International Journal of Students Research in Technology & Management
Vol 1(3), May 2013, ISBN 978-93-83006-01-4, pg 353-360

INLINE PATCH PROXY FOR XEN HYPERVISOR

Neha Rana, Pankaj Singhal Dnyanesh Kulkarni Pratik Bhangale

Department of Computer Engineering

Pune Institute of Computer Technology, University of Pune, India.

neharana226@gmail.com singhal.pankaj@outlook.com

kdnyaneshv@gmail.com pratik.bhangale1992@gmail.com

Abstract

Application softwares running on end user or application servers are always prone to various attacks. These attacks not only harm applications but also waste network resources. Solutions to these problems are available as patches since a long time. Generally, people have been reluctant to patch their systems immediately, because patches are perceived to be unreliable and disruptive to apply. To address this problem we propose an inline patch proxy solution for Xen hypervisor.

Inline solutions provided are vulnerability-specific, exploit-generic network solutions installed on end systems. The Inline patch module examines the incoming or outgoing traffic, vulnerable to applications, and removes these vulnerabilities to maintain secure traffic. The motive of the idea is to reduce the time difference between the release of a software patch and its actual deployment. Currently patching is promised by software developers, generally within hours (Varies as per the service level agreements) of occurrence of a vulnerable attack. The proposed idea is based on the reducing this time gap to a few seconds by placing the proposed module within the system. For unexposed attacks, time is needed to create new signatures which are generated in update server and pulled by the software running on host.

Keywords: Inline Patching, Vulnerability Signature, Network Filter, Generic Protocol Analyzer, Xen Hypervisor, PF Ring

I. INTRODUCTION

In earlier days Physical servers were used to host web or distributed applications. As number of such applications has increased tremendously these days, services and resources provided by physical servers are very limited and insufficient. Concept of Virtual servers arises as a solution to the above problem. A hypervisor is a software program residing over physical server and interacting with system hardware to provide ability of hosting many virtual servers over a single physical server.

Xen Hypervisor is an emerging platform widely used by giants like Amazon, Google, Microsoft, Rackspace, GoGreed and many more. In the past decades various applications running on guest domains of Hypervisor have been vulnerable to different types of attacks like DoS, SQLi and others. This gives rise to the need for efficient handling of network traffic. Solutions to such attacks have been

available as patches since long. But conventional approaches to patching have a number of significant disadvantages:

- *Resource-consuming*: The traditionally available solution provides more security patches with requirement of more human resources, also consuming more time and money.
- *Vulnerability*: Patching later as done in such traditional methods are more prone to outbreaks and Security policy violation.
- *Disruption*: Installing a patch typically involves re- booting, at the very least, a particular host service, and possibly an entire host system. An administrator for whom system and service uptime are crucial may therefore be unable to tolerate the required service or system disruption.
- *Unreliability*: Software patches are typically released as quickly as possible after vulnerability is discovered, and there is therefore insufficient time to do more than cursory testing of the patch.
- *Unawareness*: An administrator may simply miss a patch announcement for some reason, and therefore be unaware of it, or have received the announcement but neglected to act on it.
- *Multiple patches*: Multiple patches may be created for a single modification in a program. The program to be patched might also have different flavours of some executable code module, for each of which, the process of patch creation and distribution is repetitive. It becomes the user's job to select and apply the patch for the appropriate flavour of the module.
- *Size of patches*: Some patches may be considerably large in size, making them difficult to store and distribute. This problem multiplies for patches having multiple flavours.

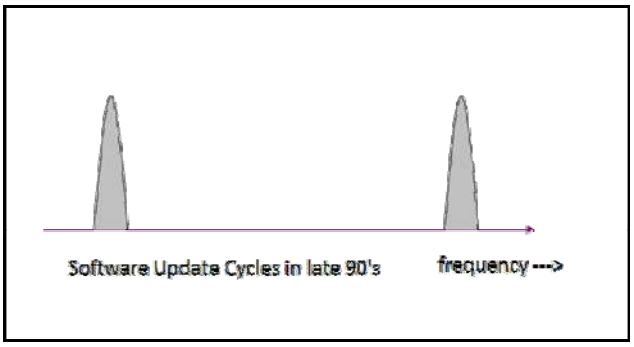

Fig. 1. **Frequency of updation – Earlier**

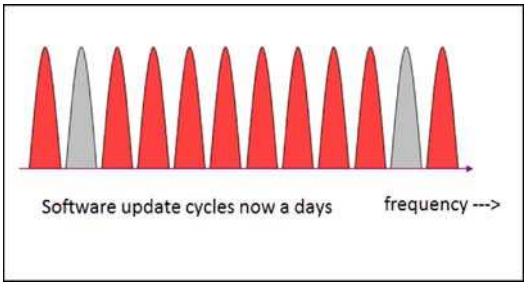

Fig. 2. **Frequency of updation – Earlier**

- *Virus attacks*: Applying patches requires a user to be logged into an administrative account having modification permissions, enabling viruses that seek to modify aspects of the target computer system, using liberal permissions, to attack.[7]

Currently vulnerabilities are handled through software patching which promises to solve the problem within the time specified in service level agreements (usually in the order of hours or days), after the occurrence of an attack. Reduction in the time required for patching is the urge in today's network communication. To address this problem we propose an inline patch proxy solution for Xen hypervisor**,** based on the idea of reducing the time difference between the release of a patch and its deployment on the end system, by building an inline module which would reside within the system and handle different attacks based on signature detection of packets.

