The new signature generation method based on an unpacking algorithm and procedure for a packer detection

Donghwi Shin[1], Chaetae Im[2], Hyuncheol Jeong[2], Seungjoo Kim[1], Dongho Won[1]
[1]*Sungkyunkwan University*
[2]*Korea Internet & Security Agency*
[1]*dhshin@security.re.kr,* [2]*chtim@kisa.or.kr,* [2]*hcjung@kisa.or.kr*
[3]*skim@security.re.kr,* [3]*dhwon@security.re.kr*

Abstract

Recently, a malware is growing rapidly and the number of malware applies various techniques to protect itself from the anti-virus solution detection. The reason of this phenomenon is that a longer resident on an infected host guarantees the more profit. As a result, these many protection techniques are applied to a malware, a representative of those is a Packing. It is not an exaggeration that most of the malware currently is distributed. In other words, a packer is widely used for a malware protection. Therefore analysts must determine whether the malware was packed or not and if the malware is packed, what packer is used, before an analysis of the malware. For these procedures, some packer detection tools were released and used. But, the detection performance is not good and there is some false positive and false negative. Therefore we propose a signature generation method that is based on an unpacking process and algorithm in this paper. And we offer the packer detection experiment result using the proposed packer detection signature generation method.

Keywords: *Packer, Unpacking, MUP(Manual Unpacking), Malware*

1. Introduction

In the past, the malware has grown in the quantitative and in the qualitative. The quantitative growth is presented by the chart of the AV-Test.org "New Unique Samples Added to AV-Test.org's Malware Collection".

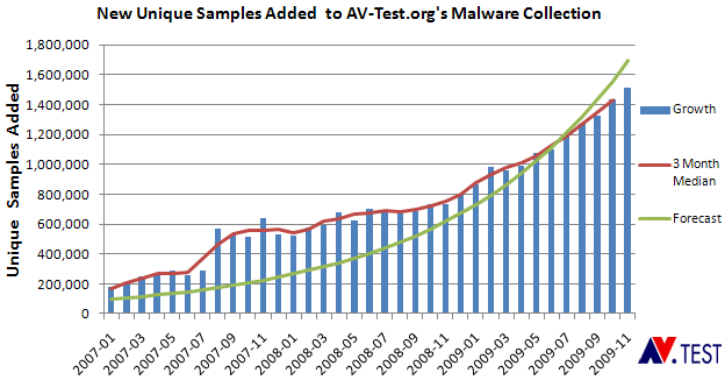

Figure 1. AV-Test "New Unique Samples Added"

These increases in a quantitative make harder analyze the malware. In addition, the growth in qualitative is becoming more difficult to analyze the malware. These growth in qualitative

can be identified by various analysis interference techniques. Maybe, the most commonly used interference techniques are "Packing", "Anti-Debugging", "Anti-VM", and "Rootkit". The progress of these techniques makes harder the malware in a short time.

The representative technique is a "Packing" among many analysis interference techniques. The packer gets a binary as an input and transforms or compresses the binary. As result, the packer creates a different type binary. Also various protection features that the packer offers protects the original or packed binary. Currently the most of malware is being deployed as a packed binary. Therefore an analyst must to determine whether the target binary is packed or not and if the target binary is packed, what packer is used.

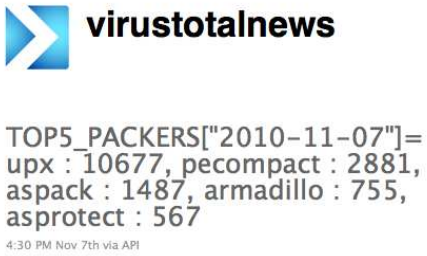

Figure 2. VirusTotalNews Twitter Packer Top5

To determine whether a binary is packed or not, we use some tools(ex. PEiD, DiE) or a message that some debugger(ex. Ollydbg, IDAPro) shows when open the binary. But the use cannot know what packer is used through a debugger. So many analyst may use the packer detection tools because they must determine both whether it is packed or not and what packer is used. The representative tools are aforementioned PEiD and DiE. These tools use the signature-based detection technique to detect the packer. However the packer detection signature is not updated frequently and so many packers is newly appeared. So the increasing number of packer does not detected by the older signature. And there are some false-negative and false-positive by the signature.

In this paper, we propose the new packer detection signature generation method. The signature generation concept is that after the unpacking algorithm is executed, the packed code is unpacked and the original code is on the memory. In the section 2, we describe some popular packer and in the section 3 some popular packer detection tools and signature generation method of the PEiD, in the section 4 the proposed signature generation method, and an experiment result, and in the section 5 we conclude.

2. Packer

The most widely used and most simple method is a packing for a protection malware. After writing an arbitrary code, we compile and link that. Then we get the binary equivalent to the code. If a debugger opens the binary, we can see the content and algorithm in an assembly language. But a malware wants to reside on the infected host without detecting itself by the anti-virus solution. So a malware author transforms the binary through some packers. As result, the packed file looks like a following figure.

