Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 96 (2016) 1627 – 1636

Procedia Computer Science

20th International Conference on Knowledge Based and Intelligent Information and Engineering Systems

A Cyber Attack-Resilient Server Using Hybrid Virtualization

Fumikazu Sano[a], Takeshi Okamoto[a,*], Idris Winarno[b], Yoshikazu Hata[b], Yoshiteru Ishida[b]

[a]*Kanagawa Institute of Technology, 1030, Shimo-ogino, Atsugi, Kanagawa 242-0292, Japan*
[b]*Toyohashi University of Technology, Tempaku, Toyohashi, Aichi 441-8580, Japan*

Abstract

This paper describes a novel, cyber attack-resilient server using hybrid virtualization that can reduce the downtime of the server and enhance the diversity of operating systems by adding a Linux virtual machine. The hybrid virtualization consists of machine- and application-level virtualization. The prototype system virtualizes a machine using VMware ESXi, while the prototype system virtualizes a server application using Docker on a Linux virtual machine. Docker increases the speed at which a server application starts while requiring fewer resources such as memory and storage. Performance tests showed that the prototype system reduced the downtime of the DNS service by exploiting a vulnerability with no false positive detections compared with our previous work.

Peer-review under responsibility of KES International

Keywords: resilient computing; biological diversity; cyber attack; hypervisor; vulnerability

1. Introduction

The growth of the Internet has led to increased demands for mission-critical servers. Although the key feature of a mission-critical server is fault tolerance, the latter does not cover cyber attacks that exploit vulnerabilities in server applications. In particular, cyber attacks on vulnerabilities that allow an attacker to execute arbitrary code can compromise the most important aspects of information security, i.e., confidentiality, integrity, and availability.

Security products for personal computers, such as Norton Security and Virus Buster, can detect and prevent known cyber attacks, but not unknown cyber attacks. Next-generation security products, such as Palo Alto Networks traps, CrowdStrike Falcon Host, FFRI yarai, and Microsoft EMET1, can detect and prevent both known and unknown cyber attacks. A well-known security method for a server is mandatory access control (MAC), which is supported by many

* Corresponding author. Tel.: +81-46-291-3264; fax: +81-46-291-3272.
E-mail address: take4@nw.kanagawa-it.ac.jp

Peer-review under responsibility of KES International
doi:10.1016/j.procs.2016.08.210

operating systems. MAC is a type of access control by which the operating system restricts the capabilities of a process or a thread that can access or perform operations on objects, such as files and network ports. MAC proactively protects the operating system and applications from external and internal threats, including from zero-day attacks. In addition, various techniques to detect unknown attacks have been developed, including ROPGuard[1], ROPecker[2], and SecondDEP[3]. However, by the time these products and techniques detect a cyber attack, the process of the server application has already lost normal execution control. Therefore, these products and techniques must terminate the process of the server application. Restarting the server application is not a viable solution, as the same cyber attacks can again cause the server application to restart.

Our previous work proposed a cyber attack-resilient server inspired by the concept of biological diversity[4], which plays important roles in species survival and adaptability, as not all species or individuals are infected with the same infectious agent. Similarly, a cyber attack-resilient server diversifies its own operating system and the implementation of server applications by using multiple virtual machines that are running different operating systems and different implementations of the same server protocol specification. This approach is based on findings that not all implementations are affected by the same vulnerability, because they depend on the implementation, except for vulnerabilities in specification and on shared libraries such as libc and OpenSSL. The prototype system of a cyber attack-resilient server suppressed the downtime of the DNS service by exploiting a vulnerability to below 4 seconds, with no false positives detected.

This paper describes a proposed novel cyber attack-resilient server using hybrid virtualization that reduces the downtime of the service and enhances the diversity of operating systems by adding a Linux virtual machine. Linux can virtualize applications at the application level using Docker, which enhances the speed at which an application starts and requires fewer resources, such as memory and storage.

2. Cyber attacks

Representative security incidents involving servers include hacking, website alteration, information leakage, and denial of service (DoS). Except for flooding attacks, which make a server or network resource unavailable, these incidents are most frequently due to vulnerabilities. These vulnerabilities can include flaws or weaknesses in the design or implementation of a system (e.g., a buffer overflow) and its operation or management (e.g., a weak password). The former can be solved by technical approaches, including anti-virus software and security updates, while the latter can be overcome by organizational approaches, such as organized training to improve the security literacy of staff members.

