Hypervisor-based Security Architecture for Validating DNS Services (Poster)

Dilshan Jayarathna, Udaya Tupakula, Vijay Varadharajan
Advanced Cyber Security Research Centre, Macquarie University, Sydney, Australia
{dilshan.jayarathna, udaya.tupakula, vijay.varadharajan}@mq.edu.au

Abstract

Domain Name System (DNS) is one of the critical services in the current Internet infrastructure. However DNS is vulnerable to a range of attacks. One of the fundamental weaknesses with the existing DNS protocols is that the request and response messages are transmitted on the network as plain text. This paper addresses important threats related to Doman Name System (DNS) using a hypervisor based security architecture. The proposed architecture leverages the hypervisor visibility of the virtual machines' traffic flows to monitor and utilise Virtual Machine Introspection (VMI) techniques to inspect and restore data. It also uses inbuilt snapshot/restore capabilities of the hypervisor to completely restore virtual machines if required. Objective of the proposed architecture is not to actively prevent attacks, but provide a means of identifying different attacks by passively monitoring DNS related conversations coming in and out of virtualised system hosting the DNS. Our model can alert the external monitoring agent(s) or security administrator and actively restore the system if the attack has already compromised the DNS.

Keywords: DNS, Security attacks, Virtual Machine Introspection.

1 Introduction

Domain Name System (DNS) is one of the important services in the current Internet infrastructure. DNS (Mockapetris 1987a,b) is a system for naming computers and network services that is organized into a hierarchy or a tree structure of domains. At the top of the hierarchy is the root, a single domain represented by a dot (".") in a domain name. Below the root are top level domains (TLDs) (e.g. .com, .edu, or .au) and the next level consists of enterprise level domains (ELDs) owned by individual entities. A DNS zone refers to an administrative entity in the DNS that provides DNS services for a group of domains under one ELD. Multiple zones or sub-zones may exist within an ELD. DNS is an essential component of the functionality of the Internet, where the primary purpose of the DNS is to translate easily memorised/readable Internet or private network domain and host names to IP addresses.

DNS is vulnerable to different types of attacks. In this paper, we propose hypervisor or virtual machine monitor (VMM) based techniques for counteracting DNS attacks. A virtual machine monitor (VMM) (Rosenblum and Garfinkel 2005, Barham et al. 2003) is a software abstraction layer that enables multiple virtualised operating systems to run concurrently on a single physical computer. An instance of a virtualised operating system along with its applications is referred to as a virtual machine (VM). Use of VMs in enterprise, government, and consumer applications is becoming increasingly important due to the diverse security requirements and the different levels of trust associated with different applications, systems and devices where VMs can provide strong isolation between different environments.

The design issues involved in developing viable trusted systems based on VMMs pose several technical challenges such as scalability in terms of both modelling and architecture as well as its application to various scenarios such as peer to peer, grid and cloud computing, and applications in e-commerce. While most of the VMMs provide application security from the network and host perspective, they are lacking in control over the contents specific to the critical applications running on VMs. This paper proposes techniques for validating DNS which is one of the critical applications in the current Internet infrastructure.

The paper is organised as follows. Section 2 discusses the various types of attacks against the DNS in detail and develops an attacker model. Section 3 describes our proposed security architecture to counteract the security attacks against the DNS considered in Section 2. Section 4 describes our implementation of the security architecture and presents an analysis of the results. Finally, Section 5 concludes the paper.

2 Attacker Model

There are several well-known attacks against DNS. In this section, first we consider these attacks in detail and then formulate the attacker model before developing the security architecture.

Let us consider a sample zone or domain with an authoritative DNS server and caching DNS server as shown in Figure 1. The authoritative name servers are original sources for all the DNS resolutions for the zone. The client hosts within the domain and any of the external hosts can make resolution requests for the services within the domain. The caching servers perform resolutions for the stub resolvers. For example, when the client machines request for resolution that does not belong to the domain, then the caching servers can initiate further queries to other DNS servers and cache the received response. The received information is stored in the cache until the TTL is valid or based on queuing mechanisms (such as FIFO

or least resolved) and load on the caching server. Now let us consider the attacks that are addressed in this paper.