If the packed binary is executed, the unpacking algorithm in the packed binary unpacks the packed binary. Then we can see the unpacked code on the process memory area in the original. And it execute the unpacked code from the OEP(Original Entry Point). So the packed binary eventually is unpacked and reveals the original code.

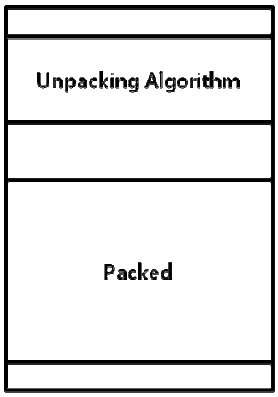

Figure 3. Packed Bianry Structure

Among some packers, the most widely used packer is the UPX, ASPack, Themida, and so on. We can see that what packer is the most widely used and how many used from the VirusTotal. In this section, we describe characteristics of some packers and an unpacking method according to the statistics of the most widely used packer.

2.1. UPX

The UPX(Ultimate Packer for eXecutables) was released in March 1998. That is the first beta version. And then recently the version 3.07 was released in September 2010, the UPX is created by the Markus Oberhumer and Laszlo Molnar. And that is distributed in GPL(General Public License). The UPX offers the more high compression ratio than the Winzip or GZIP. And the decompress speed is faster than the others compression applications. The compression speed of the UPX is about 10MB/sec on the Pentium 133 and about 200MB/sec on the Athlon 2000. Also the UPX supports many file formats and various platforms.

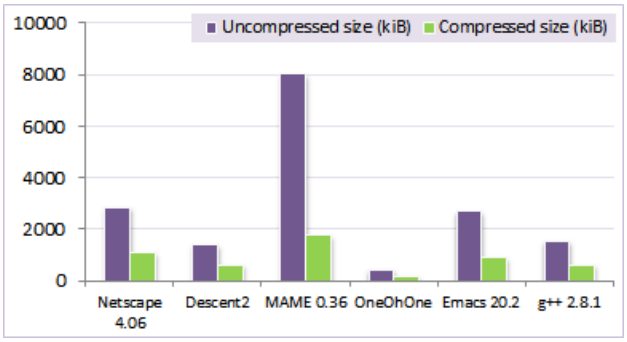

Figure 4. UPX Compression Ratio

As explained earlier, the UPX compression ratio is superior other applications and is a commonly used algorithm. However, the UPX is already a widely known packing algorithm so, the packed binary as the UPX is able to unpack.

From now on, we describe how can unpack a binary that is packed by the UPX. This unpacking procedure will be used to explain the new signature generation method later. The method for unpacking the packed file with the UPX is divided into two cases. First, it is a method using some tools. This uses the UPX packing tool for unpacking the packed file. It is so simple and anyone can do it. Second, there is a method to manually unpacking. Once the user open the packed binary on a debugger(ex. Ollydbg, IDAPro). And dump the process

memory area and recover the IAT(Import Address Table) after some procedure. Here, we describe how to manually unpack the packed binary with the UPX.

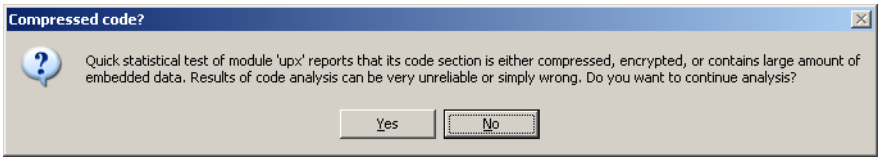

Figure 5. Ollydbg Messagebox when the packed binary opened

After the "Messagebox" in the debugger, "pushad" opcode can be found in the debugger. And we can see the "popad" opcode after the binary is executed. Then, we can see the jump opcode that jump to any address under the "popad" opcode. The target address of the jump opcode is the entrypoint, OEP(Original Entry Point), was set when the binary is compiled and created at first time. After running up here, the original code will appear in the process memory area. Now, if we just dump the process memory area and recover the IAT, we get the original file before packing.

2.2. ASPack/ASProtect/ASProtect SKE

The ASPack/ASProtect/ASProtect SKE is a commercial packer and is released by the "StarForce Technologies" company. The ASPack is the simplest packer among these packers is distributed by the StarForce Technologies. Until now, the ASPack 2.21 version is released. This provides the packing feature on the EXE, DLL, OCX. And its compression ratio is a maximum 70 percent. In addition, the ASPack supports a long file name and is superior to competition products in decompression routine thought the StarForce Technology web page. Finally, the ASPack supports various Microsoft Windows Platform (Microsoft Windows 95/NT 4.0/98/ME/2000/XP/2003/VISTA). The StarFroce Techologies describes that the ASPack offers about 40~70% compression ratio and supports various compile. As following.

- Microsoft Visual C++
- Visual Basic
- Inprise(Borland) Delphi
- C++ Build
- ...

