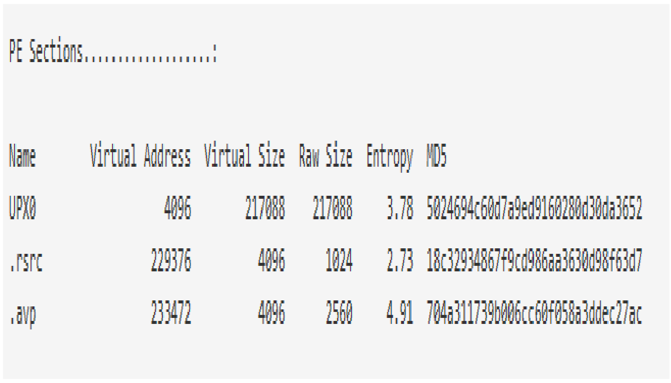


```
PE Sections.................:


Name       Virtual Address  Virtual Size  Raw Size  Entropy  MD5
.text                 4096         24576         0     0.00  d41d8cd98f00b204e9800998ecf8427e
.data                28672         28672     26624     7.84  571b4ec6f92cc309dbb5d2ac17dee1a1
.rsrc                57344          4096      1024     2.83  bc1a2c6109d33ce4fb3a7d4fedc6fcd2
.rdata               61440           512       512     5.01  0693b4390199be4a3ab677d63435772b
```

Where SHA is Secured Hash Algorithm, MD 5 is Message Digest. Second variant is packed with “InstallShield 2000” resulting as per following.

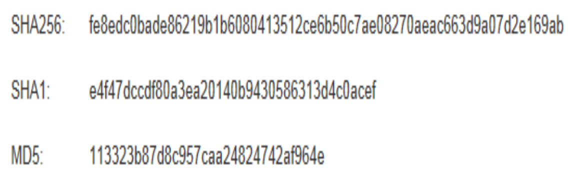


```
SHA256:    54f243bd2070da61354eb3a994d19106d7932c7766de590b7070c5129d1a7eb4

SHA1:      78b2dcbf8ae4b21026d5ad85b49a7ae199fdfa63

MD5:       edf6cba6bcc021564c389d2755b3d0c9
```

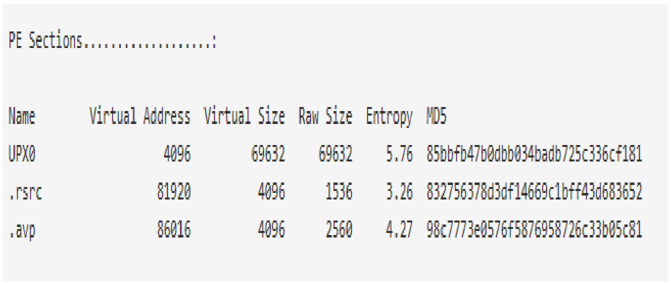


```
PE Sections.................:


Name       Virtual Address  Virtual Size  Raw Size  Entropy  MD5
UPX0                  4096        217088    217088     3.78  5024694c60d7a9ed9160280d30da3652
.rsrc               229376          4096      1024     2.73  18c32934867f9cd986aa3630d98f63d7
.avp                233472          4096      2560     4.91  704a311739b006cc60f058a3ddec27ac
```

Next variant is packed with “Armadillo V1.71” and analysis resulted as follows.



```
SHA256:    fe8edc0bade86219b1b6080413512ce6b50c7ae08270aeac663d9a07d2e169ab

SHA1:      e4f47dccdf80a3ea20140b9430586313d4c0acef

MD5:       113323b87d8c957caa24824742af964e
```



```
PE Sections.................:


Name       Virtual Address  Virtual Size  Raw Size  Entropy  MD5
UPX0                  4096         69632     69632     5.76  85bbfb47b0dbb034badb725c336cf181
.rsrc                81920          4096      1536     3.26  832756378d3df14669c1bff43d683652
.avp                 86016          4096      2560     4.27  98c7773e0576f5876958726c33b05c81
```

The examples shown above are for same malware packed with different packers. Correspondingly they all have resulted different signatures. This concludes that packers provide an effective camouflage to malware. Although benign software uses packers to compress the size of their code and to make their code more safe from cracking but their number in application is very less as compared to malware.

