Security Patterns and Evolution of MTA Architecture

[Extended Abstract]

Munawar Hafiz
University of Illinois at Urbana-Champaign
201 N Goodwin Ave
Urbana, IL 61801, USA
mhafiz@uiuc.edu

ABSTRACT

In this paper, we describe the evolution of architecture of Mail Transfer Agents(MTA). We consider the design of MTA as a sequence of design decisions [10]. Many of the design decisions are examples of security patterns. Thus, this paper is another example of how patterns generate architecture [5].

Categories and Subject Descriptors: D.2.11 [Software Engineering]: Software Architectures—Patterns

General Terms: Design, Reliability, Security.

Keywords: Architectural Requirements, Mail Transfer Agent (MTA), Security Patterns.

1. INTRODUCTION

A mail transfer agent (also commonly called a mail server) or MTA is a computer program that transfers electronic mail messages from one computer to another. The most popular MTA is *sendmail* written by Eric Allman. Other popular Unix compatible MTAs are D.J. Bernstein's *qmail*, Wietse Venema's Postfix, *exim* developed in Cambridge University, *MMDF* (Multi-channel Memo Distribution Facility) for SCO Unix, *Smail 3.x*, *Courier* and *ZMailer*.

The poster presents the architecture of four leading MTAs - *sendmail* [3], *qmail* [2], Postfix [1] and *sendmail X* [4]. It reveals the evolution of MTA architecture in accordance with the changing requirements. The early *sendmail* architecture was perfect in its time because the key architectural requirement was flexibility and the architecture supported a lot of diverse protocols accordingly. The requirements of MTA changed with the introduction of internet and security and reliability became the key issue. The problems of *sendmail* architecture were illustrated by the constant detection and fix of the security vulnerabilities.

The architecture of *qmail* is motivated by the series of security breaches in *sendmail*. However, *qmail* is not only more secure than *sendmail*, it is also more efficient and easier to understand. The security of *qmail* is based on a few patterns and understanding its architecture can help people make other applications secure.

Postfix has gained popularity as an MTA because it has the same interface like *sendmail*, yet it does not have problems with security and reliability. The architecture of Postfix closely follows the design principles of *qmail*. A lot of security patterns of the *qmail* architecture are found in Postfix. Additionally, Postfix improves performance. It shows better performance than *qmail* and *sendmail* in benchmark tests. In many cases Postfix uses the same security patterns like *qmail*. However, there are design areas where Postfix uses different patterns and these design choices are motivated by performance. Along with that, there are mechanisms adopted at different places in Postfix architecture to improve performance. Thus the Postfix architecture provides a good example of consideration of performance along with security and reliability.

sendmail X is targeted to provide a *sendmail* system that is secure and reliable. The development of *sendmail X* is still under way. With *sendmail X* architecture, the evolution goes full circle as *sendmail X* design decisions come from the best practices in *qmail* and Postfix. This illustrates the adaptation of MTA architecture in response to feedback generated by usage scenarios and evolving requirements. Evolution of architecture based on feedbacks is absolutely vital to ensure that it withstands challenges imposed by the new requirements.

2. *sendmail* ARCHITECTURE

The *sendmail* program pre-dates the internet evolution. It was the first widely used MTA. The main architectural requirement of *sendmail* was flexibility. Because of absence of Internet standards, *sendmail* was intended to support a lot of diverse protocols. One key requirement that was not considered during the design of *sendmail* was security. With the wide-spread use of Internet, security became the critical factor.

sendmail runs as one big process with root privilege. *sendmail* process is configured through a complex configuration file. Another important part in the architecture is the mail queue. The monolithic *sendmail* process runs with maximal privilege level (root privilege), because some of its tasks require root privilege.

With such a monolithic architecture, if one malicious user gets control, he can use the root privilege to get control of the whole system.

3. *qmail* ARCHITECTURE

An alternative to the *sendmail* way of implementing everything as a single process is to implement the components of MTA as separate processes. This security pattern is called Compartmentalization [11] and *qmail* follows this pattern by partitioning the task among several processes. The

OOPSLA'05, October 16–20, 2005, San Diego, California, USA.
ACM 1-59593-193-7/05/0010.

processes that does not require root privilege can now run with a lower privilege level. Again the processes run under different users (Distributed Delegation pattern [7]) and therefore security compromise in one process does not impact other processes. The processes also verify their inputs before working on them (Trust Partitioning pattern [8]).

The mail queue and the mailbox are maintained by the MTA and updates in these storages have to be reliable. The reliability issue comes from the fact that multiple processes try to access these shared storages, and the storage should remain in a consistent state. The Unique Atomic Chunks pattern [6] ensures reliable update of mail queue and mailbox. This pattern is also adopted by Postfix, *sendmail X* and other MTAs for implementation of reliable mail storages.

