

Serial Variant Evasion Tactics

Techniques Used to Automatically Bypass Antivirus Technologies

Gunter Ollmann, VP of Research, Damballa

Introduction

Organized crime and malicious code authorship collided some time ago to create a hybrid breed of cyber-criminals...

Creators of malicious software and botnet agents use a broad spectrum of tools and techniques to create one-of-a-kind packages that easily bypass traditional antivirus technologies. Armed with either the source code or a compiled version of their favorite malware, the cyber-criminal can selectively apply manipulation technologies that radically alter the fabric of malware. The result is a stealthy threat that evades signature-based detection systems, static analysis tools, behavioral monitoring environments and sandbox technologies. And yet, the malware retains the same core malicious functionality and remote control mechanisms.

This paper examines the tools and techniques cyber-criminals use to automate the generation of new variants of malware and bypass enterprise detection technologies. One of the most critical of these concepts is based upon serial variant malware production.

The Botnet Players

Organized crime and malicious code authorship collided some time ago to create a hybrid breed of cyber-criminals and an entire ecosystem of malware production and service delivery capabilities. Each member of the cyber-criminal fraternity plays a specific role in propagating attacks and the siphoning of ill-gotten gains.

The result is a complex web of criminal relationships, tool development specialists and fraud delivery services that can be difficult to understand. For the sake of clarity, these can often be summarized as follows:

Malware author – Original malware creator(s) that provide “off-the-rack”, do-it-yourself (DIY) malware construction kits and custom malware solutions.

Botnet master – Individuals or criminal teams that own botnets and maintain ultimate control over the distributed Command-and-Control (C&C) infrastructure. The botnet master may assume the role of botnet operator or rent the botnet to criminal operators.

Botnet operator – Sets up botnets by leveraging malware and other tools, and then monitors/manages botnets for ongoing fraud or attack operations.

Criminal operator – The individual or team actually using the botnet to conduct malicious activities and financial gain.

Serial Variants

The process of automatically churning out new variants of malware on a massive scale has been ongoing for quite some time. The Storm worm was perhaps the most famous botnet to employ the tactic of creating multiple variants of a particular malware agent in advance of the attack, and then releasing each new variant at scheduled intervals to constantly remain ahead of antivirus protection updates.

Serial variant production has evolved into a streamlined business process since Storm. Botnet creators and malware authors now employ an arsenal of tools and technologies through which they automate the production of these variants. A wide assortment of off-the-shelf commercial software protection tools and specialized anti-antivirus manipulation tools have standardized the process and commoditized its pricing.

The serial variant production systems now in operation by malware developers and botnet operators can be broken down in to several distinct stages:

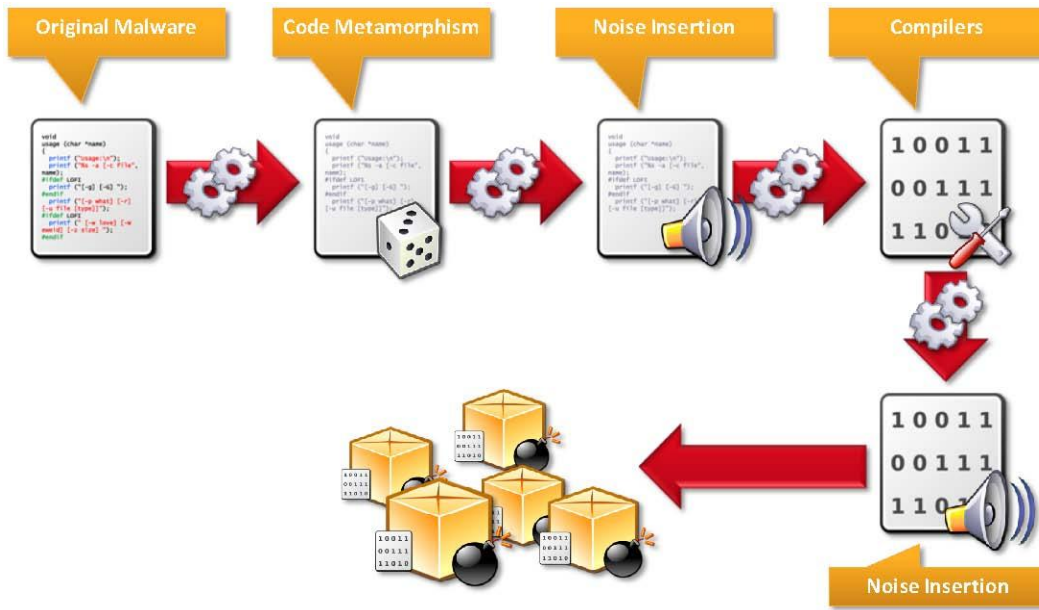


Figure: Serial variant production

There are four key production steps for creating serial variants capable of bypassing most common antivirus detection technologies:

1. Code metamorphism
2. Noise insertion (Code)
3. Compiler settings
4. Noise insertion (Binary)

Code Metamorphism

Code metamorphism (including polymorphism) is a common tactic that attempts to bypass pattern recognition systems employed by antivirus, intrusion prevention systems (IPS) and data leakage prevention (DLP) technologies.

Automated tools seek to manipulate the structures of the source code of the bot malware – altering their “shape” – by reordering and replacing common programmatic routines. For example:

- Swapping of equivalent code constructs
 - Converting one loop type in to another – e.g. *For...Next* becomes *If...While* or even *Case...When*
 - Assigning variable values in different ways – e.g. *i=1++* versus *i=i+1*
 - Clearing memory and registers
- Changing the order of the code
 - Swapping registers
 - Reordering instructions
 - Defining functions in a different order

```

Equivalent constructs [edit]
while (condition) {
    statements;
}

is equivalent to

if (condition) {
    do {
        statements;
    } while (condition);
}

or

while (true) {
    if (!condition) break;
    statements;
}

or

goto TEST;
LOOPSTART:
    statements;
TEST:
    if (condition) goto LOOPSTART;
    
```

Figure: Equivalent loop constructs (courtesy Wikipedia)

Noise Insertion (code)

“Noise” refers to redundant code that either does nothing or is interpreted as doing nothing, and has no negative impact upon compiled code the malware author wishes to execute. Noise code is designed to help the malware bypass signature-based detection systems – but may also be used to thwart basic behavioral and heuristics based detection systems (e.g. by identifying the framework or operating system in which the malware is executed within, and performing different functions depending upon the specific environment).

Examples of noise insertion tactics include:

- Whitespace and noise instructions
 - Code constructs that do nothing and incur no load to the compiled application – e.g. *if 1=1* or even *sleep(0)*
- NOP/NOOP
 - For example *i+1;*
- Unused functions and procedures
 - Inclusion of code with use exceptions that will never occur or will never be called – for example, calculate Pi of var.A is not equal to var.A.
- Unused variables and arrays
 - The definition of variables and arrays that are not used in the malicious code segments.

Compiler Settings

The compiler(s) used by the malware author or botnet master can have a significant effect on the final output code. Different compilers, different versions of compiler and even different compiler settings can result in substantially different binary code output.

Minor tweaks to compiler settings are easy to automate and provide a reasonable degree of flexibility in the creation of serial variants. Scripts and automation frameworks are often used by malware authors to speed up this compiler-oriented variant production process.

In general, the number of compiler parameters that can be tweaked and used to create malware variants that continue to work under the most common operating systems is finite and often restrictive in scope. For this reason, serial variant authors often combine compiler setting modulation with other techniques.

Noise Insertion (binary)

Malware authors can also insert noise into a compiled malicious binary. This noise is typically applied to the beginning or end of the binary, rather than the functional malicious code in the middle.

This binary noise (often non-interpretable garbage) is very easy to add and can be used to grow a malicious binary to any size a botnet master wants. For example, the author may want to seed Torrent networks with fake movies that happen to be exactly the same size as the real ones, but actually contain the malicious bot agent.

Serial Variant “Protection”

Increased competition amongst malware authors continues to drive change in the way these groups seek to protect their “investment” in the malicious code they produce. It’s no longer enough to just increase the sophistication of a bot agent. Now, authors seek to protect their investment from copy-cat competition and reverse engineering by security researchers.

The techniques used by cyber-criminals have been publicly available for well over a decade, originating within the commercial software sphere as copyright protection for games and digital rights management (DRM) for a range of media. Now, these same protection strategies have been applied to the automated production of malware. In almost all cases, this automated process extends the diversity of serial variant malware production – although the core purpose remains to evade advanced automated detection systems (e.g. behavioral and heuristic analysis) and the manual reverse engineering techniques used by threat analysts.