The unpacking method that unpacks the packed binary with the ASPack is known well. If you search on the Google, a lot of tutorial document and video can be found. Briefly describe how to manual unpacking, which is similar to the UPX manual unpacking method. Like the UPX, if the packed file with the ASPack is opened with a debugger, the "pushad" opcode is appeared. And then go along the code so can meet the "push + retn" opcode sequence. The "push + retn" means that jump to any place. This is so similar to the jump code after the "popad" opcode in the packed binary with the UPX. It is a simple trick. Therefore execute the binary until the previous step of "push + retn" and if execute the "push + retn", then jump to any place. The place is a OEP. Likewise with the UPX, execute the binary until the OEP, the packed binary with the ASPack is unpacked completely.

Next, the ASProtect is distributed through the "StarForce Technologies" and is a commercial packer. The recent version of the ASProtect is version 1.5. The ASProtect

supports the executable file compression and various Microsoft Windows platforms. In addition, it supports an encryption with a compression and some protection features. The example is may be a process memory dump protection using a "ProcDump". And then the following features are supported by the ASProtect. The ASPack does not offer these features. Especially, the ASProtect uses the cryptographic algorithm and generates a registration key using the host specific information.

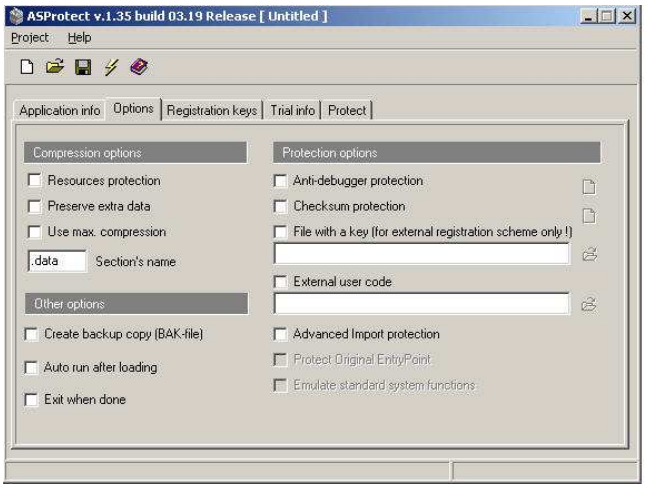

Figure 6. ASProtection Features

- Application Integrity Check
- Counteraction to dumping application memory
- Counteraction to debugger and disassemble
- Counteraction to memory patching

From now, we describe the ASProtect unpacking method. There are various ASProtect unpacking methods. We describe the simplest method in this paper. Once, disable all exceptions that the Ollydbg support and run the binary. So you can see the "push + retn" opcode sequence. As mentioned in the ASPack, the "push + retn " means the jump opcode. After the "push + retn" opcode is executed, there is the "pushad" opcode. Here, using the "ESP Trick" we arrive the jump opcode and these jump target address is the OEP. Because I found the OEP, dump the process memory and recover the IAT then, we get the original binary. Of course, there are another method using exceptions. If you wonder this method, you refer to the another reference document.

The ASProtect SKE provides greater security features than the aforementioned ASProtect and version 2.58 is released. The ASPrortect SKE offers various protection features with the packing feature.

- Anti-Debugging
- Checksum Protection
- Import Function Protection
- Password to run
- Activation Key

- Expire

Among these features, some features commonly used by the malware are the "Anti-Debugging" and "Checksum Protection".

Relatively it is so difficult to unpack the ASPrortect SKE than the ASPack and the UPX. This reason is various features that the ASPrortect SKE offers. But there are various tutorial(document and video) for unpacking the ASProtect SKE. Simply look at the process of unpacking the ASProtect SKE but the process is not simple, we find the OEP using a method that uses the number of exceptions. This method is similar to the ASProtect unpacking method. After finding the OEP, dump the process memory area and recover the IAT. So we get the original binary. It is the ASProtect SKE unpacking process. In the section, we describe the ASProtect SKE unpacking method simplely. It is beyond the scope of this paper to describe more about the method. Refer to any other tutorial.

2.3. Armadillo

The Armadillo is a commercial application that is distributed by the "The Silicon Realms Toolworks". The official name of Armadillo is the "SoftwarePassport v8" and the current version 8 has been released. And the Armadillo is not a tool for packing only. So the Armadillo offers many features without the packking. The SoftwarePassport v8 are the four kinds of packages. These are the "Microsoft Windows 32bit", "Microsoft Windows 64bit", "Premium Bundle", "Coporate Edition". In this section, the basis of description is the "Coporate Edition". The Digital River web site provides the trial version of SoftwarePassport v8.

The Armadillo offers more protection features than the other packers was described. Protection feature are as follows.

- Hardware Locking, Fingerprinting
- Pass-Along Protection
- Compression
- License Key Delivery
- Primary and Secondary File Protection
- File Types Protected

Brief description of aforementioned proetection features, the hardware locking control some events using a hardware based information such as BIOS, HDD, CPU, MAC address and so on. Among these Armadillo protection features, the primarily features used by the malware may be "Memory Dumping" and "ArmAccess.DLL". The malware protect itself from dumping the process memory area through the "Memory Dumping". And the "ArmAccess.DLL" modifies an address of both a loaded dll file and the called API. The modified address is different from the typical address. As result, it makes harder identify that the any DLL is loaded or any API is called. Also the Armadillo protects itself from an unpacker or a stripper. Explaining the exact feature is outside the scope of this paper. So refer to the document provided by the Silicon Realms Toolworks.

