
Intrusion Tolerance and Worm Spread *

Tomás E. Uribe

Steven Cheung

Joshua Levy

Alfonso Valdes

System Design Laboratory
SRI International
333 Ravenswood Ave.,
Menlo Park, CA 94025
uribe@sdl.sri.com

Abstract

We show how the Dependable Intrusion Tolerance (DIT) server architecture prevents the effects and propagation of some common Internet viruses and worms. This results from complementary detection and prevention mechanisms that provide defensive depth, and the application of the principle of least privilege at the network level, including the use of signature-based IDS to enforce higher-level specifications.

1 Introduction

The Dependable Intrusion Tolerance (DIT) project [8] aims to develop a prototype intrusion-tolerant Web server architecture. The architecture focuses on content integrity, and trades off performance vs. integrity guarantees. The purpose of this abstract is to describe how the DIT architecture also provides protection against the spread of Internet worms and viruses.

The Code Red virus [5] and the SQL slammer worm [1] are recent and notorious examples of fast-spreading Internet worms. In addition to the damage that such a worm might do to the individual system it infects, the large amounts of bandwidth it consumes when propagating through the Internet results in denial of service, even for machines that are not infected [4].

2 The DIT Architecture

Figure 1 presents a high-level view of the DIT architecture. A hardened *proxy server*, which can be replicated for additional resiliency, mediates client requests, which are forwarded to one or more commercial off-the-shelf (COTS) application servers that provide the actual

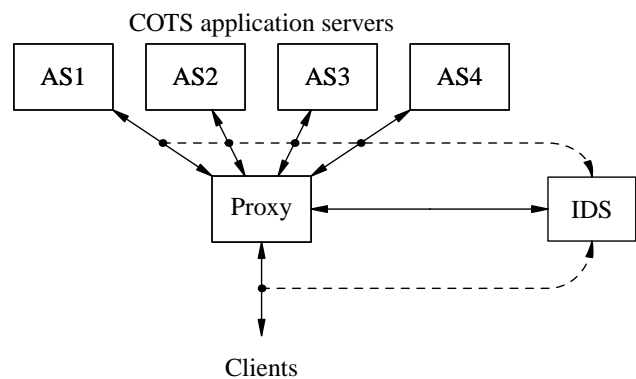


Figure 1. DIT architecture

content being served. The current *regime* determines how many redundant application servers are used for each query. Once the proxy receives the replies from the queried application servers, it compares their content. If a majority agree, the result is forwarded to the client.

An Intrusion Detection System (IDS) monitors both the external and internal networks [6]. Other monitoring mechanisms include a challenge-response protocol for verifying proxy and application server integrity, and runtime monitors based on formal specifications that detect anomalies in the proxy [3].

An *alert manager* receives inputs from these monitoring mechanisms. It uses this information, together with the content agreement results, to decide whether the current regime should be made stronger or weaker.

3 Detecting and Preventing Worm Spread

Some application servers may be running vulnerable COTS operating systems and applications. A number of complementary mechanisms will prevent them from being infected with a virus such as Code Red:

*This research is sponsored by DARPA under contract number N66001-00-C-8058. The views herein are those of the authors and do not necessarily reflect the views of the supporting agency.

Sanitizing Requests: The proxy server normalizes and sanitizes requests, reducing the chance that a syntactically suspect query reaches an application server.

IDS Subsystem: The IDS subsystem is configured to detect known attacks; however, we cannot assume that all novel attacks will be detected.

Diversity: The application servers are chosen to maximize diversity of implementation and operating systems (this, however, is traded off against the cost of installing and maintaining the system). The servers are rebooted periodically from read-only media, to make it more difficult for a determined attacker to successively compromise each vulnerable machine until a majority are compromised.

Other mechanisms protect against the *effect* of any successful attack or infection:

Content Agreement: If a worm or virus succeeds in altering the content served by one of the application servers the content agreement will filter this out, provided a majority of the servers are still uncorrupted.

Private Network Configuration: Even if a vulnerable application server were infected, the spread of the worm would be prevented by the configuration of the internal network, which follows the principle of least privilege.

Because the expected traffic patterns for the internal network are few, it is possible to specify them completely. In particular, for TCP traffic, the expected patterns are as follows:

- The proxy initiates HTTP and challenge-response connections to the application server hosts.
- The network intrusion detection appliance initiates TCP connections to the proxy for alert reporting.

A few Snort [7] rules are sufficient to detect any violation of the above requirements. Note that even though Snort is commonly classified as a signature-based IDS, in this case we can use it to perform, in effect, specification-based intrusion detection.

If any of the application servers tries to initiate an outbound connection, the IDS on the private net detects this and raises an alarm; the proxy itself blocks the connection. Alternatively, the network can be physically configured so that each application server is unreachable from all the others, and traffic that does not meet the above specification is blocked at the source.

Hardened Proxy: The above security mechanisms, of course, are of little use if the proxy server itself can be infected. Since it has access to the outside world, it is also a potential source of infection for third-party machines. However, we claim that the hardened proxy is

less likely to be susceptible to virus or worm infection: it is running a small, customized code base on an operating system configured for maximum security. For instance, the proxy code is compiled with Stackguard [2], a buffer-overflow prevention mechanism, and all non-essential OS services are turned off.

4 Conclusions

There are two main reasons that the DIT system is less vulnerable to worm-based attacks. First, the system provides defensive depth against attacks in general. The second reason is the application of the principle of least privilege at the network and architectural level: The application servers do not need to communicate with any machine other than the proxy and, even then, the network traffic is restricted to the minimum necessary.

References

- [1] MS-SQL server worm, Jan. 2003. CERT Advisory CA-2003-04.
- [2] C. Cowan et al. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *Proc. 7th USENIX Security Conference*, pages 63–78, Jan. 1998.
- [3] J. Levy, H. Saïdi, and T. E. Uribe. Combining monitors for run-time system verification. *Electronic Notes in Theoretical Comp. Sci.*, 70(4), Dec. 2002.
- [4] D. Moore, V. Paxson, S. Savage, and C. Shannon. The spread of the Sapphire/Slammer worm. Technical report, Silicon Defense, 2003.
- [5] R. Permech and M. Maiffret. .ida “Code Red” worm. Security Advisory AL20010717, eEye Digital Security, July 2001.
- [6] P. Porras and P. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the 20th National Information Systems Security Conference*, pages 353–365, Baltimore, MD, Oct. 1997.
- [7] M. Roesch. Snort: Lightweight intrusion detection for networks. In *USENIX LISA’99*, Nov. 1999. www.snort.org.
- [8] A. Valdes et al. An architecture for an adaptive intrusion tolerant server. In *Security Protocols Workshop*, LNCS. Springer-Verlag, 2002.