PolyPack:
An Automated Online Packing Service for Optimal Antivirus Evasion

Jon Oberheide, Michael Bailey, Farnam Jahanian
Electrical Engineering and Computer Science Department
University of Michigan, Ann Arbor, MI 48109
{jonojono, mibailey, farnam}@umich.edu

Abstract

Packers have long been a valuable tool in the toolbox of offensive users for evading the detection capabilities of signature-based antivirus engines. However, selecting the packer that results in the most effective evasion of antivirus engines may not be a trivial task due to diversity in the capabilities of both antivirus and packers. In this paper, we propose the creation of an online automated service, called PolyPack, that uses an array of packers and antivirus engines as a feedback mechanism to select the packer that will result in the optimal evasion of the antivirus engines. Towards understanding the utility and efficacy of such a service, we construct an implementation of PolyPack which employs 10 packers and 10 popular antivirus engines. We show that PolyPack provides 258% more effective evasion of antivirus engines than using an average packer and out-evades the best evaluated packer (Themida) for over 40% of the binary samples.

1 Introduction

PolyPack is a web-based service that employs an array of packers and antivirus engines to pack an input binary with the packer that results in the maximal evasion of detection by the antivirus engines. A typical usage of the PolyPack service is as follows: (1) a user submits an unpacked binary via PolyPack's web interface, (2) the binary is packed using an array of packers and each packed version is analyzed by an array of antivirus engines, and (3) the detection results from the antivirus engines are analyzed to select the optimally packed version to provide the most AV evasion and the results are returned to the user.

PolyPack is targeted at providing a useful service for penetration testers who require the ability to create payloads that will evade the signature detection of a number of antivirus engines. For example, a penetration tester may be unaware of the antivirus product running on a target host/gateway and may desire maximal evasion or may be faced with an infection vector where AV-disabling shellcode may not be applicable. While it is certainly possible to manually pack a binary and test it against local antivirus engines, we believe that offering an automated service can offload the burden of a manual effort and offer advanced features and considerable extensibility.

Beyond its legitimate utility for penetration testers, PolyPack is an interesting concept with respect to the crimeware industry. While crimeware such as LuckySploit, Mpack, WebAttacker, phishing kits, and others have traditionally been sold in an ad-hoc manner, crimeware authors may find that a software-as-a-service (SaaS) or subscription model is an attractive alternative. Many of the benefits such as centralized development, management, and updating that drive the SaaS model for traditional software may translate effectively for crimeware tools as well. We believe that PolyPack represents the sophistication and automation of a crimeware-as-a-service (CaaS) deployment that is likely to emerge in the underground in the near future.

As with many penetration testing tools, PolyPack's offensive capabilities may be used for both good and evil. While unrestricted public access to such a service may border on irresponsible, we believe that PolyPack can be responsible deployed or licensed in a similar vein as many of the existing penetration testing frameworks such as CANVAS [7]. In addition to its offensive capabilities, PolyPack allows defensive researchers to better understand the current state of packer efficacy and the resulting impact that packers may have on host-based protection systems such as antivirus. Better understanding of the offensive capabilities of attackers can lead researchers to

[1]The PolyPack name is a spoof of the PolyUnpack research [20]

[2]The PolyPack service is available for restricted use at `http://polypack.eecs.umich.edu`

create more effective and resilient defenses.

Towards these goals, our paper makes three primary contributions:

- An analysis of a large dataset of malware of nearly 100,000 samples detailing the detection coverage of 23 antivirus engines against malware packed by 35 packer classes.
- The development of the PolyPack service, complete with a backend of 10 popular antivirus engines, 10 common packers, and supplementary features such as live updating.
- An evaluation of the increased evasion capabilities of the PolyPack service using 208 distinct malware samples compiled from source, each packed by 10 packers for a total of 2288 samples.

Our paper is structured around these contributions: In Section 2, we explore the current use of packers by malicious software and the diversity of antivirus detection capabilities using a dataset of nearly 100,000 samples. In Section 3, we describe the architecture and implementation of the PolyPack service. In Section 4, we evaluate the efficacy of our current set of antivirus engines and packers supported by PolyPack. Finally, we conclude in Section 5.