The Armadillo unpacking process is so difficult. Because the Armadillo offers so many protection features. Therefore the manual unpacking of Armadillo is difficult. Though we find

the OEP and dump a process memory, the IAT recovery process is not easy. So there are many unpacking tools for the unpacking Armadillo. However, there are many tools depending on the version of Armadillo. So use these tools after determine the version and applied features of Armadillo.

- Arama.Intruder.0.4
- Armadillo Find Protectd v1
- Armadillo Killer 2
- Armadillo Reducer 1.7
- Armadillo.DLL&OCX
- Armadillo.Sections.Stripper 1.22
- Armadillo_Key_Generator 1
- ArmadilloCleaner
- ArmadilloTools v1.2
- ArmaGeddon v1.1.0 by Condzero

2.4. PE Compact

The Bitsum Techology distirbutes the PECompact tool. This tool is a packer and supports various file type, EXE, DLL, OCX and so on. The PECompact has a superior performane on the compression rate and supports from the Microsoft Windows 95 to 7. Also it supports a 32 bits and 64 bits.

The PECompact supports various features by many subsystems. These are check by the GUI version screenshot and command & parameters of console application.

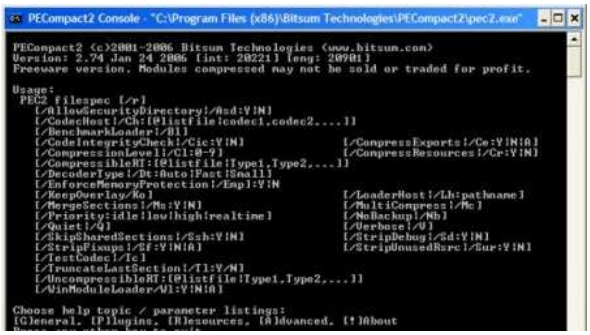

Figure 7. PECompact Console Parameters

Also the user can develop the necessary features using the SDK beyond the supported features. These features, that the user develops, are available in form of a plug-in for the PECompact. The Bitsum Technology web site currently provides some of the plug-in. A notable of these plug-in is an "API Hooking" plug-in. This plug-in allocates APIs memory address dynamically and especially supports separates feature for APIs associated with "Anti-Debugging".

The unpacking procedure about a packed binary by the PECompact is simpler than the unpacking procedure about the Armadillo. The user can unpack the PECompact packed

binary easily like the UPX if you refer to the PECompact unpacking document. Also the PECompact unpacking tools are many. But the user must select the appropriate tool depending the PECompact version and features. These PECompact unpacking tools are distributed as follows:

- PECompact 2.xx Unpacker v0.2 OllyScript
- UnPECopact

2.5. Themida

The Themida is released by the "Oreans" and a current version is 2.1.6.0. And we can get the demo version from the Oreans website. The Themida offers many various features and protect a binary using a special method that differs from the other packers. The Themida applies a protection feature to each code block that is divided from the decompiled binary. But the other packers apply a protection feature to a whole binary.

Among packers at the VirusTotal statistics, the Themida may offer so many protection features. The following list is the Themida supported features.

- Anti-debugger techniques that detect/fool any kind of debugger
- Anti-memory dumpers techniques for any Ring3 and Ring0 dumpers
- Different encryption algorithms and keys in each protected application
- Virtual Machine emulation in specific blocks of code
- Multiple polymorphic layers with more than 50.000 permutations
- Random garbage code insertion between real instructions
- Anti-Memory patching and CRC techniques in target application
- Advanced Entry point protection

Figure 8. Thermida Protection Features

As shown above, various feature of the Themida can find out through the Themida options panel. So, because of many protection features, it is a difficult thing to unpack the packed binary with the Themida. And we use another unpacking method for unpacking the Themida per an applied protection feature. Also the Themida unpacking tool is a little.

- OKDODO's Themida Unpacker

- UnThemida

3. Packer Detection Tools

The malware analyst must determine whether the malware is packed or not because so many malwares is packed and distributed. There are some packer detection tools. The representative tools are the PEiD and DiE. These tools use a signature-base packer detection method. So there is a little the false-negative and false-positive rate. In this section, we simply describe some packer detection tools and a signature generation method .

3.1. PEiD

The current version of PEiD is 0.95 and is not updated recently. The distribution website of the PEiD releases the nPEiD for detecting the packer in network streams. Maybe the PEiD is the most well-known packer detection tool. It detects the packer using a signature. We can get the signature from the "External Signatures" of PEiD forum or "Neil's Plugins" website. And the PEiD offers a plug-in feature. The most typical features are an unpacking, a CRC checking. Among these plug-in, there is a signature update plug-in from the Neil's Plugins website. This plug-in offers a signature generation method and the user can add a signature manually.