This paper focuses on cyber attacks that exploit flaws or weaknesses in the design or implementation of a system, such as vulnerabilities that allow arbitrary code execution. This paper, however, excludes cyber attacks on vulnerabilities (e.g., CVE-2014-0160[5]) that allow information leakage because of the difficulties in detecting the theft of information until the vulnerability is revealed.

3. Cyber attack-resilient server using hybrid virtualization

A resilient server can be divided into three configurations (Fig. 1). The first configuration involves use of heterogeneous virtual machines running different operating systems and different implementations of the same server protocol specification. This configuration provides resilience against cyber attacks, as not all implementations will be affected by the same vulnerability, except for vulnerabilities in specification and on shared libraries. For example, the number of server implementations for a DNS service is 20. If four major operating systems (i.e., BSD, Linux, Solaris, and Windows series) are supported by all implementations of the DNS service, the maximum combinations of an operating system and a server implementation is 80, a number thought to make a server significantly more resilient. Attack-resilient servers must not include shared libraries common to all server applications, because using a common shared library may introduce a security hole in the attack-resilient server if the shared library possesses a vulnerability shared library. For example, the “HeartBleed” vulnerability in the “OpenSSL” shared library (i.e., CVE-2014-0160) affected many implementations on a Web service. The second configuration involves a fault-tolerant system consisting

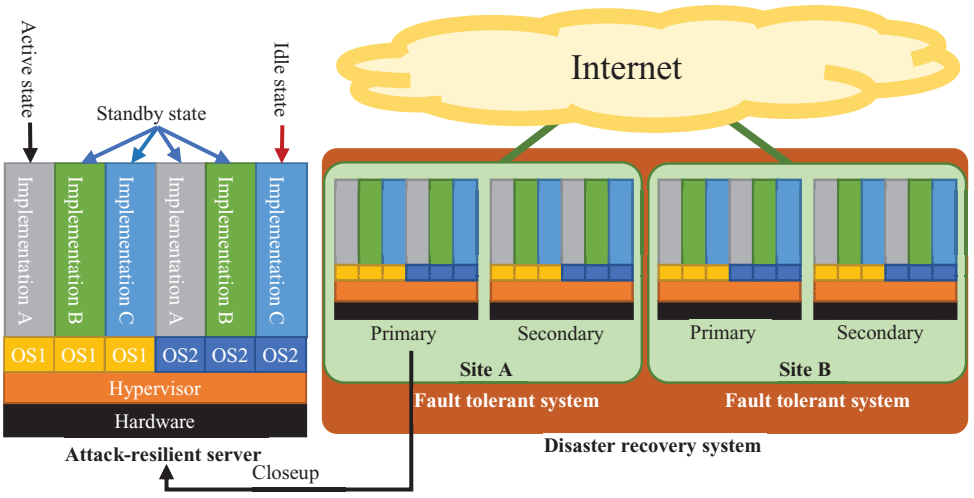

Fig. 1. Overview of a resilient server.

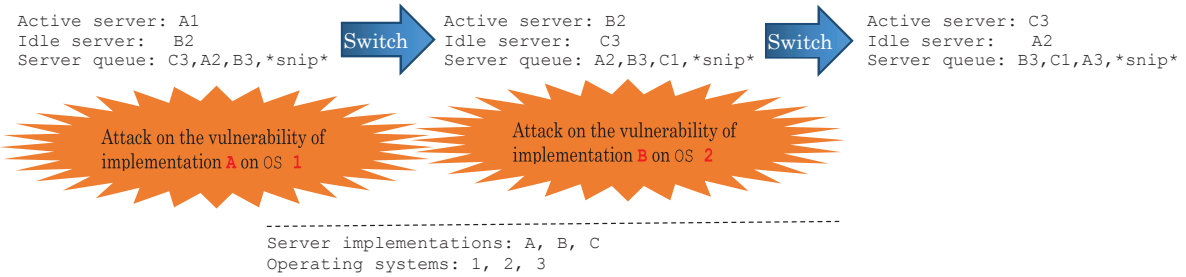

Fig. 2. Switching flow of virtual machines.

of two or more redundant attack-resilient servers. This configuration provides resilience against failures on physical machines. The third configuration involves a disaster recovery system consisting of two or more redundant fault-tolerant systems. This configuration provides resilience against disasters, such as earthquakes and floods.