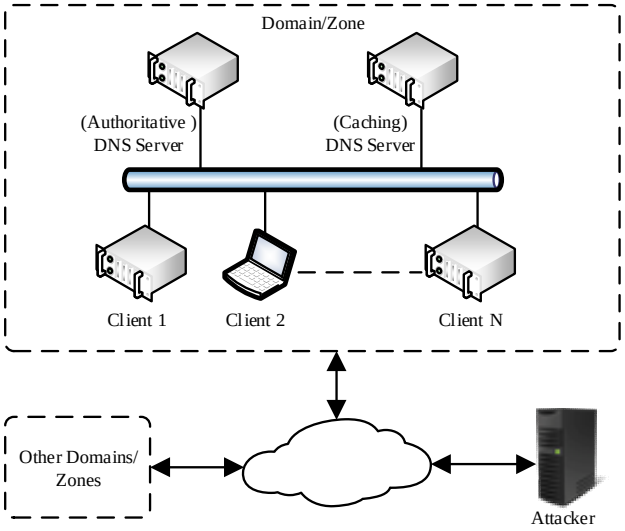

Figure 1: Attacker Model

In the first case, we consider the case of compromise of the DNS server. Both, the authoritative server and the caching server can be compromised by the attacker. For example, the attacker can exploit some weakness in the operating system or the DNS application and alter the authoritative records or install malicious software such as rootkits. Now the attacker can use these malicious programs to provide false resolution for any requests received by the compromised DNS servers.

In the second case we consider how the attacker can exploit the DNS's usual behaviour of sending an entire query or response in a single unsigned, unencrypted UDP packet which makes these attacks particularly easy. In this case an attacker eavesdrops on the communication to initiate a man-in-the-middle attack when the caching DNS server makes a query to authoritative servers in other domains. For example, consider that the Client 1 in Figure 1 makes a request to the caching server for name resolution of www.google.com. The caching server initiates a query to the authoritative server in the Google domain (what is the IP address of www.google.com?). The attacker intercepts the DNS query and responds with forged information before the Google Authoritative DNS server responds with the correct IP address of www.google.com. The caching DNS server considers the IP address from the attacker as that of www.google.com and also forwards it to the Client 1. Now the Client 1 connects to host with IP address issued by the attacker (assuming that it is connecting to www.google.com), but reaching a host controlled by the attacker, which may be hosting a fake site. When the Google authoritative name server responds with the correct IP address for www.google.com, it is ignored by the caching DNS server since it has already received a "valid" spoofed response from the attackers system. However note that in this case, the attacker should be able to capture the query from the caching DNS server.

In the third case, the attacks are similar to second case but it does not require the attacker to capture the query from the caching DNS server. The query ID in DNS packet structure is used by the DNS servers to relate the query messages with the received response messages. Since there are 16 bits in the query ID, the attacker just needs to generate 65535 response packets with increasing query ID to poison the caching DNS server. Since most of the DNS servers use standard ports (port 53), directing the responses to these standard ports will be successful in some cases to poison the cache of the DNS server.

3 Securing DNS Services

Now we propose techniques to deal with the attacks considered in the previous section. First we present a high level overview of our model and then describe the architecture components in detail.

3.1 Design Overview

The proposed solution monitors the DNS server communication at the hypervisor level and captures the DNS requests and corresponding responses. Then it validates the responses against multiple external DNS server using a secure communication method to determine the accuracy of the DNS responses. Inaccurate response can be bogus, incomplete or simply incorrect due to an administration error. It also validates the running processes of the monitored DNS server to determine the integrity of the DNS server by inspecting VM memory. Depending on the decision of the validation process, the reason for wrong response can be one of the DNS attacks described in Section 2.

