Engineering Challenges in Immunizing Systems Against Malicious Code

Craig Chamberlain
Principal Technical Consultant
Verdasys, Inc.

Background

Organizations are confronted with troubling developments the malware problem phenomenon

- Proliferation of malware of increasing volume and quality
- Targeted attacks
- Professional developers taking the place of amateurs
- Various organizations developing capabilities to produce malware on an industrial basis

Background

A novel malware approach is presented that was developed and implemented as a feature extension to a commercial tool for the Windows platform:

Digital Guardian - Verdasys, Inc. http://www.verdasys.com

Requirements

1. Prevent de novo infection by malicious programs including viruses, worms and Trojan horse programs.
2. ...without first obtaining new code, virus definitions, signature files, or data.
3. ...consistent with intermittent connectivity including no facility to download data files.
4. Maximally generic and thus independent of specific pattern matching or algorithmic methods of code examination and classification.
5. Be configurable for mandatory enforcement, i.e., disallowance of override once installed.
6. Be nevertheless manageable at enterprise scale and with a constant flux of changes and adds to people and facilities.
7. Fail closed.

The Malicious Code Problem

Cohen (1987) proved that no single algorithm can detect all virus programs which might exist, and separately (1994) showed that access control is ineffective against malicious code.

The Malicious Code Problem

Szor (2005) and Chess & White (2000) both point out that there is a threshold of code mutation beyond which viruses can exist which no algorithm can detect.

The Malicious Code Problem

Grayaznov (1999) notes that integrity checking tools are themselves targeted by slow infector viruses, which wait for files to be modified and then piggyback their infection onto the legitimate modification thus to escape notice.

The Malicious Code Problem

As described by Szor (2005) , first- and second-generation virus detectors used a variety of code and pattern matching techniques and, later, algorithmic detection in a custom language.

The Malicious Code Problem

Next came CPU emulation techniques to detect polymorphic viruses (which partially mutate their own code in order to frustrate pattern matching scanners). Szor (2005) notes the significant challenge posed by polymorphic viruses that use entry point obscuring techniques (effectively randomizing their location within the infected host file in order to make detection more difficult).

The Malicious Code Problem

Szor also notes that some metamorphic viruses, which produce true mutations that do not resemble the parent, would require a pattern matching against an infeasible number of patterns.

Christodrescu, et. al, (2003) found that even simple code obfuscation techniques, such as inserting NOP instructions defeated commercial virus scanners.

The Malicious Code Problem

Szor describes geometric detection (examining changes to file structures), heuristic analysis and behavior blocking tend (which generate false positive and false negatives). Neural networks can produce acceptable rates of false positives, but require considerable training, are subject to overtraining, and tend toward unacceptable false positive ratios when malicious code closely resembles non-malicious code.

Intrusion Prevention

Like intrusion detection, intrusion prevention technologies require significant training and administration and face technical challenges in achieving acceptable rates of false positives and false negatives using behavioral and statistical anomaly detection techniques.

Misuse Detection e.g. Code Signature Matching

- Signature matching is inherently reactive and cannot offer detection or prevention of newly emerged or “zero-day” malicious programs.
- Signature matching is ineffective against unique programs that never see widespread distribution and are consequently never classified by a signature producer.
- Signature matching is ineffective against programs for which source code is available, facilitating the compilation of binaries which are sufficiently different that they match no available code signatures.
- Signature matching will generally ignore dual-use applications.
- Signature matching cannot scale indefinitely in the face of industrial malware production.

The Immunity Problem

In the absence of a strong generalized defense, biologic hosts must experience an infection to develop antibodies to it. Inherently reactive, this offers little or no resistance to a previously unknown pathogen..

The Immunity Problem

For that, the organism cannot rely on time-lapsed development of pathogen-specific antibodies, but must instead be able to immediately disambiguate "self" and "not-self," which Forrest, et. al, (1996) were perhaps the first to attempt adapting to the digital world.

The Immunity Problem

Our challenge, as engineers, is to make such determinations on the fly in a way that is both effective, prompt, and non-interfering with needed work output of people and/or machines

Defining “not self”

Option one: programs which are attempting to enter the file system from a foreign (e.g. network) location are “not self”

Option two: programs which are sourcing from anywhere except a trusted process and / or user account such a software distribution tool are “not self”

Specification Based Object Level Security

- Specification based detection / policy enforcement is highly effective
- Without limitations of misuse or anomaly detection
- Requires defining a policy and the ability to enumerate all behaviors which do not violate policy

Impelmentation

Using kernel level I/O system call interception, I/O system calls are subjected to security policy.

Fine-grained rules can be enforced against a rich set of event attributes including process names or cryptographic hashes, user accounts, file types or paths, network sockets, and regular expressions.