We can know how to generate the packer detection signature of PEiD from the pefile project. This project is developed in the Python and offers a signature generation feature for the PEiD. The following code is a signature generation code for the PEiD. Depending on this code, we can know and draw a signature area in the PE file. The following figure shows a signature area in the PE file.

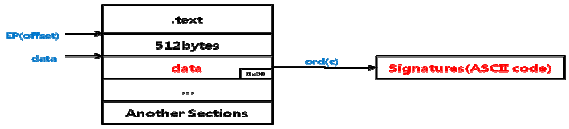

Figure 9. PEiD Signature on PE file base on Entrypoint

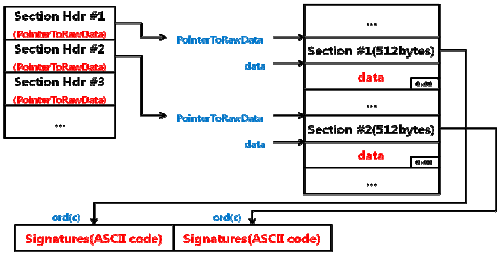

Figure 10. PEiD Signature on PE file base on Section

Because the PEiD uses a signature-based detection method and the signature is not updated frequently, both the false-negative and the false-positive rate are not low.

3.2. DiE

The DiE is the abbreviation of "Detect It Easy". This tool is implemented in the Russia and the current version is 0.64. This tool has long since been update. Therefore there are a little

false-negative and false-positive likewise the PEiD. The DiE offers more information as a PE file analysis tool. For example, it is a PE section, import functions, disassembles result, and so on. Also the DiE supports a plug-in feature. This plug-in feature is similar to the plug-in of PEiD. So a plug-in feature of DiE is almost equal to the plug-in of PEiD. The packer detection method of DiE is not known exactly. But, maybe the method is similar to the method that is used by the PEiD. That is the DiE uses a signature-base detection method.

3.3. ExeInfoPE

The ExeInfoPE offers PE file analysis features. The packer detection feature is a part among these features. The current version is 0.0.2.8 and is updated at December 2010. The characteristic of ExeInfoPe is that it detects what a packer is used and it tell you where you that you can unpacking it by the “Lamer Info”. But, the “Lamer Info” site is not available now.

3.4. ProtectionID

The ProtectionID is known that it detects and tells about CD lock information. And it offers so many features. A packer detection features is a part among many supported features. The ProtectionID can detect so many packers and can detect exactly. The log panel show the packer detection result about a target binary. But, we do not know yet how to detect the packer.

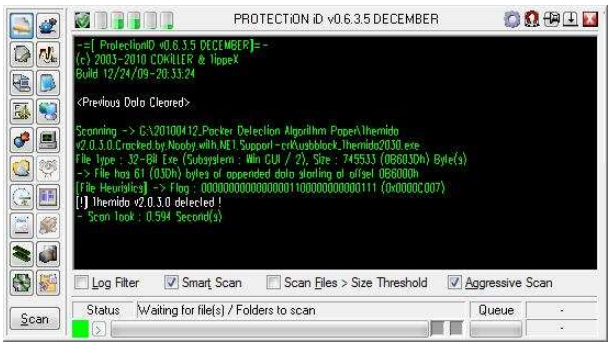

Figure 11. ProtectionID Log Panel

There is a method using the PEiD signature for packer detection. This interlocking feature is not enabling at default. If you want to use the PEiD signature, you set the interlocking feature in a configuration panel.

3.5. RDG Packer Detector

RDG Packer Detector is developed by the RDGMax and the current version is 0.6.6. This tool offers a file checksum calculation, an entropy calculation, and so on with a packer detection feature. And we can update a signature from the web and generate a signature manually. Also its special feature is that what find the OEP. Though the OEP is distortive, it can find the OEP.

3.6. FastScanner

The FastScanner was created by a group "AT4RE"(Arab Team 4 Reverse Engineering). in 7. 2010, the version 3.0 has been release. The FastScanner is developed in an assembly language for high speed packer detection. The FastScanner offers a signature update from the web. But, it is not sure how to generate a packer detection signature.

Figure 12. FastScanner Update Option

4. Packer Detection Tool Experiment

In this section, we use some packer detection tools and measure the detection, false-negative, and false-positive rate. The experiment content is the packer name and version. This experiment target tools are “PEiD”, “DiE”, “ExeInfoPE”, “ProtectionID”, “RDG Packer Detector”, “FastScanner”. And the experiment samples are packed with 11 packers.

4.1. The packer name detection

The packer name detection means that some packer detection tools detect an exact packer name. So if a packer detection tool can detect the kind of packer, the packer name is correct in the detection result. The following table is the detection result. In the result, almost packer detection tools can detect the kind of packer exactly. But, the detection rate of PEiD is 52% and the false-negative rate of PEiD is 58%.