This paper focuses on attack-resilient servers using hybrid virtualization, which consists of machine-level virtualization using a hypervisor and application-level virtualization using a container, reducing server downtime. Each attack-resilient server consists of a hypervisor and multiple virtual machines.

Each virtual machine can be in any of three states: active, idle, or suspended. An active and an idle machine runs on the server, but the idle machine does not provide its own service, thus avoiding competition with the service of the active machines. Suspended machines stand by in a queue. Each virtual machine uses a security module for intrusion protection. If the intrusion protection system detects a cyber attack on its own service, it suspends the service.

The hypervisor manages virtual machines using a service monitor and a virtual machine changer. The service monitor periodically checks the service status of the active machine. When the service monitor detects that the service has stopped, the monitor notifies the virtual machine changer, which suspends the active machine and labels that machine as compromised. At the same time, the idle machine is promoted to be the active machine and starts to provide the service. This is followed by promoting a suspended machine from the queue of suspended machines to an idle machine. Finally, the virtual machine changer notifies administrators and security analysts that it has detected a cyber attack. In this way, the attack-resilient server maintains its own service even if a vulnerability on its server application is attacked. Figure 2 shows the switching flow when the active machine detects a cyber attack.

When administrators and security analysts receive notification of cyber attacks, they can resume use of the compromised machine and analyze the process memory of the suspended server application. If they identify the vulnerability of the server application and find a security update for that vulnerability, they can apply the security

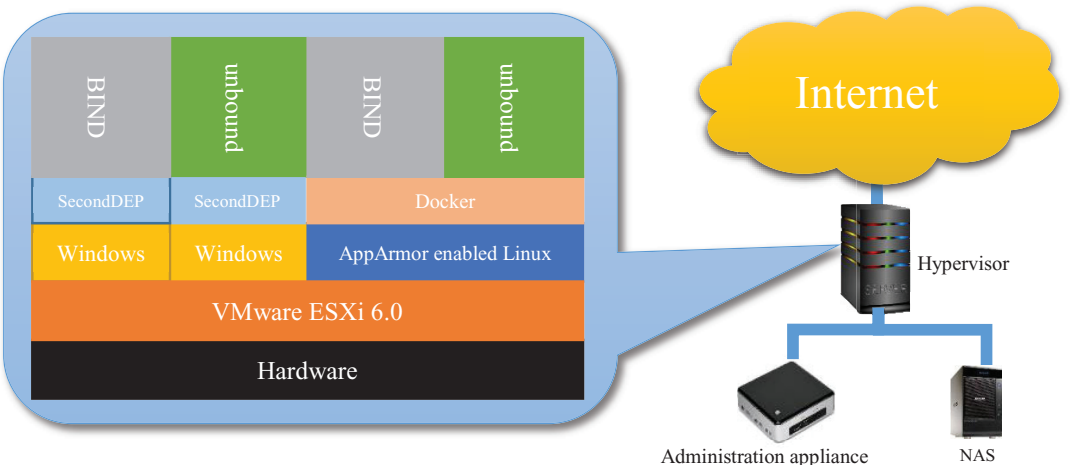

Fig. 3. Overview of a prototype system

update and add the machine to the queue. If all virtual machines are compromised, the attack-resilient server stops the service until administrators or security analysts apply the security update to the compromised machines.

The attack-resilient server will be able to maintain its own service if it consists of at least four virtual machines composed of a combination of two different operating systems and two different implementations of the same server protocol specification. Not all implementations are affected by the same vulnerability, except for vulnerabilities in specifications and on shared libraries, and two different vulnerabilities have not been found in two different implementations at the same time.