Architecture

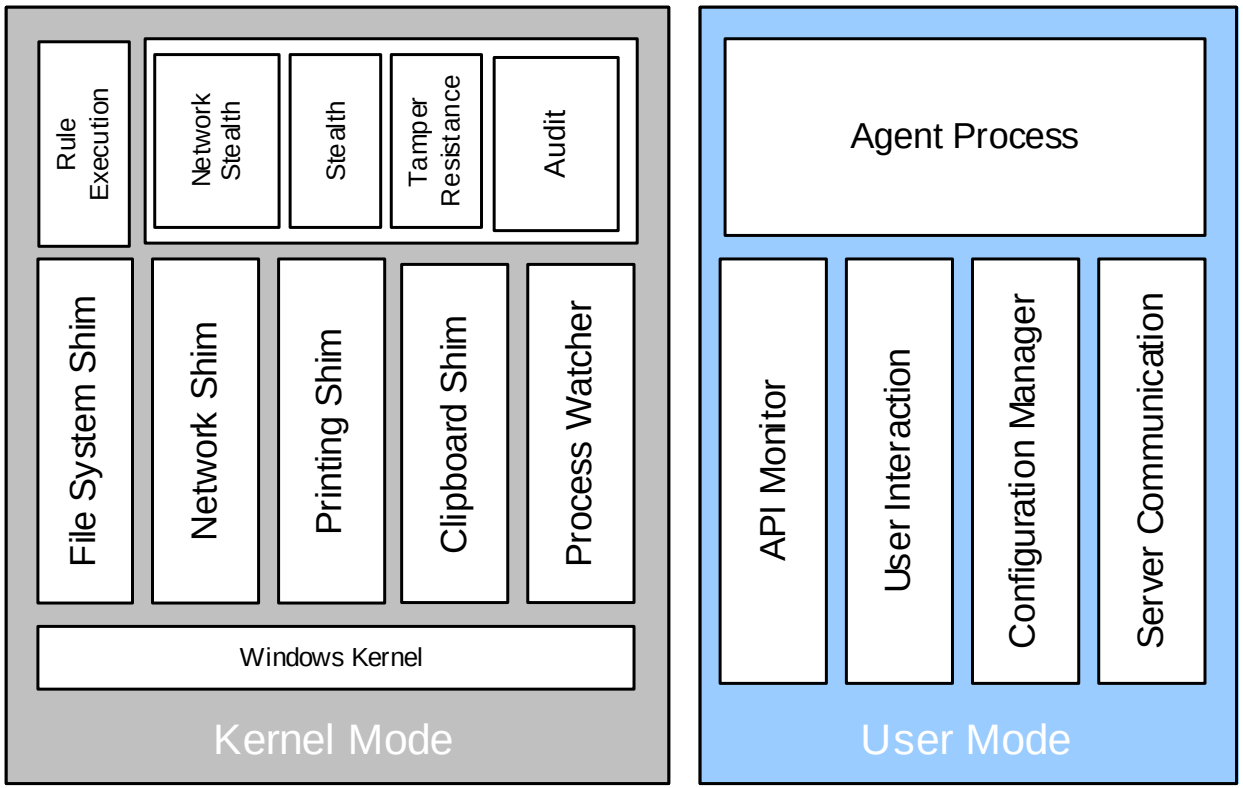

Figure 1. The Digital Guardian Agent Functional Components

Example: A Browser Helper Object (BHO) Detect

8/25/2004 5:03:36 PM iexplore.exe File Write Fixed incfindbho.dll
8/25/2004 5:02:43 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 5:02:43 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 5:02:41 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 5:02:40 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 5:02:36 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 5:02:36 PM iexplore.exe File Read Fixed s.dat
8/25/2004 5:02:36 PM iexplore.exe File Read Fixed s.dat
8/25/2004 5:02:05 PM iexplore.exe File Write Fixed incfindbho.dll
8/25/2004 4:56:07 PM iexplore.exe File Write Fixed incfindbho.dll
8/25/2004 4:52:27 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80
8/25/2004 4:50:39 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80

Solution: Prevent Event One

8/25/2004 5:03:36 PM iexplore.exe File Write Fixed incfindbho.dll

8/25/2004 5:02:43 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80

8/25/2004 5:02:43 PM iexplore.exe Connect Outbound TCP IP ADDRESS REDACTED 80

Malware Mitigation Policy Logic

- Block untrusted processes from writing to system paths.
- Block Internet facing applications from writing or renaming executable files.
- Block winzip and other file compression utilities from writing or renaming files with executable extensions.
- Block web browsers from writing or renaming files with executable extensions.
- Block newsreaders from writing or renaming files with executable extensions.
- Block messaging applications from writing or renaming files with executable extensions.
- Block P2P applications from executing, from making network connections or from writing or renaming files with executable extensions.
- Block email applications from writing or renaming files with executable extensions.
- Block reading and / or writing of executable file types on network drives.
- Block reading and / or writing of executable file types on removable media.

Malware Immunization Policy Logic

- If the program binary's hash does not match the hash of a trusted program such as a software distribution tool, don’t permit it to write executable files to the file system or create / modify files within system paths.

- If the program is not running within a trusted user context, do not permit it to write executable files to the file system or create / modify files within system paths.

Status

- Functional lab testing and initial interoperability testing successful.
- Field trials underway.
- Future directions include determining best method of exception granting (code signing? Source address?) ; file header inspection ; whitelist integration

Credits

David Chess and Steve White. *An Undetectable Computer Virus.* Presented at the Virus Bulletin Conference, September 2000

Mihai Christodorescu and Somesh Jha. Static Analysis of Executables to Detect Malicious Patterns. Proceedings of the 12th USENIX Security Symposium, August 2002, Washington, DC.

Fred Cohen, "Computer Viruses: Theory and Experiments", Computers and Security 6 (1987)

S. Forrest, S. Hofmeyr, and A. Somayaji. “Computer Immunology”.. *Communications of the ACM* Vol. 40, No. 10, pp. 88-96 (1997).

Dmitry Gryaznov. *Scanners of The Year 2000: Heuristics* in Proceedings of the Fifth International Virus Bulletin Conference, 1999.

Peter Szor. *The Art of Computer Virus Research and Defense.* 2005 Symantec / Addison Wesley, 2005

Q & A

Craig Chamberlain
Consultant
mail@craigchamberlain.com
http://craigchamberlain.com

Daniel Geer, Jr. Sc.D.
VP, Chief Scientist
Verdasys, Inc.
geer@verdasys.com
http://www.verdasys.com