Upon validation of the accuracy of the DNS response and the integrity of the DNS server, it will alert the security administrator, restore the affected files or perform a complete system restore using a VM snapshot of the last known good configuration.

3.2 Architecture

Generally, in a standalone computing environment, the operating system has the full control over the execution of all applications and works as an interface between the applications and the hardware. In a virtual environment, VMM controls the access to physical resources while each guest operating system manages its own applications. Our architecture makes use of the VMM capabilities to ensure secure operation of the DNS.

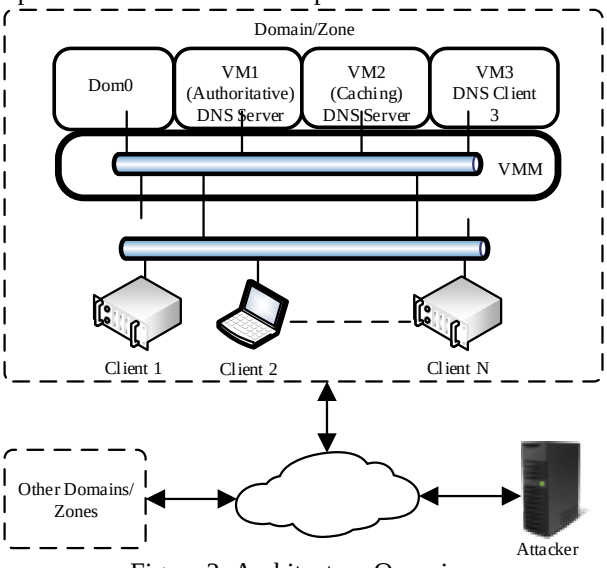

Figure 2: Architecture Overview

Figure 2 shows the architecture to identify DNS attacks. For simple presentation, we consider that the authoritative DNS server and the caching DNS sever are implemented on the same VMM. DNS client (*e.g.* VM3) can also be in the same VMM. However note that operation remains same even if the servers are implemented on different VMM.

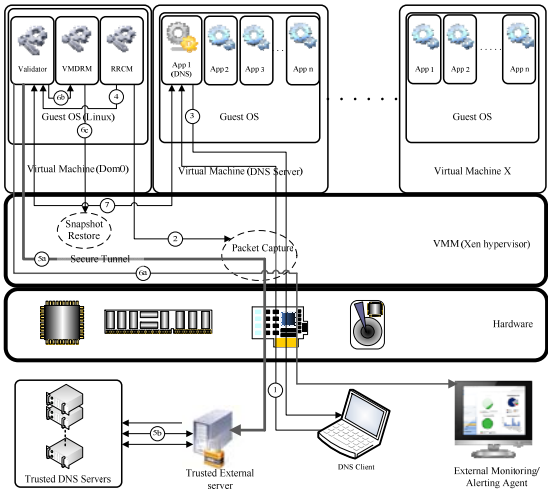

Figure 3: DNS response validation mechanism

The proposed architecture leverages the hypervisor level visibility to monitor traffic flows in and out of the virtual machines. As shown in Figure 3, our architecture consists of three logical components, which are strategically placed in Dom0 in order to make use of the functionalities such as snapshot/restore already available at hypervisor layer. We also assume there is a pool of trusted DNS servers which support secure communication (such as DNSSEC and IPsec) of DNS data. The main reason for using different secure communication techniques is to address the lack of wide deployment of DNSSEC. We also assume that the trusted servers are not compromised and provide correct name resolution. The trusted servers can be hosted by ISPs and/or authoritative name servers of popular domains.

3.3 Logical Components

3.3.1 Resolution Capture Mechanism (RCM)

RCM captures all traffic flows to and from virtual machines by simply sniffing the all virtual bridge interfaces of the hypervisor. It filters all DNS related traffic and records all DNS requests and corresponding responses while discarding the other traffic. It keeps track of to which DNS server (authoritative or caching) the query was directed and the source of the request. The RCM also keeps track of the queries and corresponding responses from virtual DNS server to external DNS servers. This information is used to determine whether the inaccurate responses are provided by the monitored DNS server or external DNS server or both.