While the previous discussion of serial variant production focused upon the manipulation of malware or bot agent source code, other manipulation techniques focus upon compiled binary files, utilizing distinct technologies that protect and obfuscate file content as well as substantially increase the diversity of serial variants that can be produced.

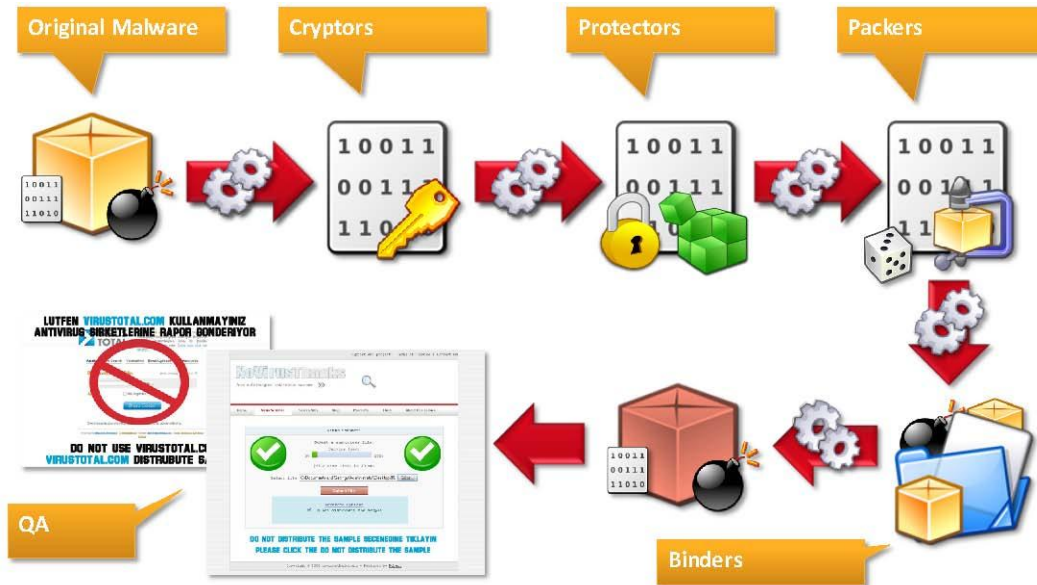


Figure: Automated bot agent production

There are a number of distinct processes and tools that can be (and are) used by criminal operators that enhance the robustness of malware (serial variant or otherwise) from dynamic detection technologies and analysis processes. These steps typically fall into the following categories:

1. Crypters
2. Protectors
3. Packers
4. Binders
5. Quality Assurance

“Quality Assurance” normally entails verifying that each malware sample cannot be detected by dozens of different commercial antivirus engines. It is often provided as a managed service to malware authors by other cyber-criminals.

Crypters

Crypters (or “Cryptors”) encrypt malware so that signature detection systems and static analysis processes are ineffectual. Crypters typically encrypt the contents of the malware executable, and then only decrypt sections of code that are in the process of being executed on the victim’s computer.

This technique means that host-based detection technologies cannot inspect the executable content prior to it being loaded from disk and into memory. It also means that popular reverse engineering tools such as IDA Pro will struggle with their analysis without full knowledge of the decryption algorithm in order to work properly. Similarly, signature-based antivirus products work by sifting the executable code of the malware for atypical coding markers, and then search for pre-defined regular expressions. If antivirus detects known malicious strings, it will delete or quarantine the suspicious file. Crypters prevent antivirus scanners from seeing these key coding markers.



Figure: Common crypter tools

Crypter tools such as the examples shown above make the process of employing cryptographic routines and memory protection algorithms a trivial task. Many tools allow a cyber-criminal to select their own cryptographic algorithm and encryption keys to encode the malicious. In many cases Crypters have also added packer functionality. As such, several popular “packer” and “binder” tools in use by malware builders and bot masters also include encryption capabilities.

Protectors

Protectors are a relatively new class of evasion employed by malware authors. The “protector” technology was originally designed for commercial use to protect online games from abuse (e.g. reverse engineering that led to game stats manipulation, etc.) and as a DRM protection technology. Ironically it is now more commonly employed by criminals.