Table 1. Packer Type Detection

Packer	PEiD v0.95	DIE v0.64	Exeinfope v0.0.2.6	ProtectionID v0.6.3.5
UPX3.03	2(1)	3(0)	3(0)	3(0)
NsPack2.3	0(3)	3(0)	3(0)	3(0)
NsPack3.4	0(3)	3(0)	3(0)	3(0)
FSG1.33	3(0)	3(0)	3(0)	3(0)
FSG2.0	3(0)	3(0)	3(0)	3(0)
AsPack2.12	3(0)	3(0)	3(0)	3(0)
Themida2.0.3.0	0(3)	3(0)	3(0)	3(0)

Themida1.9.9.0	0(3)	3(0)	3(0)	3(0)
Armadillo 5.42	1(2)	3(0)	3(0)	3(0)
ASProtect SKE 2.1	3(0)	3(0)	3(0)	3(0)
ASProtect SKE 2.4	2(1)	3(0)	3(0)	3(0)
Detection Ratio (FalseNegative)(%)	52(48)	100(0)	100(0)	100(0)

4.2. The packer version detection

The packer version detection rate means that some packer detection tools detect an exact packer version. So if a packer detection tool can detection the packer version, the packer version is correct in the detection result. The following table is the detection result. In the result, the detection rate of almost packer detection tools is lower than the packer name detection rate.

Table 2. Packer Version Detection

Packer	PEiD v0.95	DIE v0.64	Exeinfope v0.0.2.6	ProtectionID v0.6.3.5
UPX3.03	0-0(3)	3-0(0)	2-1(0)	3-0(0)
NsPack2.3	0-0(3)	0-3(0)	3-0(0)	0-3(0)
NsPack3.4	0-0(3)	0-3(0)	0-3(0)	3-0(0)
FSG1.33	3-0(0)	3-0(0)	3-0(0)	0-3(0)
FSG2.0	3-0(0)	3-0(0)	3-0(0)	0-3(0)
AsPack2.12	3-0(0)	3-0(0)	3-0(0)	3-0(0)
Themida2.0.3.0	0-0(3)	0-0(3)	0-3(0)	2-1(0)
Themida1.9.9.0	0-0(3)	0-0(3)	0-3(0)	0-3(0)
Armadillo 5.42	0-0(3)	0-0(3)	0-0(3)	0-3(0)
ASProtect SKE 2.1	3-0(0)	0-3(0)	0-3(0)	0-3(0)
ASProtect SKE 2.4	0-0(3)	0-0(3)	0-3(0)	0-3(0)
Detection Ratio -FalsePositive Ratio (FalseNegative)(%)	36-0 (64)	36-27 (37)	42-39 (19)	33-67 (0)

5. The new packer detection signature

5.1. The signature generation method

The basic feature of packer is a packing using the own algorithm for transforming a binary. In other words, each packer has its own packing algorithm. For a description, let some definitions. Let's call "Packer A 1.0" that a packer name is "A" and a packer version is "1.0". And there is two original executables, "P" and "Q". The packed executables "X" and "Y" are packed with the "Packer A v1.0".

Eq1. on X : X = Packer A v1.0(P)

Eq2 on Y : Y = Packer A v1.0(Q)

As aforementioned, a packer has its own algorithm. So, the packed binary X, Y has a characteristic of the " Packer A v1.0" . Therefore both " X" and " Y" have a unpacking algorithm, of " Packer A v1.0" and these algorithms in the " X" and " Y" almost the same.

Unpacking_Algorithm on X ≒ Unpacking_Algorithm on Y

Eventually, the opcode sequences of the " Packer A ver1.0" unpacking algorithm is almost same about both " X" and " Y" . Therefore we take the unpacking algorithm area as a packer detection signature. This is due to the similarity of unpacking algorithm in case of the same version packer. The following equation describes the packer detection signature generation method.

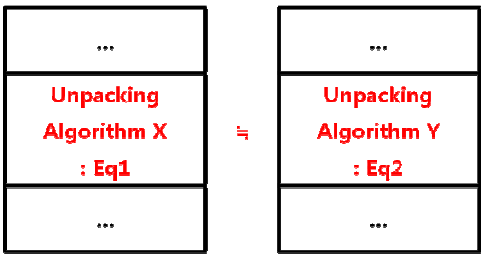

Figure 13. Unpacking Algorithm on the Binary

In case of " X"

RangeX = DetectUnpackingAlgorithmArea(X)

SequenceX = ExtractUnpackingAlgorithm(RangeX)

OP.SequenceX = ExtractOpcode(SequenceX)

SignatureX = LEFT(hex(OP.SequenceX), 2) // First 2bytes of the hex(OP.SequenceX)

In case of " Y"

RangeY = DetectUnpackingAlgorithmArea(Y)

SequenceY = ExtractUnpackingAlgorithm(RangeY)

OP.SequenceY = ExtractOpcode(SequenceY)

SignatureY = LEFT(hex(OP.SequenceY), 2) // First 2bytes of the hex(OP.SequenceY)

SignatureX ≒ SignatureY

So in theory, we can generate a detection signature and detect any packer that match to the generated signature. Now, we verify the signature using the UPX sample.