Since Sept.19, 2012, the websites of Bank of America, JPMorgan Chase and PNC Bank (Fortune 500), have all suffered day-long slowdowns and been sporadically unreachable for many customers. The attackers who attacked Bank of America first, went after their targets in sequence.

According to recent published reports SQL injections are the top attack vector making up 19% of all the security breaches examined by WHID. Similarly, in the "Breach Report for 2010" released by 7Safe the same year, a whopping 60% of all breach incidents examined, involved SQL injections.[6]

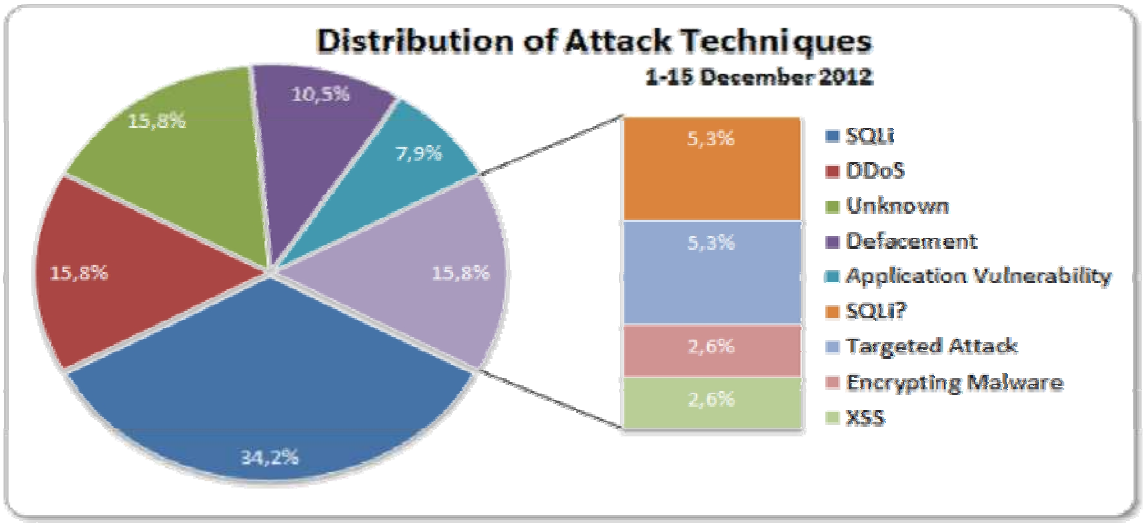

Fig. 3. **Distribution of attacks[8]**

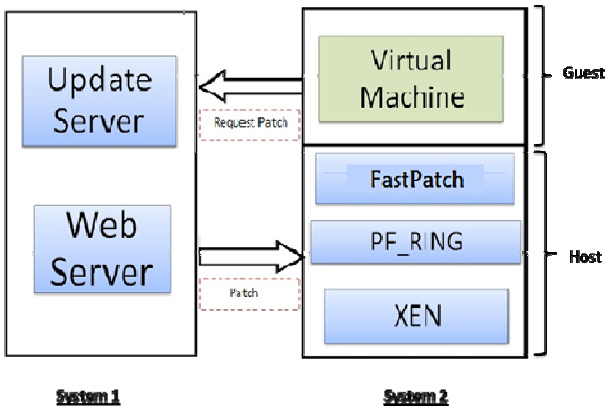

Fig. 4. **System Architecture**

II. DESIGN

A. Basic Overview:

The basic design that we present in this paper aims to securely pass patches to the guest OS on which application servers are running. Without patching, Xen hypervisor is vulnerable to attacks and thus all the application servers running on it. In general scenario, incoming packets arrive on Dom-0 (Host OS) which directs them to PF_Ring. PF_Ring is a high speed packet-capturing packet-filtering tool, which is used as an interface to capture the incoming packets. As PF_Ring is installed on Dom-0 the traffic going to and from all virtual servers is monitored centrally. Same logic goes for the applied patch also. Once a patch against a specific vulnerability is linked with PF_Ring, all the virtual servers running on the hypervisor are secured against that vulnerability. Hence PF_Ring is used to identify the parameters of the packet and pass them to our module (FastPatch). FastPatch analyzes the signature of the packet and searches for it in the database, providing security level information. The threat category is identified and a patch is generated for the vulnerable packet detected. This patch is passed back to PF_Ring which securely passes it to the application server running on guest OS.