4. Prototype system

A prototype system for a DNS service was built to evaluate the continuity of the service (Fig. 3). A DNS service is an infrastructure for the Internet, such that enhancement of the resilience of the DNS service is expected to contribute to the resilience of the Internet. The prototype system consists of an administration appliance, a hypervisor, and a network attached storage (NAS). The administration appliance consists of a VMware vCenter Server Appliance 6.0.0 on another VMware ESXi 5.5. The NAS is used to share images of virtual machines between two hypervisors for fault tolerance in future work. The specifications of a physical machine for the hypervisor and the NAS are:

- Hypervisor
  - CPU: AMD Opteron 6234 2.4 GH × 2 CPUs
  - Motherboard: Supermicro H8DGi
  - Memory: 16 GB
  - Disk: 3 TB
  - Network: GbE × 2 and 10GbE × 1
- NAS
  - Vendor: NETGEAR
  - Model: ReadyNAS 102
  - CPU: Marvell Armada 370 1.2GHz
  - Memory: 512 MB
  - Disk: 3 TB (RAID 1)
  - Network: GbE × 1

4.1. Hypervisor and virtual machines

The hypervisor is VMware ESXi 6.0, which is included in VMware vSphere 6 Enterprise Plus. The hypervisor has three virtual machines, two running Windows and the third running Linux. A single virtual machine running Linux is sufficient because Linux can virtualize two server applications at the application level using Docker 1.10.3. The application level virtualization has the advantage of much faster switchover between applications than the switchover between virtual machines.

4.1.1. Windows on virtual machines

One virtual machine runs ISC BIND 9 as a DNS server application and another runs unbound. All server applications are equipped with SecondDEP, which detects and prevents the execution of a small piece of executable code, called "shellcode," in cyber attacks, based on evidence showing that shellcode calls Windows APIs from a data area. SecondDEP can detect a zero-day attack if that attack uses the shellcode.

When SecondDEP detects a cyber attack, it suspends the server process, enable the latter to be analyzed by administrators or security analysts. Simultaneously, the service monitor detects stoppage of the service, because SecondDEP suspends the server process.

In addition, the registry value "`ArpRetryCount`," which controls the number of times that Windows broadcasts a gratuitous ARP request message for duplicate IP address detection while initializing[6], is set to zero, reducing the time required to connect a virtual network adapter to an operating system. Because `ArpRetryCount` has a default value of 3, it takes at least 3 seconds to connect the virtual network adapter to Windows.

The configurations of the virtual machines for Windows are:

- OS: Windows Server 2012
- CPU: 1 CPUs × 2 Cores
- Memory/Disk: 4 GB/64 GB
- SCSI controller: VMware paravirtual
- Disk provisioning: Thick (Eager Zeroed)
- Network adapter for the Internet: VMXNET3 enabled RSS
- Network adapter for the service monitor: VMXNET3 enabled RSS
- Security module: SecondDEP
- Power management: High performance
- `ArpRetryCount` registry value: 0
- DNS implementation: BIND 9 or unbound

4.1.2. Linux on a virtual machine

Linux on a virtual machine runs both ISC BIND 9 and unbound on Docker, which provides virtualization at the application level. Docker makes it faster to switch between containers and requires fewer resources, such as memory and storage. A disadvantage of Docker is the deterioration in the strength of security (see Section 6) because Docker is still in development. All server applications are protected by AppArmor, a security module that restricts the capabilities of programs such as network access, raw socket access, and the permission to read, write, or execute files on matching paths. AppArmor is easier to set up than SELinux, although fewer operations are restricted by AppArmor than by SELinux (see Section 6).

When AppArmor denies a certain access, an `inotify cron` daemon launches a shell script that sends the log file of AppArmor to administrators or security analysts. At this time, the service monitor detects the stoppage of service, because the server application loses its own normal execution control.

The configuration of the virtual machine for Linux is:

- OS: Ubuntu 15.10
- Kernel version: 3.19.0
- CPU: 1 CPUs × 2 Cores
- Memory/Disk: 4 GB/40 GB
- SCSI controller: VMware paravirtual
- Disk provisioning: Thick (Eager Zeroed)
- Network adapter for the Internet: VMXNET3
- Network adaptor for the service monitor: VMXNET3
- Security module: AppArmor
- DNS implementation: BIND 9 or unbound

4.2. Administration appliance

An administration appliance runs VMware vCenter Server 6 standard, which provides a centralized platform for managing VMware vSphere environments, including VMware ESXi. The administration appliance runs a service monitor and a virtual machine changer, which are implemented as a bash script.