3.3.2 Validation and Attack Detection (VAD)

VAD maintains a copy of records from the local authoritative DNS server. This is used as reference to determine if the hosted authoritative DNS is compromised. For validating caching server resolutions, VAD looks up n number of trusted DNS servers to obtain correct resolution for the FQDN queries captured by RCM. Then the VAD compares the trusted server resolution responses against the response sent by the monitored virtual DNS server or the external DNS server. Note that unlike virtual DNS servers which operate using standard insecure UDP protocol, this component makes use of secure communication using DNSSEC and IPSec VPN tunnel to lookup pool of trusted DNS servers to avoid eavesdropping or man-in-the-middle attacks. Although there can be time delays for such resolutions, this will not impact our model since we are not performing active prevention of attacks.

The VAD is also used for monitoring the runtime state of the virtual DNS server and detect the compromise of the DNS servers. If the attacker exploits vulnerabilities in the DNS application or the operating systems in the virtual machine and install rootkits, such attacks are also detected by the VAD component. At regular intervals VAD checks what processes are running on the monitored DNS server. We have analysed the default processes for The Berkeley Internet Name Daemon (BIND) 9.8.1-P1 running as the DNS on Linux server with 3.5.0-23 kernel. There are total of 59 processes in the clean state installation of DNS on Linux. Figure 4 shows partial list of the processes running in the DNS server virtual machine and the highlighted process named which is located in specified path (/usr/sbin/named) is responsible for receiving and responding to the client resolution requests). We maintain the details of these 59 processes in the VAD and use this as reference to monitor the DNS server during runtime. We make use of the VMI interface to directly extract the runtime information from the monitored DNS. Figure 5 shows the partial list of processes and their corresponding address location obtained by the VAD component. If there is any variation from the default process list then the DNS sever is considered to be compromised. Note that it is not an easy task for the attackers to alter the process list obtained by the VAD since it is running in the VMM.

Now let us consider how the VAD can deal with the ID guessing attacks. Although the attacker can successfully poison the caching DNS server by sending multiple responses with incremental query ID, this will result in anomaly at the victim DNS server. The VAD considers the case of multiple responses with incremental query ID as suspicious. In this case, the VAD makes use of the trusted servers for correct resolution. If these resolutions vary from the resolutions in the caching DNS server then the VAD alerts the administrator and triggers VMDRM.

Note that for case 1 and case 2 of attacks considered in Section 2, the detection of the attack is based on the random validation of the resolution requests and responses. However the detection of attacks in case 3 is based on the suspicious event of multiple responses with incremental query ID. Furthermore, we are using process validation to determine the compromise of based on the suspicious event of multiple responses with incremental query ID. Furthermore, we are using process validation to determine the compromise of monitored DNS server and resolution from trusted servers for determining the correct

```
root       814     1  0 21:05 tty2     00:00:00 /sbin/getty -8 38400 tty2
root       816     1  0 21:05 tty3     00:00:00 /sbin/getty -8 38400 tty3
root       824     1  0 21:05 tty6     00:00:00 /sbin/getty -8 38400 tty6
root       838     1  0 21:05 ?        00:00:00 acpid -c /etc/acpi/events -s /va
root       839     1  0 21:05 ?        00:00:00 cron
daemon     840     1  0 21:05 ?        00:00:00 atd
whoopsie   846     1  0 21:05 ?        00:00:00 whoopsie
bind       854     1  0 21:05 ?        00:00:00 /usr/sbin/named -u bind
root       914     1  0 21:05 tty1     00:00:00 /sbin/getty -8 38400 tty1
```