Once we open the packed bianry with the UPX on the Ollydbg. Then we can see "pushad"

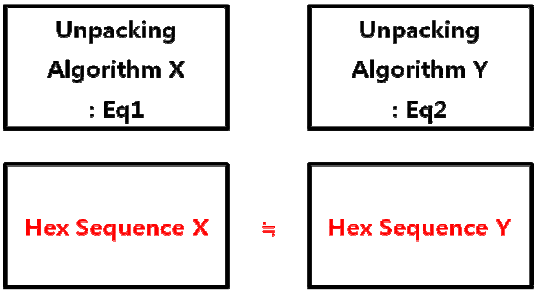

Figure 14. Signature Generation Method

Figure 15. Signature Area in the Ollydbg(UPX)

opcode. And we can see "popad" and "jump" opcode in the end of unpacking algorithm.

We extract assembly commands from "pushad" to "jump" for generating a packer detection signature. And then we extract opcode sequence from the extracted assembly command. Finally we extract left 2bytes from the opcode sequence.

5.2. The detection signature experiment

In this section, we experiment the new packer detection signature likewise section 4. This experiment target tools are “PEiD”, “DiE”, “ExeInfoPE”, “ProtectionID”, “RDG Packer Detector”, “FastScanner”. And these samples are the same in the section 4. These are packed with 11 packers. And in the following description, we describe the packer detection process with the proposed packer detection signature about Themida 2.0.3.0 and UPX 3.03. Finally we show the total experiment result.

First, we extract the signature from the three binary that is packed using the UPX 3.03. The result is the following table.

Table 3. Hex Sequence of UPX

Sample1(303)	Sample2(303)	Sample3(303)
60	**60**	**60**
BE 00E04300	BE 00000101	BE 00204200
8DBE 0030FCFF	8DBE 0010FFFF	8DBE 00F0FDFF
57	**57**	**57**
83CD FF	83CD FF	EB 0B
EB 10	EB 10	90

90	**90**	**8A06**
90	90	**46**
90	90	**8807**
90	90	**47**
90	90	**01DB**
90	90	**75 07**
8A06	**8A06**	**8B1E**
46	**46**	**83EE FC**
8807	**8807**	**11DB**
47	**47**	**72 ED**
01DB	**01DB**	**B8 01000000**
75 07	**75 07**	**01DB**
8B1E	**8B1E**	75 07
83EE FC	**83EE FC**	8B1E
11DB	**11DB**	83EE FC
72 ED	**72 ED**	11DB
B8 01000000	**B8 01000000**	11C0
01DB	**01DB**	01DB

From this table, we extract the first 2bytes and compare with each other.

Table 4. The extracted 2byte sequence of UPX

2bytes(Sample1)	2bytes(Sample2)	2bytes(Sample3)
60	**60**	**60**
BE	**BE**	**BE**
8D	**8D**	**8D**
57	**57**	**57**
83	**83**	**EB**
EB	**EB**	90
90	90	**8A**
90	90	**46**
90	90	**88**
90	90	**47**
90	90	**01**
90	90	**75**
8A	**8A**	**8B**
46	**46**	**83**
88	**88**	**11**
47	**47**	**72**
01	**01**	**B8**
75	**75**	**01**
8B	**8B**	75
83	**83**	8B
11	**11**	83

72	**72**	11
B8	**B8**	11
01	**01**	01

Then, we can see that the extracted 2bytes sequence almost is similar with each other. Among three samples, the hex sequence of first and second sample is exactly same. And In the sample 1 and 3, there is 83% consistent.

Second, we extract the signature from the three binary that is packed using the Themida 2.0.3.0. And we extract the first 2bytes again. The result is the following table. We can see that sequence is almost same.

Table 5. Hex Sequence of Themida

Sample1(2030)	Sample2(2030)	Sample3(2030)
B8 00000000	**B8 00000000**	**B8 00000000**
60	**60**	**60**
0BC0	**0BC0**	**0BC0**
74 68	**74 68**	**74 68**
E8 00000000	**E8 00000000**	**E8 00000000**
58	**58**	**58**
05 53000000	**05 53000000**	**05 53000000**
8038 E9	**8038 E9**	**8038 E9**
75 13	**75 13**	**75 13**
61	**61**	**61**
EB 45	**EB 45**	**EB 45**
DB 2D 37 E0	DB 2D 37 00	DB 2D 37 50 43
FF	05	FF
FF	01	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
FF	FF	FF
3D	FF	**3D**
40	FF	**40**
E8 00000000	**3D**	**E8 00000000**
58	**40**	58
25 00F0FFFF	**E8 00000000**	25 00F0FFFF

Table 6. The extracted 2byte sequence of Themida

2bytes(Sample1)	2bytes(Sample2)	2bytes(Sample3)
B8	**B8**	**B8**

60	**60**	**60**
0B	**0B**	**0B**
74	**74**	**74**
E8	**E8**	**E8**
58	**58**	**58**
05	**05**	**05**
80	**80**	**80**
75	**75**	**75**
61	**61**	**61**
EB	**EB**	**EB**
DB	**DB**	**DB**
FF	05	**FF**
FF	01	**FF**
FF	FF	**FF**
FF	FF	**FF**
FF	FF	**FF**
FF	FF	**FF**
FF	FF	**FF**
FF	FF	**FF**
3D	FF	**3D**
40	FF	**40**
E8	**3D**	**E8**
58	**40**	**58**
25	**E8**	**25**

In the Themida case, the hex sequence of first and third sample is exactly same. These characteristic appear in the same version packer. In the different version case, there a little difference between each version.