These patterns are not new for *qmail*, but *qmail* is an example of how to use them effectively. One of the key principles of *qmail*'s architecture is Defense in Depth [11], which means that *qmail* does not depend on any single pattern to achieve security, but has several layers of security. First, the way it is divided into modules tends to decrease the damage caused by security break-ins, and eliminates some kinds of errors. The module decomposition makes each module simpler, so it can be inspected for correctness. It makes multiprocessing more efficient. The way that *qmail* uses the file system makes queuing and delivering the mail more reliable. The low-level coding patterns eliminate important classes of errors such as buffer overflows. The result is an MTA architecture that is simpler and more reliable than that of *sendmail*.

4. POSTFIX ARCHITECTURE

Postfix follows the *qmail* design principles by compartmentalizing the processes. It is implemented as a resident master server that runs Postfix processes on demand. The child processes are pre-forked for performance reasons. This reduces process creation overhead. However, the problem with pre-forking is that it creates a worker pool and if the pool contains daemon processes then some malicious attacker has more chance of utilizing one compromised process to infect other processes through some shared channel. To avoid this, the child processes are created up to a configurable number, are re-used for a configurable number of times, and are killed after a configurable amount of idle time or after processing a configurable number of requests. Thus the Secure Pre-forking [7] pattern limits the vulnerability associated with daemon processes. In addition, the Postfix approach has the advantage that a service such as address rewriting is available to every Postfix component program, without incurring the cost of process creation just to rewrite one address. These contribute to the better performance of Postfix.

qmail processes run under different owners. Postfix reduces the number of process owners by running the processes under the same user. Apparently, this would mean that the if one process is compromised then other processes would also be under threat. However, the common user has very low privilege level. Again, Postfix uses the chroot Jail [7] pattern to run the processes in a contained environment to limit the exploits. Finally, the Postfix processes validate the inputs before working with them.

5. *sendmail X* ARCHITECTURE

The new generation of *sendmail* is *sendmail X*. *sendmail X* is a completely new design and is not an evolution of previous versions of *sendmail* (*sendmail* version 8).

sendmail X architecture is completely different from *sendmail*. It does not have a single root process. Instead, it follows the architecture of Postfix very closely. *sendmail X* processes are compartmentalized [11] according to their functionality. Most of the *sendmail X* design decisions are influenced by the architectural principles of *qmail* and Postfix.

Currently, one of the major threats in public email system is handling spam and phishing type attacks. *qmail* architecture does not have spam handling support built in it. It was added as separate feature later. This is because in 1997, when *qmail* was written, spam handling was not a major issue. Postfix, which came later, uses internal and external tables to detect and remove spam. *sendmail X* has elaborate internal and external mechanisms built in for handling spam. This illustrates that *sendmail X* architecture is not a copy of *qmail* and Postfix, but the design is evolving in response to the changing requirements.

6. CONCLUSION

The poster provides a synopsis of the evolution of MTA architecture in response to changing requirements. It shows several security and reliability patterns that contribute to making the MTA secure.

The architectural quality requirements are not always orthogonal and the final design decision is a tradeoff between the requirements. Again, in the architectural life cycle, experience and best practices impact future architecture [9]. This is clearly evident in the *sendmail X* 's adoption of best practices of *qmail* and Postfix.

7. ACKNOWLEDGMENTS

This poster is based on the findings from my masters thesis titled 'Security Architecture of Mail Transfer Agents' [7] with Professor Ralph Johnson.

8. REFERENCES

[1] Postfix home page. *Maintained by Wietse Zweitze Venema.* `http://www.postfix.org`.

[2] *qmail* home page. *Maintained by Daniel Julius Bernstein.* `http://cr.yp.to/qmail.html`.

[3] *sendmail* home page. *Maintained by Sendmail Consortium.* `http://www.sendmail.org`.

[4] C. Assmann. *sendmail X*: Requirements, architecture, functional specification, implementation, and performance. Technical report, *Sendmail Inc.*, September 2004.

[5] K. Beck and R. Johnson. Patterns generate architectures. *Lecture Notes in Computer Science*, 821:139–??, 1994.

[6] M. Hafiz. Unique atomic chunks: A pattern for security and reliability. In PLoP 2004 Proceedings, 2004.

[7] M. Hafiz. Security architecture of mail transfer agents. Master's thesis, *University of Illinois*, Urbana, IL, June 2005.

[8] M. Hafiz, R. Johnson and R. Afandi. The security architecture of *qmail*. In PLoP 2004 Proceedings, 2004.

[9] R. Kazman, M. Klein and P. Clements. ATAM: Method for architecture evaluation. Technical report, *CMU/SEI-2000-TR-004*, August 2000.

[10] D. Parnas and P. Clements. A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*, 19(2):251–257, February 1993.

[11] J. Viega and G. McGraw. *Building Secure Software - How to Avoid Security Problems the Right Way.* Addison-Wesley, 2002.