The service monitor checks whether the active machine is providing the DNS service by executing the command `nanonslookup`, which we implemented to support an arbitrary timeout with a nanosecond resolution, specifying a timeout of 0.5 seconds every 0.5 seconds. If the command times out twice consecutively, the service monitor determines that the active machine is stopping the service and notifies the virtual machine changer. The monitor subsequently checks whether the number of threads every second is equal to the maximum number of threads. If the number of threads is equal to the maximum number of threads, the service monitor determines that the active machine has difficulty receiving requests due to a DoS attack and notifies the virtual machine changer.

After receiving notification from the service monitor, the virtual machine changer retrieves the operating systems of the current and next virtual machines. If the current virtual machine is running Windows, or if the current virtual machine is running Linux and the next virtual machine is running Windows, the virtual machine changer switches the current virtual machine to the next virtual machine. Under other conditions, the virtual machine changer switches the current Docker container to the next Docker container. The switchover between virtual machines or between Docker containers is performed by the `vmrun` command-line utility, which is included in VMware VIX API libraries, to control virtual machines and to automate operations on virtual machines. However, it takes approximately 2.6 seconds for the `RunProgramInGuest` command of the `vmrun` utility to run a program on a virtual machine, and it takes 1 second for the service monitor to detect the stoppage of service. Therefore, it takes at least 3.6 seconds for the virtual machine changer to switch between virtual machines. Unfortunately, this time was slightly longer than in our previous work, since the latter did not have to use the `vmrun` utility. To reduce the time to connect to ESXi, the `RunProgramInGuest` command was divided into `vmrun-server` and `vmrun-client` programs, which were implemented using VIX APIs. The `vmrun-server` connects to ESXi (required time: 0.7 sec.) and waits for requests from the `vmrun-client`. When the `vmrun-server` receives four parameters (i.e., a VMX file name of a virtual machine, credentials for the virtual machine, a program path and its arguments) from the `vmrun-client`, the `vmrun-server` opens the virtual machine (0.6 sec.), logs in with credentials (0.1 sec.), and executes the program (1.1 sec.). Consequently, the switchover time can be reduced approximately 0.7 seconds to connect to ESXi.

4.2.1. Switchover between virtual machines

When switching between virtual machines, the virtual machine changer changes the IP address of the active machine to that for an idle machine with administrator privilege through the `vmrun-client` command by executing the following command on the active machine:

```
> netsh.exe interface ip set address "Ethernet0 2" static IDLE-MACHINE-ADDRESS
   SUBNETMASK GATEWAY-IP-ADDRESS
```

At the same time, the virtual machine changer changes the IP address of the idle machine to the one for an active machine and starts service on the idle machine. For example, the Windows virtual machine for BIND executes the following commands with administrator privilege though the `vmrun-client` command with a VBS script.

```
> netsh.exe interface ip set address "Ethernet0 2" static ACTIVE-MACHINE-ADDRESS
   SUBNETMASK GATEWAY-IP-ADDRESS
> C:\Program Files (x86)\ISC BIND 9\bin\named.exe -f
```

At this time, the idle machine is promoted to be the active machine. The suspended machine is resumed, and it is promoted to be the idle machine. The compromised machine is suspended, and it is labeled as compromised.

4.2.2. Starting and switchover between Docker containers

At startup, Linux in a virtual machine runs all Docker containers bound to different port numbers on the host to more quickly and surely switch between containers. One container has a port number of 53 and the other has a port number of 60050. For example, the Docker container for ISC BIND 9 executes the following commands with *non-root privilege* though the `vmrun` utility with a bash script.

```
$ docker run --name bind --security-opt apparmor:usr.sbin.bind --publish \
   ACTIVE-MACHINE-ADDRESS:53:53/udp --publish ACTIVE-MACHINE-ADDRESS:53:53/tcp \
   -td Ubuntu-bind-image
$ docker exec bind start-stop-daemon --start --oknodo --quiet --exec \
   /usr/sbin/named --pidfile /var/run/named/named.pid -- -u bind
```

The first command runs the container in the background through the `docker` command with four options. The "`name`" option assigns a name to the container, the "`security-opt`" option specifies a profile of AppArmor for the daemon of the service, the "`publish`" option publishes the container's port to the host, and the "`td`" option runs the container in the background with a pseudo terminal for `/bin/bash`. The second command starts the daemon on the container through the `start-stop-daemon` command with six options. The "`start`" option starts the daemon specified by the "`exec`" option, the "`oknodo`" option returns to exit status 0, the "`quiet`" option does not display any messages except for error messages, the "`pidfile`" option checks whether a daemon process has created the file specified by its option, the "`--`" option is a terminator of the `start-stop-daemon` command, and the "`u`"

option is an option of the ISC BIND 9 daemon, which sets the UID of the daemon process to the "bind" user to drop all root privileges except for the ability to bind to the 53 port and to set process resource limits.