2 AvP: Antivirus vs. Packers

In this section, we investigate the diversity of packer usage observed in the wild as well as the varying detection capabilities of antivirus engines using a dataset of 98,801 malware samples from Arbor Network's Arbor Malware Library (AML) [12]. AML is composed of malware collected in the wild using a variety of techniques such as distributed darknet honeypots, spam traps, and honeyclient spidering. The 98,801 samples represent over a year's worth of AML collection.

2.1 Packer Classification

It is important to understand the current use of packers in the wild before investigating their efficacy against modern antivirus engines. Many factors may influence the diversity of packers used by malware authors including their feature sets, resistance to antivirus and reverse engineering, and availability of the packing tool.

Table 1 and Table 2 show the breakdown of packer usage in our dataset as determined by SigBuster [24] and PEiD [2] respectively, both signature-based packer identification tools. While the usual suspects top the list of the packers commonly used in the wild, the top 10 packer classes only represent 55.3% and 33.3% of the total samples respectively, indicating a substantial distribution of packers.

SigBuster Identifier	Count
Allaple	22050
UPX	11324
PECompact	5278
FSG	5080
Upack	3639
Themida	1679
NsPack	1645
ASPack	1505
tElock	1332
Nullsoft	1058

Table 1: The top ten packers classes in our AML dataset as determined by SigBuster.

PEiD Identifier	Count
UPX	11244
Upack	6079
PECompact	4672
Nullsoft	2295
Themida	1688
FSG	1633
tElock	1398
NsPack	1375
ASpack	1283
WinUpack	1234

Table 2: The top ten packers classes in our AML dataset as determined by PEiD.

In addition, a significant portion of the malware samples remain unidentified by SigBuster and PEiD, two of the most common tools for identifying packers. Of the 98,801 samples, 28,827 and 39,731 were unsuccessfully identified by SigBuster and PEiD (with BoB's userdb.txt) respectively. Of the 39,731 unidentified by PEiD, only 8,174 (20.6%) compress by more than 20%, indicating that the vast majority of these samples are indeed packed by unidentified packers. In addition, analysis of the overall binary entropy and the number of IAT entries when comparing the malicious binaries with a set of legitimate ones indicates that over 90% of all the samples in our dataset appear to be packed.

2.2 Antivirus Detection

More interesting than packer diversity is the wide range of detection capabilities that antivirus engine have when analyzing packed binaries. Such information can help

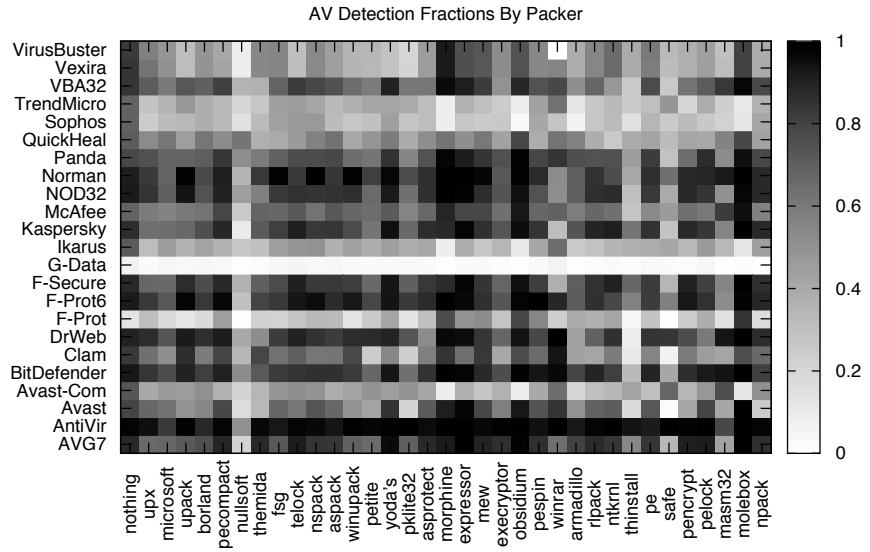

Figure 1: The fraction of detected binaries for 23 anti-virus systems and 35 most popular packers.

us determine if there are antivirus engines that are more effective against a wide range of packers or packers that are effective at evading a wide range of antivirus engines.

Figure 1 shows a set of 23 antivirus systems (a subset of those supported by VirusTotal [6]) crossed with the top 35 most popular packer families determined by PEiD in our malware dataset (including non-packed and compiler classes such as Borland). Each square represents the detection coverage of the particular antivirus engine against the malware samples packed by the particular packer. The greyscale shade of the square represents the percentage of samples successfully detected as malicious, ranging from 0% (white) to 100% (black).