Figure 4: VM processes for BIND DNS server

resolution and detecting the attacks for cache poisoning and ID guessing.

```
[  814] getty (struct addr:1eab2e00)
[  816] getty (struct addr:1eb21700)
[  824] getty (struct addr:1b219700)
[  838] acpid (struct addr:1ba84500)
[  839] cron (struct addr:19ba0000)
[  840] atd (struct addr:1aa4ae00)
[  846] whoopsie (struct addr:1b21ae00)
[  854] named (struct addr:1aa4dc00)
[  914] getty (struct addr:1aa4c500)
```

Figure 5: Runtime process validation using VMI

3.3.3 VM Data Restore Mechanism (VMDRM)

The VMDRM inspects to see whether the incorrect data has already been written to file using a VMI technique and categorises what kind of threat the DNS server was under. Depending on attack, VMDRM restores the related DNS records and cache files from the last known good configuration by restoring the full virtual machine from a snapshot. If a VM receives an inaccurate response from an external DNS server, VMDRM will flush the DNS cache from the VM.

3.4 Trusted Server

Trusted server is an external entity to our model. The VAD makes use of DNSSEC if it is supported by the Authoritative servers for secure resolution. For the cases where the authoritative servers do not support DNSSEC, we assume that there is a trusted server (see Figure 3) which can be used by the VAD to determine correct resolution. The trusted server can make further queries to multiple trusted servers for determining the correct resolution. Also any of secure communication such as DNSSEC or IPsec is used for communication between trusted servers.

4 Implementation and Analysis

We have implemented our model using Xen VMM and DNS servers running as virtual machines on the VMM. Xen VMM started off as a research project at the University of Cambridge and it was first introduced by Barham et al. (2003) as a high-performance resource-managed virtual machine monitor, which enables applications such as distributed web services, server consolidation and secure computing platforms. Xen is a native (or hypervisor-based) VMM where it runs directly on the hardware as lowest and most privileged layer. In Xen terminology “domain" refer to a running virtual machine within which a guest OS executes. “domain 0" (Dom0) boots with the hypervisor and works as the control interface with special management privileges which has direct access to underlying physical hardware. All other virtual machines are called “domain U" (domU) in Xen terminology. Initially, on x86 architecture, Xen kernel code runs in Ring 0, while the hosted domains run in Ring 1 or Ring 3. Running operating system in Ring 1 or 3 instead of usual Ring 0 required operating system to be modified in order to suit the new privilege levels. This means only paravirtualized (i.e. modified operating systems) guests were able run on Xen. But Xen version 3.0 and above can use unmodified guest operating systems (e.g. Microsoft Windows XP) for hardware-assisted virtual machines (HVM) with supporting underlying hardware(e.g. Intel VT and AMD Pacifica).

4.1 Design Choices

Components RCM and VMDRM are placed in Dom0 of the hypervisor as RCM needs to be in the data path of the bridge interface using the VMMs backend driver domain and VMDRM require restoration and prevention abilities which can trigger commands available within the hypervisor. VAD component which is placed inside the Dom0 is used for process validation, DNSSEC validation with authoritative servers and establishing permanent secure tunnel using IPSec VPN with an external trusted server.

5 Conclusion

In this paper we have analysed different types of attacks on the DNS server and proposed techniques to deal with these attacks. The main focus of our work is on the passive detection of attacks. Although our architecture is able to detect different types of attacks, it will not prevent the client from being attacked. In our current work, we are developing active defence techniques to counteract these attacks and secure the clients.

6 References

Barham, B., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A. (2003), Xen and the art of virtualization, in `SIGOPS Oper. Syst. Rev.', Vol. 38, No. 5, ACM Press, New York, NY, USA, pp. 164-177.

Mockapetrisi, P. (1987), Domain Names - Concepts And Facilities, RFC-1034.

Mockapetrisi, P. (1987), Domain Names- Implementation And Specification, RFC-1035.

Rosenblum, M. & Garfinkel, T. (2005), Virtual Machine Monitors: Current Technology and Future Trends, in `Computer', Vol. 38, No. 5, IEEE Computer Society, Los Alamitos, CA, USA, pp. 39-47.