When switching between Docker containers, the virtual machine changer stops and removes the active container, and modifies the network address translation table on the host to change the host's port number, which is bound to the idle container's port 53, to port 53. At this time, the compromised container is labeled as compromised. For example, the Docker container for unbound is promoted to the active Docker container by the following commands with root privilege though vmrun-client command with a bash script:

```
# docker stop bind && docker rm bind
# iptables -t nat -R DOCKER 2 -d ACTIVE-MACHINE-ADDRESS/32 ! -i docker0 -p udp \
   -m udp --dport 53 -j DNAT --to-destination 172.17.0.2:53
# iptables -t nat -R DOCKER 3 -d ACTIVE-MACHINE-ADDRESS/32 ! -i docker0 -p tcp \
   -m tcp --dport 53 -j DNAT --to-destination 172.17.0.2:53
```

The first command stops and removes the Docker container for BIND. The second and third commands change the port number on the host to port 53.

Finally, if all machines are compromised, the virtual machine changer stands by until administrators or security analysts apply a security update to the compromised machine.

4.3. Administrators or security analysts

Administrators or security analysts copy the image of the compromised machine to the physical machine, on which VMware Workstation has been installed, to analyze the server application process. The compromised machine is resumed and the server application is checked to determine whether it has been attacked. If the server application has been suspended or abnormally terminated, administrators or security analysts determine whether the server has been attacked, and then analyze the server application process to identify a vulnerability that has been exploited. If they identify the vulnerability and find an appropriate security update, they apply the security update to the server application and add the virtual machine to the queue of the suspended machines on VMware ESXi.

5. Performance tests

The continuity of the service provided by the prototype system was tested to evaluate the resilience of the attack-resilient server. First, we evaluated the downtime of the DNS service due to a cyber attack via a vulnerability in a server application. We implemented a vulnerable DNS server application for Linux and Windows. The server applications have a stack buffer overflow vulnerability triggered when parsing the QNAME field in a DNS query that allows an attacker to execute arbitrary code. This test assumed that a zero-day attack exploited an unknown vulnerability, because not every vulnerability is known in all the latest DNS server applications.

The downtime of the server application on Windows consists of the time spent changing the IP address of the idle machine and starting the service of the idle machine (or the time spent changing the IP address of the active machine), and the timeout period of the nanonslookup command. The average downtime of 10 trials was 3.07220 seconds, consisting of an average time to change the IP address and to start the service of 1.74896 seconds and an average timeout period of 1.0 seconds. The maximum downtime was 3.37173 seconds, consisting of a maximum time to change the IP address and to start the service of 1.79769 seconds and a maximum timeout period of 1.0 seconds.

The downtime of the server application on Linux consists of the time spent stopping and removing the active container and changing the host port bound to the idle container to port 53, as well as the timeout period of nanonslookup command. The average downtime of 10 trials was 2.49986 seconds, and the maximum downtime was 2.84766 seconds.

These maximum downtimes were reduced less than in our previous work, and these results are considered acceptable for users of critical mission servers, because these downtimes were shorter than the timeout period of the nslookup command.

Finally, we ran the prototype system without attacks for 14 days to evaluate the rate of false positives. The results indicated that the prototype system could maintain the service without false positives.

6. Discussion

6.1. Resilience against DoS attacks

DoS attacks can be divided into three types. The first type exploits a vulnerability that causes the stoppage of service. This type of vulnerability has been found in several versions of BIND 9 (e.g., CVE-2012-1667[7] and CVE-2015-5722[8]). The second type of attack exhausts resources of the server, such as SYN flood attack and HTTP POST DoS attack. The third type of attack markedly reduces network bandwidth by sending massive numbers of requests or replies, such as a UDP-based amplification DoS attack using a DNS or a NTP service.