It is important to take caution while deriving definitive results from the datapoints in Figure 1 as the overall detection rate by an antivirus engine may depend on not only the packer employed but also the underlying malware payload. Engines like AntiVir, which has a fairly dark stripe, may appear to be quite effective at detecting many packer classes, but this may be due to overaggressive heuristics flagging binaries with high entropy as malicious and can lead to significant false positives.

More important than individual antivirus or packer performance is observing the diversity of detection coverages across the board. The detection ranges can vary widely not only within a particular antivirus vendor against different packers but also across vendors with a particular packer. These results hint that the selection of a packer to ensure optimal evasion of antivirus engines may not be a trivial task and provide the primary motivation for our construction of the PolyPack service to automate this process.

3 The PolyPack Service

In this section, we briefly describe the overall architecture and implementation of the PolyPack service and then discuss several of the service's additional features that may be beneficial to pen-testers or other offensive parties.

3.1 PolyPack Architecture

The PolyPack architecture consists of three components: the web frontend, the packer service, and the antivirus service. An overview of the architecture is described in Figure 2.

3.1.1 Web Frontend

Users of PolyPack interact with the system through a simple web interface. The user can upload an unpacked binary and have it submitted for processing. PolyPack will process the binary with its collection of packers and antivirus engines and return the packed binary for optimal AV evasion to the user. Results from the PolyPack service can either be displayed in real-time in the user's browser or simply emailed to the user for later retrieval.

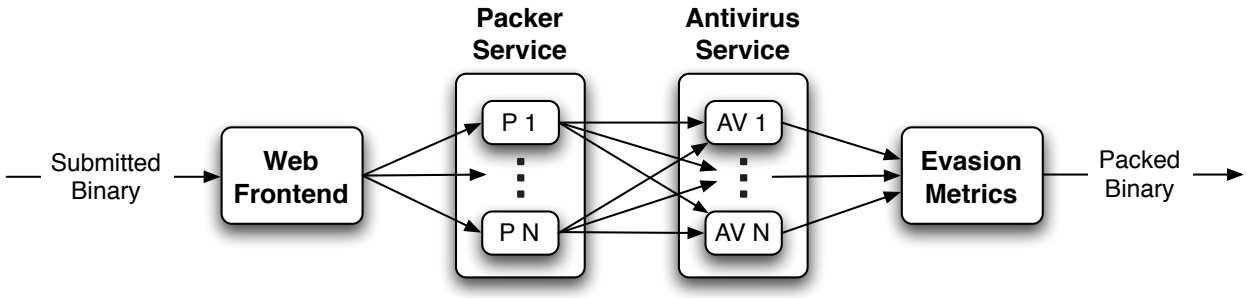

Figure 2: Conceptual overview of the PolyPack architecture.

3.1.2 Packer Service

Submitted binaries are first delivered to the packer service to be processed by an array of packers before being passed on to the AV service.

PolyPack's packing service currently supports 10 packers: ASPack [22], FSG [1], NsPack [13], Nullsoft [14], PECompact [3], tElock [23], Themida [17], UPX [9], WinUpack [5], and Yoda [4]. These packers were chosen as they represent some of the most common packers used by malware observed in the wild. While choosing common packers may result in greater detection rate by antivirus than more obscure packers, it provides a more representative view of the detection coverage experienced in the real world. We plan to significantly extend the set of supported packers in the future as adding additional packers is a trivial task. Indeed, only 32 lines of code on average are needed to support a packer.

3.1.3 Antivirus Service

After submitted binaries are processed by the packer service, they are fed into the antivirus service to be analyzed by a number of antivirus engines. The results from the various engines are collected and evaluated to determine the optimal packing scheme.

PolyPack's antivirus service is based on the CloudAV architecture [16, 15] which supports a backend of 10 popular antivirus engines including Avast, AVG, BitDefender, ClamAV, F-Prot, F-Secure, Kaspersky, McAfee, Symantec, and Trend Micro. These engines are hosted in disposable virtualized environments to maintain the integrity of the service. Again, the framework is designed around being extremely extensible for new AV engines, requiring only 42 lines of code on average to support an engine.

3.2 PolyPack Features

While the use of antivirus engines as a feedback mechanism to ensure evasion before releasing malware is a common practice among attackers, we believe that automating the process in a centralized service allows us to offer considerable extensibility and unique features beyond what may be feasible by a manual effort.