4.2 Packed malware solutions

Two broad categories of solution are in use for existing packer problem. (1) To manually reverse engineer the packed binaries and create unpackers for all known existing packers (2) To keep doing malware analysis and adding signatures of know malware and families to AV databases.

Both of these methods are not generic and need high level of reverse engineering and assembly language skills. They are also time consuming and does not stand tall against zero day malware. There is no effective generic unpacker and signature to the new and existing obfuscated malware. There are static and dynamic analysis approaches in use. Static analysis of a packed malware is safe, effective and portable but is not generic. It entails significant investment of time and efforts by highly skilled engineers. Looking at the history of malware growth this investment seems to be keep growing at a fast dynamic rate. On the other hand dynamic analysis on a VM or an emulator is less effective because new malware are smart enough to recognize the presence of an emulator or VM. They come loaded with anti-emulating techniques. Also there is no black and white line or a heuristic method to decide the benchmarks of dynamic analysis.

Now we are facing challenge of Zero day malware; plus old malware and packers never die they just get reinvented. In this situation we need an out of box thinking fight against obfuscated malware.

5 Conclusion

Malware code obfuscation is an effective method used by majority of malware authors to avoid detection from antivirus software and harden reverse engineering analysis to hide the true nature of their code from researchers. Because signature-based malware detection mechanism rely on byte code sequence, it is very easy for malware authors to change the byte code sequence and hence change the malware signature using tools like packers.

Our future work will be on analyzing stuxnet and flame malware using reverse engineering methods.

References:

[1] M. Combs, Impact of the Stuxnet Virus on Industrial Control Systems, *XIII International forum Modern information society formation - problems, perspectives, innovation approaches* 5 - 10 September, 2012, St.-Petersburg.

[2] M. Faisal, M. Ibrahim, STUXNET, DUQU and Beyond, International Journal of Science and Engineering Investigations, Vol. 1, Issue 2, March 2012, pp. 75 – 78.

[3] S. Sheng, W. Yingkun, L. Yuyi, L. Yong, J. Yu, Cyber attack impact on power system blackout, *IET Conference on Reliability of Transmission and Distribution Networks* (RTDN 2011), 22-24 Nov. 2011, pp. 1 – 5.

[4] P. Kerr, J. Rollins, C. Theohary, The Stuxnet Computer Worm: Harbinger of an Emerging Warfare Capability, *Congressional Research Service*, Dec 9, 2010, pp. 1-9.

[5] T. Miyachi, H. Narita, H. Yamada, H. Furuta, Myth and Reality on Control System Security Revealed by Stuxnet, *SICE Annual Conference*, 13-18 September, 2011, Tokyo, Japan, pp. 1537-1540.

[6] N. Falliere, L. Murchu, E. Chien, W32.Stuxnet Dossier, *Symantec Security Response*, pp. 1-68.

[7] http://www.securelist.com/en/blog/208193522/

[8] http://www.securelist.com/en/blog?weblogid=208193568

[9] K. Babar, F. Khalid, Generic Unpacking Techniques, *2nd International Conference on Computer, Control and Communication IC4*, 17-18 Feb. 2009, pp 1-6

[10] F. Guo, P. Ferrie, T. Chiueh, A Study of the Packer Problem and Its Solutions,*RAID '08 Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, pp. 98–115, Springer-Verlag Berlin, Heidelberg 2008.

[11] Vinod P. V. Laxmi,M.S.Gaur, Survey on Malware Detection Methods, *3rd Hackers' Workshop on Computer and Internet Security,* March 17-19, 2009, pp 74-79

[12] S. Treadwell, M. Zhou, A Heuristic Approach for Detection of Obfuscated Malware, *IEEE International Conference on Intelligence and Security Informatics*, 2009. ISI '09. 8-11 June 2009, pp 291-299