Functionalities of the components on the hypervisor system (system 2) are as follow:

1. *Xen:*
   - supporting virtual application servers
   - interaction with system hardware through Dom-0
   - forwarding network traffic to PF_Ring
2. *PF_Ring:*
   - packet capturing
   - packet filtering
   - passing packet signature to FastPatch module
   - forwarding packet to its destination
3. *Fast Patch:*

- packet signature analysis
- determining type of vulnerability (if any)
- patch selection
- discarding malicious packets

4. *Virtual machine*:
   - hosting application server
   - hosting application database
   - patch request

B. Virtualization:

Several guest machines running on the Hypervisor are vulnerable to attacks. Instead of independently patching these virtual machines, the hypervisor is patched, thus automatically securing all the virtual machines against the vulnerabilities. This reduces the load of the virtual machines to secure themselves.

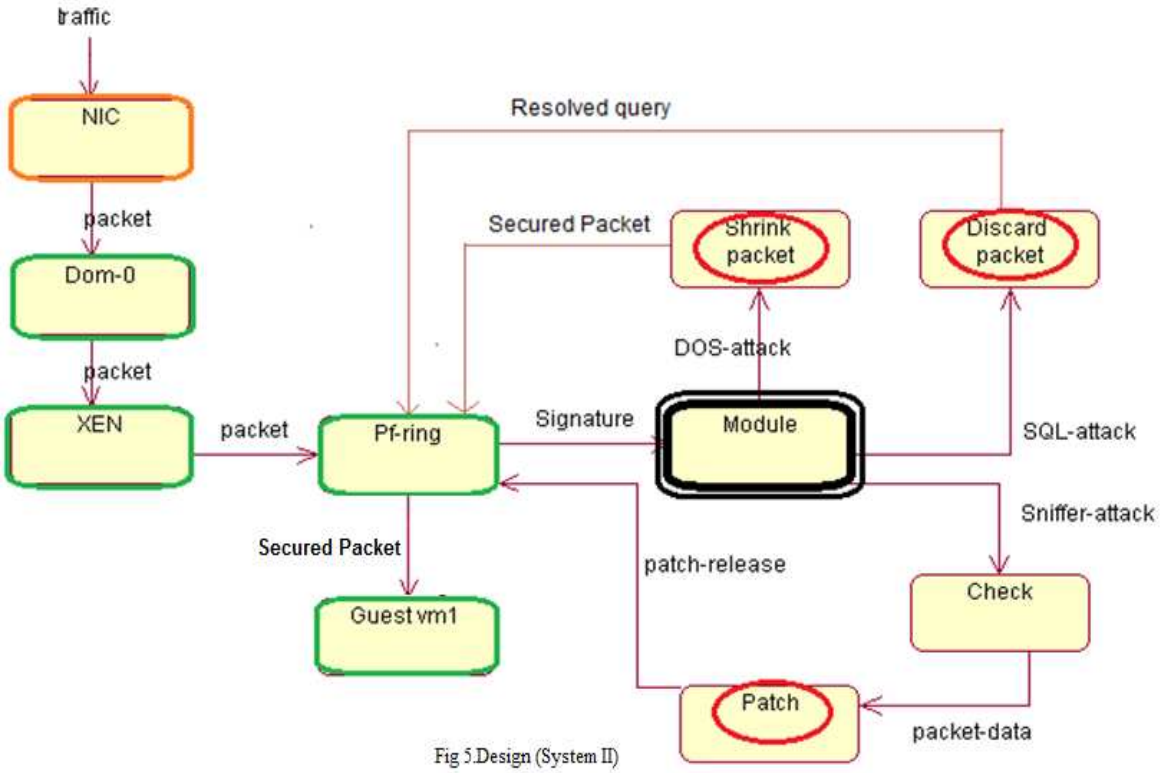

Fig 5.Design (System II)

C. Web Server:

The Web server hosts our website. The main purpose of this web site is to provide the user with an interface to the update server. Users need to register to get access to the patching facilities provided by update server. Users can view and patch their system against vulnerabilities as they desire. PRO-FTPD details of the user are taken in order to provide secured transactions of the patches.