3.2.1 Evasion Metrics

What constitutes "optimal" evasion may differ based on the goals of the user of PolyPack. The default metric for determining how effectively a particular samples evades antivirus is by a simple count of the number of engines that fail to detect it. However, in the real world, not all AV vendors are deployed uniformly and some are significantly more popular and commonly deployed than others. Therefore, PolyPack offers the ability to weight evasion metrics by AV vendor market share size. In the future, we will allow users to specify their own weighting preferences based on the specific engines they would like to evade.

3.2.2 Live Updating

Once a malware sample is released into the wild, it may fall into the hands of an analyst or AV vendor that produces a signature to block it. Continually monitoring of a large number of AV engines may be a tiresome task for a malware author to determine if a new signature update detects their malware. Since PolyPack possesses both the submitted binary and the signature update feeds of the AV engines, it can automatically re-pack and push out a new optimally packed binary to the PolyPack user whenever a AV signature update is received. For up-to-the-minute evasion, live binaries may even be pulled from PolyPack directly and delivered to the victim via HTTP for certain infection vectors instead of being pushed to the PolyPack user.

Packer	Total	Median	Average
Unpacked	212	1	1.02
ASpack	+128	+0	+0.61
FSG	+39	+0	+0.19
NsPack	+239	+1	+1.15
Nullsoft	+646	+3	+3.11
PECompact	+509	+2	+2.45
tElock	+424	+2	+2.04
Themida	+935	+5	+4.50
UPX	+91	+0	+0.44
WinUpack	+230	+1	+1.11
Yoda	+654	+3	+3.14
Average	+389	+2	+1.87
PolyPack	+1005	+5	+4.83

Table 3: For each packer, we list the increase over the unpacked binaries of the total number of AV evasions across all binaries (out of 2080) and the median/average number of evasions per binary (out of 10).

Packer	Optimal Occurrences
Themida	122
Nullsoft	59
Yoda	24
PECompact	3

Table 4: The number of occurrences a packer produced the optimal packing for each of the 208 distinct samples.

3.2.3 Submission Confidentiality

As payloads may contain sensitive information or techniques, it is important that submissions to such a service be kept confidential. Unlike services such as VirusTotal [6] that share submitted samples with AV vendors, PolyPack maintains the confidentiality of submitted binaries. In addition, the backend AV engines are isolated from the network to ensure that samples or hash identifiers are not collected by the engine and leaked back to the AV vendor.

3.2.4 Future Capabilities

While our current implementation offers useful functionality, we plan to extend PolyPack to support more sophisticated capabilities:

- **Additional Packers:** Including additional exotic and sophisticated packers and techniques [11] in addition to the current set of well-known packers should significantly increase the flexibility of PolyPack's evasion.
- **Additional File Formats:** Besides the currently supported Portable Executable (PE) binaries, it would be useful to support additional binary formats such as ELF and Mach-O, as well as other common file types used to transport malcode such as PDFs, documents, and media files.
- **Automated Unpacking Resistance:** A significant number of projects have attempted to tackle the problem of automated unpacking [10, 21, 8, 18, 20, 19]. Providing resistance to these tools, in addition to AV engine evasion, is desirable.
- **Behavioral Antivirus:** Many antivirus engine augment their signature databases with runtime behavioral detection of malware. Instrumenting the backend to provide feedback on whether a binary tripped an AV engine's behavioral detection would be valuable to the PolyPack user.

4 Evaluation

To evaluate the efficacy of the PolyPack service, we analyze a dataset of 208 malware samples compiled from source. Each of these unpacked binaries is fed through the PolyPack service, resulting in a total of 2288 (208 unpacked and 2080 packed) samples to be scanned by the antivirus engines. Since we can analyze both the packed and unpacked binaries, we can detail exactly how effective each individual packer is at evading the antivirus engines and determine how much benefit the PolyPack service provides.

Table 3 details the results of our evaluation. The first row shows the baseline for the unpacked binaries. Across all of the 208 unpacked binaries and the 10 antivirus engines, a sample goes undetected only 212 times, out of a total 2080 scan events. On average across the 208 binaries, a sample only evades 1.02 of the 10 engines. After establishing this baseline for the unpacked binaries, we can measure how much additional evasion each individual packer provides. For example, the best individual packer, Themida, goes undetected in 935 more scan events than the baseline and corresponds to an average of evading 4.50 AV engines for a single sample. Table 3 also details the evasion efficacy if one were to choose a packer that represented the average of the 10 common packers in our evaluation.