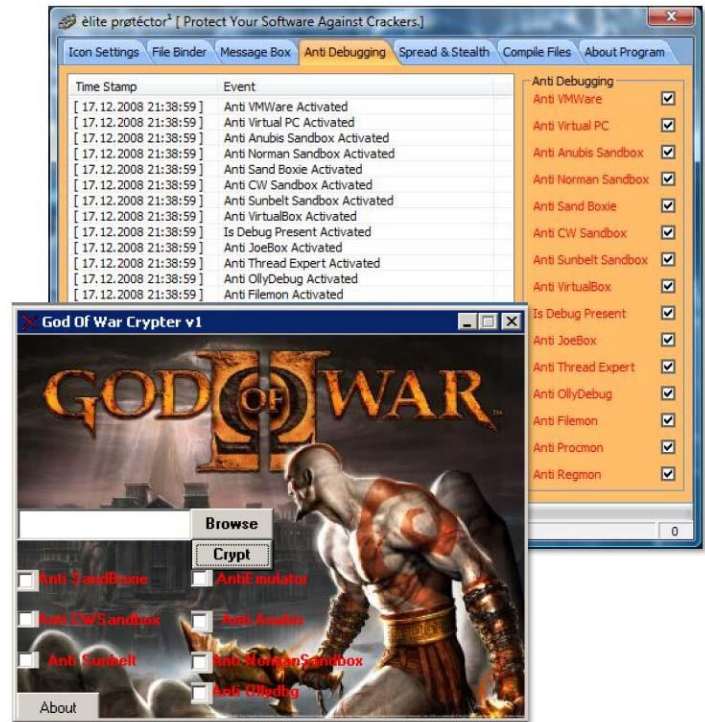


Figure: Anti-debugging features of Protectors

Protectors automatically add specific anti-debugging features to malware that prevent security researchers and automated sandbox analysis technologies from dissecting samples. As seen in the screen captures above and below, the list of anti-debugging features can be quite extensive.



Figure: Protector & Crypter combination tool

In standard operation, protectors detect the use of debuggers or virtualization techniques. If seen, the malware then causes different operations to occur – often hiding the malicious intentions of the malware or intentionally causing the particular debugging technology to falter and fail in its analysis.

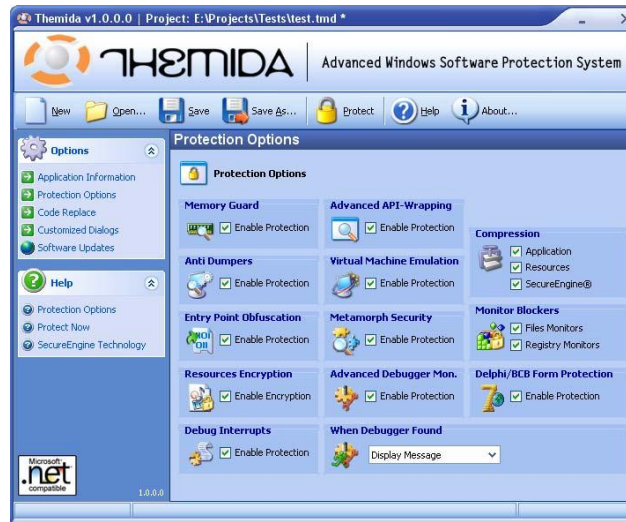


Figure: Commercial software protectors - Themida

Cyber-criminals also make use of commercial software protection technologies to protect malware investments. Many hacking sites provide detailed tutorials on how to leverage commercial software protection to produce malware serial variants. By way of example, Themida, produced by Orleans Technologies, appears to be a popular commercial tool now used by malware authors – and an extensive array of guides and tutorials can be found on hacking sites that specialize in malware production and botnet management, along with pirated copies of the tool.

Newer generations of malware creation tools (i.e. DIY Malware Kits) increasingly incorporate protector functionality. Authors simply click a checkbox for the desired functionality. Advanced protectors may also include “hack-back” routines designed to identify whether the malware is operating within a sandbox or virtual environment. If so, the malware then knows to use known exploit code in an attempt to compromise the host machine by breaking out of the sandbox.

Packers

Packers have had the longest and most successful criminal dual-use career. Packers were originally designed to make binary files and installation kits smaller and more portable, which made them quicker to download over dialup and slow Internet connections. However, malware authors discovered that smaller packages sped up the infection process while also making it much more difficult for antivirus to detect the malicious payload.