Therefore we can detect a packer name. And we can identify the packer version with a difference between different versions. The next table show a packer detection result using the new packer detection signature.

Table 7. Proposed Detection Signature Efficiency

Packer	Efficiency
UPX3.03	Enable(100%)
NsPack2.3	Enable(100%)
NsPack3.4	Enable(100%)
FSG1.33	Enable(100%)
FSG2.0	Enable(100%)
AsPack2.12	Enable(100%)
Themida2.0.3.0	Enable(100%)
Themida1.9.9.0	Enable(100%)
Armadillo 5.42	Enable(100%)
ASProtect 2.1	Enable(100%)
ASProtect 2.4	Enable(100%)

6. Conclusion

A malware analyst must determine whether a target malware is packed or not, because many malwares use a packing technique. So they use some packer detection tools for detecting a packer. Until now, a packer detection method is a signature-based detection method. However each detection tool uses different detection methods and signatures. Also because the false-negative and false-positive rate is not low, the rate cannot be ignored. Therefore we propose the new signature generation method for detecting a packer. This signature generation method is based on the own unpacking algorithm of packer. Using the new signature, we improve the packer detection rate and can identify some subtle differences on each version. In the future, packers will continue to evolve and the difference by the evolvement will be large or not. Therefore it will be more efficient to use a signature generation method depending on the behavior of the packer.

7. Acknowledgement

This work was supported by the IT R&D program of MKE/KEIT. [10035427, The Development of Automatic Analysis and Malicious Site Detection Technology Against Malware]

8. References

[1] PEiD, http://www.peid.info/

[2] Detect It Easy, http://hellspawn.nm.ru/

[3] Exeinfo PE, http://www.exeinfo.cjb.net/

[4] Protection ID, http://pid.gamecopyworld.com/

[5] RDG Packer Detector, http://www.rdgsoft.8k.com/

[6] FastScanner, http://www.at4re.com/

[7]Neil's Collection of Packer Signatures, http://www.peid.info/BobSoft/

[8] UPX, http://upx.sourceforge.net/

[9] NsPack, http://download.cnet.com/Nspack/3000-2250_4-10403507.html

[10] FSG, http://www.woodmann.com

[11] AsPack, http://www.aspack.com/

[12] Themida, http://www.oreans.com/themida.php

[13] Armadillo, http://www.siliconrealms.com/

[14] ASProtect SKE, http://www.aspack.com/

[15] PECompact, http://www.bitsum.com/pecompact.php

[16] M. Zubair Shafiq, PE-Probe: Leveraging Packer Detection and Structural Information to Detection Malicious Portable Executables, VB2009, 29-33, 2009

[17] M. Zubair Shafiq, PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime, RAID2009, Volume 5758/2009, 121-141, 2009

[18] Gaith Taha, Counterattacking the packers, McAfee, 2007

Authors

Donghwi Shin received his BS in physics from Sungkyunkwan University, Korea, in 2002, and his MS in computer science from Sungkyunkwan University, Korea, in, 2008. He is currently enrolled in a PhD course for information security in Sungkyunkwan University. He is a senior security consultant at SAMSUNG SDS. His research interests include malware & botnet analysis, reverse engineering, and exploit development.

ChaeTae Im received the B.S. degree in Computer Science from Chungnam National University, in February 20006, and the M.S. degree in Computer Science from Pohang University of Science and Technology, in 2003, He is currently working as Senior Research Associate of Korea Internet & Security Agency. His research interests include BotNet, Malware, Security of VoIP.

Hyunchoel Jeong received the B.S. degree in Statistical Computing from University of Seoul, in February 1996, and the M.S. degree in Computer Science from Kwangwoon University, in August 1998, He is currently working as Director of Korea Internet & Security Agency. His research interests include network security.

Seungjoo Kim received the BS, MS, and Phd in information engineering from Sungkyunkwan University, Korea, in 1994, 1996, and 1999, respectively. Prior to joining the faculty at Sungkyunkwan University in 2004, he was the director of the cryptographic technology team and the IT security evaluation team of Korea Internet and Security Agency for five years. Currently, he is an associate professor of the School of Information and Communication Engineering at Sungkyunkwan University. His research interests include cryptography, information security, information assurance, and digital forensics.

Dongho Won received his BE, ME, and PhD from Sungkyunkwan University in 1976, 1978, and 1988, respectively. After working at ETRI from 1978 to 1980, he joined Sungkyunkwan University in 1982, where he is currently a professor in the School of Information and Communication Engineering. His interests are in cryptology and information security. In 2002, he was the president of the Korea Institute of Information Security & Cryptology.