At first glance, it may be tempting to select Themida for packing all binaries since it appears to be the "best" of our evaluated packers. However, while Themida has the highest evasion overall, it is not the optimal choice for *all* of the binaries. Since PolyPack is able to evaluate the best packer choice for each binary individually, it is able to achieve a greater evasion rate as seen in the last row of Table 3.

Table 4 breaks down the cases when packers other than Themida provide the optimal packing (for 86 binaries of the 208 binaries). Nullsoft provides the best evasion for 59 of the binaries, Yoda for 24 of the binaries, and PECompact for 3 of the binaries. These results clearly indicate that one packer does not fit all and the variety provided by PolyPack has a real and measurable effect on a sample's evasion of AV engines. As a more diverse set of packers are added to PolyPack, we expect these results to become even more pronounced.

5 Conclusion

In this paper, we have explored the construction and use of a service for the packing of binaries for optimal evasion of antivirus engines. We believe such a service is of value to penetration testers and is likely to represent the sophistication and automation that we will continue to see in crimeware industry in the near future. Understanding the deficiencies of our current defensive measures and the ease at which attackers can offensively innovate in the asymmetric battle against malicious software can aid our own efforts in developing adequate protection mechanisms.

References

[1] Fast Small Good (FSG). http://www.woodmann.com/collaborative/tools/index.php/FSG, 2009.

[2] PEiD. http://www.peid.info, 2009.

[3] Bitsum Technologies. PECompact. http://www.bitsum.com/pecompact.php, 2009.

[4] Danilo Bzdok. Yoda's Crypter. http://yodap.sourceforge.net, 2009.

[5] Dwing. UPack. http://dwing.cjb.net, 2009.

[6] Hispasec Sistemas. Virus total. http://virustotal.com, 2004.

[7] Immunity, Inc. CANVAS. http://www.immunitysec.com/products-canvas.shtml, 2009.

[8] Min Gyung Kang, Pongsin Poosankam, and Heng Yin. Renovo: a hidden code extractor for packed executables. In *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*, 2007.

[9] Markus Oberhumer. UPX. http://upx.sourceforge.net/, 2009.

[10] L. Martignoni, M. Christodorescu, and S. Jha. Omniunpack: Fast, generic, and safe unpacking of malware. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2007.

[11] Matt "skape" Miller. Using dual-mappings to evade automated unpackers. In *Uninformed Journal Vol 10*, 2008.

[12] Arbor Networks. Arbor malware library (AML). http://www.arbornetworks.com, 2007.

[13] North Star Software. NsPack. http://www.nsdsn.com/eng/index.htm, 2009.

[14] Nullsoft, Inc. NSIS. http://nsis.sourceforge.net, 2009.

[15] Jon Oberheide, Evan Cooke, and Farnam Jahanian. Rethinking antivirus: Executable analysis in the network cloud. In *2nd USENIX Workshop on Hot Topics in Security (HotSec 2007)*, August 2007.

[16] Jon Oberheide, Evan Cooke, and Farnam Jahanian. CloudAV: N-Version Antivirus in the Network Cloud. In *Proceedings of the 17th USENIX Security Symposium*, July 2008.

[17] Oreans Technology. Themida. http://www.oreans.com/, 2009.

[18] Piotr Bania. Generic Unpacking of Self-modifying, Aggressive, Packed Binary Programs. http://piotrbania.com/all/articles/pbania-dbi-unpacking2009.pdf, 2009.

[19] D. Quist and Valsmith. Covert Debugging: Circumventing Software Armoring. In *Proc. of Black Hat USA*, 2007.

[20] Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, and Wenke Lee. PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware. In *The 22th Annual Computer Security Applications Conference (ACSAC 2006)*, Miami Beach, FL, December 2006.

[21] Monirul Sharif, Andrea Lanzi, Jonathon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators. In *Proceedings of the IEEE Symposium on Security and Privacy (Oakland '09)*, 2009.

[22] StarForce. ASpack. http://www.aspack.com/, 2009.

[23] tHE EGOiSTE/TMG. tElock. http://programmerstools.org/node/164, 2009.

[24] Toni Koivunen / Teamfurry.com. SigBuster. http://www.teamfurry.com, 2009.