“Packers” now are a core technology for botnet masters. It enables malware to bypass both antivirus signature and hash-matching algorithms. There are literally hundreds of different packers in widespread use. Only a handful of them are also commonly used for legitimate commercial business. Some of the most advanced packers employ polymorphic output capabilities, which mean that the malware binary is structurally different every time the packed version is executed.

The most popular packer technology in operation today is UPX (screenshots below). It is a popular commercial packing technology, but is also a preferred packer of choice amongst malware authors.

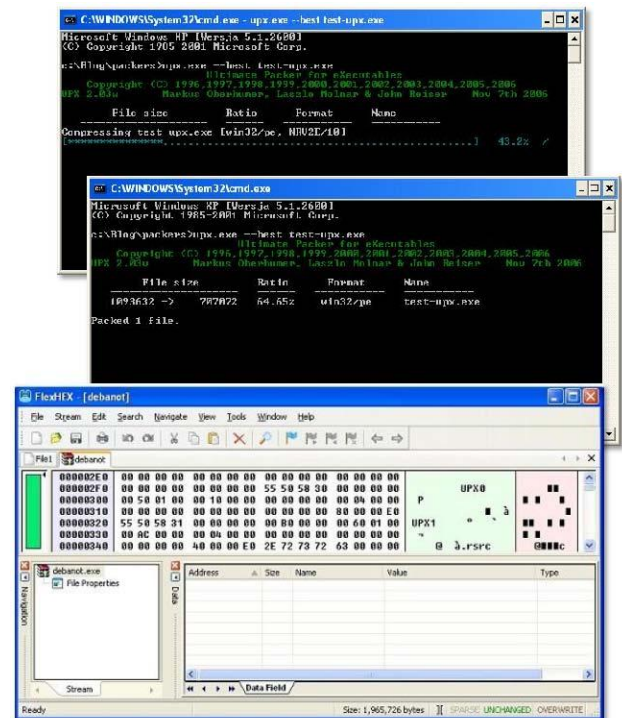


Figure: UPX packer in operation and headers visible in a file HEX editor

Binders

Binders are an old technology typically used by malware authors to “embed” and Trojan other software packages. These tools are a method for aiding propagation of the malware component, tricking victims into executing a popular file or something that looks legitimate.

Binders have proved popular with botnet masters that use Torrent networks and newsgroups to spread their malware. The malicious code is embedded inside files that are frequently searched for and downloaded. Binders are also used to create the packages that malware downloaders automatically install on victims of drive-by attacks – thereby increasing the breadth of potential victims and the probability of successful compromise.

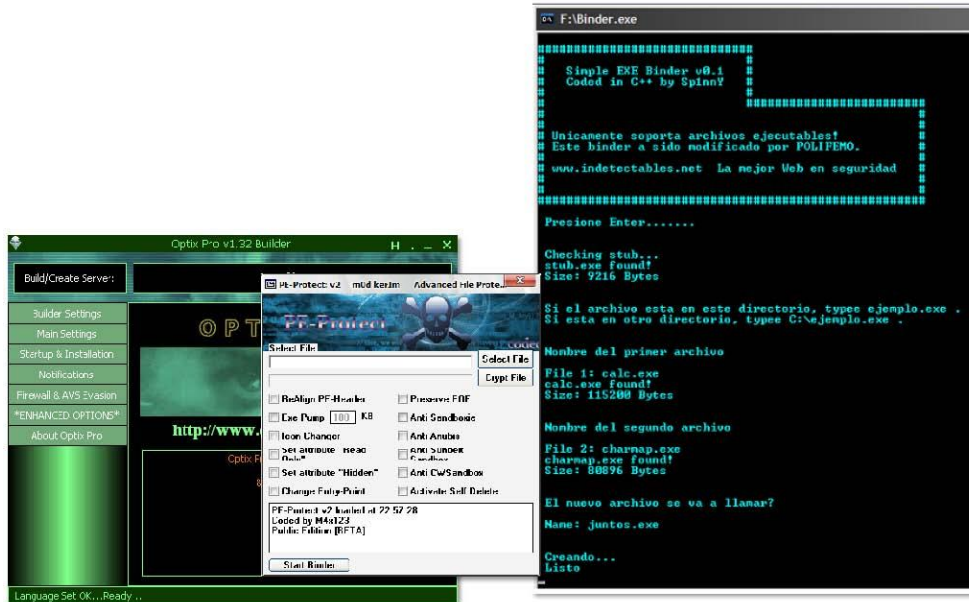


Figure: Binders in action

Quality Assurance

It is important to understand that malware authors and botnet operators also invest considerable time and effort in their quality assurance practices. Malware created using the tools and processes described above are typically passed through multiple commercial antivirus products to verify that they will not be detected prior to their criminal deployment.