D. Update Server:

Update server is the area where patches for the identified vulnerabilities are stored. Selective patches are sent to the user depending on his request. Whenever a new vulnerability is detected, a new patch is created, compiled and stored here. Modules residing in the Update server generate code which are compiled and stored as kernel objects and sent to the users as patches. Working of some of the modules is as follows:

- *SQL injection Attack Handler*: In SQL injection generally the attack source is the insertion of a malicious query in the URL. Various criteria are applied to incoming packets to look for the existence of such query in the packet URL. Packets having such URLs are discarded and the rest are forwarded to the destination guest OS.
- *Sniffer Attack Handler*: Packet with signature matching sniffer attack criteria is taken as input and packet data is analyzed. After analysis measures are taken to secure client – server communication line/path. Various filtering tests are applied to detect the third person trying to intersect the communication. Detected intruder or solutions to secure line/path are returned to user's system and are linked with PF_Ring.
- *DoS Attack Handler*: Packet with signature matching DoS attack criteria is taken as input and packet data is analyzed. After analysis measures are taken to reduce http packet size. Size is reduced in such a way that no loss of data is done. In case of DoS packet flooding, the IP of the machine sending these packets is recognised and that source IP is blocked. Depending on the type of patches requested, patches are returned to user's system and are linked with PF_Ring.

III. IMPLEMENTATION

The Idea proposed in this paper is still under development stage. Till date implementation of handling some types of SQL injection has been done. These types include Code Injection, String Manipulation, Login attacks and Timing attacks.

In this Implementation attack handling module works in collaboration with PF_Ring. A sample website is built to test these attacks. Apache server and MySQL server are used for this implementation.

The details of the results obtained so far are-

Table I. Packet Details

Packet Details	
Name	***Description***
Source IP	IP address of the host system i.e. attacker
Destination IP	IP address of the server
Destination Port	Port no of the application running on server
Transport Layer Data	Layer 3 Packet Data
Network Layer Data	Layer 4 Packet Data
IP version	IP version of the packet (4/6)

Table II. Result Statistics

Attack Name	***No. of Queries Fired***	***Detection***
SQL Injection	10	100%

IV. FEATURES

- *Inline patching of the system*: FastPatch module resides in the host system and patches it inline.
- *Update server*: New vulnerabilities are handled at the server side.
- *Auto-patch*: Updates available will be automatically transferred and installed.
- *No Restart needed*: Patches are linked dynamically to PF_RING and thus applied instantly.

V. FUTURE SCOPE

The proposed idea can be further extended to handle worm attacks. The current system will not be able to identify unrecognized attacks in the first place. In future it can be improved to identify attacks even with no prior knowledge about them.

VI. CONSTRAINTS

The system generated is for Linux OS and not for other Operating Systems like IOS, Windows OS etc as dom0.

- On x86 Xen with a Linux dom0 runs on Pentium II or newer processors.
- If the application is attacked by some worms it won't be detected unless application crashes.

- The secure module is not developed to handle vulnerabilities for IIS Windows and X5 web servers.
- Proposed solutions can be applied to specific network attacks only.
- Software needs to be updated as new vulnerabilities are detected.
- Cannot handle previously unrecognized attack.

VII. CONCLUSION

Inline Patch Proxy System provides secure patching and filtering of packets travelling from network to application side. The proposed system reduces the time required for patching vulnerable packets against the traditionally used software patching. It aims at providing an open source infrastructure for malicious packet detection and protection from network exploits, also giving an alternate solution for software patching against large scale network attacks. Unlike software patches, it will be less prone to outbreaks and security policy violations since this module shall reside in the end host system. If implemented it will be a more reliable and undisruptive solution to network exploits for Xen Hypervisor framework.

VIII. ACKNOWLEDGEMENT

We take this opportunity to express our profound gratitude and deep regards to PICT Linux User's Group (our project sponsors)and our guide Mr. Gaurav Suryagandh for his exemplary guidance, monitoring and constant encouragement throughout.

IX. REFERENCES

1. W. A. Arbaugh, W. L. Fithen, and J.McHugh. "Windows of Vulnerability: a Case Study Analysis". IEEE Computer,2000.
2. Helen J. Wang, Chuanxiong Guo, Daniel R. Simon, and Alf Zugenmaier "Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits". Microsoft Research ,2004.
3. V. Capretta, B. Stepien, A. Felty, and S.Matwin, "Formal correctness of conflict detection for firewalls," in FMSE '07:Proceedings of the 2007 ACM workshop on Formal methods in security engineering, 2007, pp. 22–30.
4. Robert Bunge, Sam Chung, Barbara Endicott-Popovsky, Don McLane "An Operational Framework for Service Oriented Architecture Network Security".IEEE, 2008.
5. Zhiyun Qian, Z. Morley Mao, Ammar Rayes, David Jaffe." Designing Scalable and Effective Decision Support for Mitigating Attacks in Large Enterprise Networks".Springer,2012.
6. forums.cnet.com/7726-6132_102-3253715.html
7. Anthony Blumfield, Gilad Golan, Jason Garms, Saud Alshibani. "Efficient patching". United States Patient, 2012
8. hackmageddon.com/tag/sql-injection