Forums dedicated to malware and botnet development frequently extol the virtues of checking antivirus coverage, and provide advice on how best to do it – including the review of online antivirus checking portals and discussions as to whether they send any submitted samples to antivirus vendors afterwards.

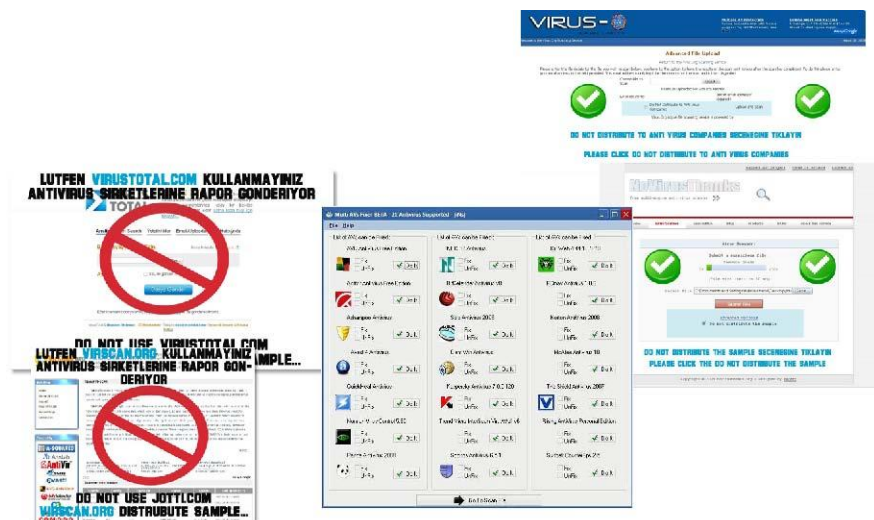


Figure: AV testing tools for botnet authors

Newer DIY tools speed up the QA process by automatically downloading trial versions of antivirus products, or use “keygens” to create working license credentials and register antivirus products, manage signature updates, stream new serial variants through dozens of antivirus tools, and modify detectable malware binaries for greater stealth.

Future Serial Variant Updates

Serial variants are a proven technique for evading malware detection technologies at both the network and host level. Most deployments of auto-generated malware have been focused upon the initial compromise vector of the victim’s host. For example, drive-by-download infection engines often have the capability to generate and serve one-of-a-kind malware variants for each potential victim in a process sometimes referred to as “server-side polymorphism” and “x-morphic exploitation”.

Given the continued advances in guest operating system (OS) security and the resultant increases in difficulty in initially compromising a host through vulnerability exploitation (excluding social engineering tactics), it is reasonable to expect that botnet masters will place an growing emphasis on retaining control of the systems that have already been compromised.

A consequence of this trend will be the adoption of serial variant bot agent updates being sent to the compromised host at regular intervals. Botnet masters already update bot agents regularly to add new functionality and fraud capabilities on an ad hoc schedule. The next step is for botnet masters to schedule serial variant bot agent updates to be applied to already compromised hosts in order to keep ahead of updates to “competing” host-based detection technologies.

Meanwhile, fierce competition between professional malware authors and tool producers will continue to drive a fast-paced evolution of the technologies used in serial variant production. The technologies that are likely to have the biggest impact in cyber-criminal serial variant production lines over the next few years include:

- **Identification of malware being operated within a monitored system.** Today several tools can implant the capability of identifying whether the malware is operating within a virtual or sandboxed environment, and then cause different code paths to be executed. Future advances in this area will likely encompass more sensitive checks to see if a real person/victim is interactively using the host rather than an automated analysis engine.
- **Cryptors that unencrypt and execute smaller sections of malware code at any point in time** to better evade memory inspection, as well as to disguise non-executed encrypted code as benign data that can bypass heuristic-based detection systems.
- **Identification of whether the malware is being studied by an automated analysis engine** (either behavioral or static analysis) and the exploitation of known vulnerabilities within these engines to cause the overall host integrity to fail.
- **Custom packers** that appear to be common commercial packers, but will unpack different coded payloads depending on whether the malware unpacked itself or an “unauthorized” external static analysis/debugger tool unpacked the malware.
- **Managed QA services** that include many more (if not all) commercial antivirus, IPS and DLP products, which will enable a cyber-criminal to upload and test batches of newly produced serial variant strains of malware for successful evasion, and then automatically alter detected the samples to ensure future evasion.
- **Enhanced anti-debugger and DRM technologies** to prevent automated and manual static analysis of malware samples captured by security vendor investigators.

Conclusions

Malware authors adopted serial variant construction as a means of evading detection. As long as they can release newer versions of their malware faster than antivirus companies can release detection updates to their customers (i.e. the malware author’s victims), these criminal operators will continue to have the upper-hand.

In response, antivirus vendors have enhanced their capabilities to obtain new malware samples (e.g. spam traps,

customer uploads, honeypots, etc.) and sought to quicken the pace at which they develop and distribute new detection signatures or algorithms. Today, many traditional antivirus vendors have launched (or are about to launch) new cloud-based systems in an effort to minimize content distribution delays and make it easier to collect malware samples.

In the past it may have taken between three to five business days to process a particular malware sample, develop a new signature, test and QA the update, and finally make it available to customers through a product update mechanism. By operating a “cloud” service, many of these antivirus vendors seek to compress this process to 24 hours or less.

Unfortunately all the malware creator or botnet operator needs to do to evade these new and improved detection update mechanisms is to continue to release serial variants fractionally faster than the antivirus vendors release their protection updates. If it takes an antivirus vendor 24 hours to turn-around a new detection signature, the botnet operator just needs to ensure that he releases a new serial variant every 23 hours.

Any detection technology that requires a malware sample to be acquired before a signature can be created will always be playing catch-up. Cyber-criminals understand this advantage, and serial variant production techniques will continue to evolve in order to maintain this lead.

Relative Effectiveness

Botnet masters have an extensive array of tool and techniques that be used to bypass traditional AV technologies. The following table provides a comparison of the various techniques discussed in this paper and their effectiveness against standard malware detection technologies.

	File checksums	RegEx Signatures	File Heuristics	Behavioral Analysis	Debugger Analysis	Static File Analysis	Reverse Engineering
Code Metamorphism	√√√	√√√	√√				
Noise Insertion (code)	√√√	√√√	√				
Compiler Settings	√√√	√√	√		√		
Noise Insertion (binary)	√√√	√					
Crypters	√√√	√√√	√√√	√√√	√√√	√√√	√√
Protectors	√√√	√√√	√√√	√√√	√√√	√√√	√√√
Packers	√√√	√√√	√√√	√√		√	√√
Binders	√√√	√√	√		√		

√ = Relative effectiveness in bypassing the detection/analysis technique

Additional Reading

“X-morphic Exploitation”, IBM, Gunter Ollmann, 2007, <http://www.technicalinfo.net/papers/Xmorphic.html>
 “The Botnet vs. Malware Relationship”, Damballa, Gunter Ollmann, 2009, [http://www.damballa.com/downloads/d_pubs/WP%20Many-to-Many%20Botnet%20Relationships%20\(2009-05-21\).pdf](http://www.damballa.com/downloads/d_pubs/WP%20Many-to-Many%20Botnet%20Relationships%20(2009-05-21).pdf)

About Damballa

Damballa is a pioneer in the fight against cybercrime. Damballa provides the only network security solution that detects the remote control communication that criminals use to breach networks to steal corporate data and intellectual property, and conduct espionage or other fraudulent transactions. Patent-pending solutions from Damballa protect networks with any type of server or endpoint device including PCs, Macs, Unix, smartphones, mobile and embedded systems.

Damballa customers include mid-size and large enterprises that represent every major market, telecommunications and Internet service providers, universities, and government agencies. Privately held, Damballa is headquartered in Atlanta. <http://www.damballa.com>

Prepared by:

Damballa Inc.

<http://www.damballa.com>

Copyright 2009. All rights reserved worldwide.