Although the prototype system was able to detect and prevent the first and second types of attack, it had difficulty preventing the third type, because the prototype system was unable to stop attackers from sending massive numbers of requests. Effective countermeasures against the third type of attack may include increases in service sites and the use of services for DoS attack countermeasures provided by Internet service providers or content delivery networks. Docker is a suitable solution for increasing service sites, because its container can be easily deployed on a wide variety of platforms, including Amazon Web Services (AWS), Microsoft Azure, and Google Compute Engine. In addition, the container on some platforms, such as AWS and Microsoft Azure, can use their mitigation method[9, 10] against DoS attacks.

6.2. Fault tolerance of DNS services

Fault tolerance for DNS servers is provided mostly by replicating servers: a master server and slave servers. The DNS must maintain consistency of zone information between master and slave servers using zone transfer. Fortunately, an attack-resilient server will not require zone transfer using VMware vSphere FT, which guarantees consistency between two hypervisors, including the consistency of zone information and memory images.

6.3. Docker Security

Although Linux was found to have security weaknesses in container technologies, [11, 12, 13, 14] most of these have been solved[15]. Docker isolates a process from other running processes using Linux kernel namespace, which prevents the process from seeing other running processes, and minimizes privileges required for containers using Linux kernel capabilities that split root privileges into multiple pieces[16]. In addition, Docker 1.10 supports Linux user namespace, which maps the root user of a container to a non-root user of the Docker host system, thus mitigating the effects of container-to-host privilege escalation[12]. However, Docker still has two weaknesses:

- The Docker daemon requires a root privilege.
- The Docker containers share the same host kernel.

These weaknesses can cause all running containers and the Docker host system to be compromised if a container with a privilege escalation vulnerability is hijacked. These weaknesses can be mitigated using mandatory access control (MAC) and secure computing mode[17].

SELinux and AppArmor are implementations of MAC for the Linux kernel. As described in Section 4.1.2, AppArmor is easier to set up than SELinux, although fewer operations can be restricted by AppArmor than by SELinux. For example, AppArmor cannot prevent the execution of Meterpreter shellcode, which is provided by Metasploit Framework, because AppArmor primarily uses path-based access control and the Meterpreter shellcode is executed without file access. In contrast, SELinux can prevent the execution of the Meterpreter shellcode with the permission of `execmem`. Therefore, SELinux enhances Docker security more than AppArmor.

Secure computing mode restricts access to specific system calls by a process, which has the potential to prevent malicious activities and attempts to exploit vulnerabilities on the Docker daemon (e.g., CVE-2014-6407[18], CVE-2014-6408[19]), the server application (i.e., the Docker container), and the Linux kernel (e.g., CVE-2008-0009[20]). Creating a set of allowed system calls for an application may be troublesome, because a large set of system calls is available, and it may be difficult to determine which system calls will be used in the application[14].

6.4. Operating costs

The more diverse the combination of an operating system and a server implementation, the more secure will be the attack-resilient server. Unfortunately, however, this will increase the operating costs for the attack-resilient server, including the costs of setup of all configurations for all server applications and all operating systems, and their maintenance, including the need for security updates. However, even an attack-resilient server with low diversity (i.e., low operating cost) can enhance security, unless the attack-resilient server has a shared library common to all server implementations. This was based on the observation, that not all implementations are affected by the same vulnerability, except for vulnerabilities in specifications and on shared libraries. For example, the operating costs of the prototype system can be reduced by decreasing the number combinations by four, such that the prototype system has no shared library common to all server applications.

6.5. Vulnerability on a hypervisor

Vulnerabilities have been found on various hypervisors (e.g., CVE-2016-1571[21] and CVE-2014-8370[22]), with some of these vulnerabilities causing stoppage of the hypervisor. The prototype system described here depends on a single hypervisor, although our other paper[23] proposed a new resilient server consisting of two physical machines (i.e., Xen-based and LXC-base virtualization machines) to mitigate the threat of attacks on the vulnerabilities of Xen and LXC software. Therefore, the diversity of hypervisors on the attack-resilient server should be enhanced in future, not only using VMware ESXi, but using various hypervisors such as Xen, KVM, and Hyper-V.

7. Related work

7.1. Enhancement of security based on the concept of diversity

The concept of diversity in operating systems was first introduced in the Tempo-C specializer of the Synthetix project[24]. This specializer focuses on the "micro diversity" of kernel modules in an operating system by modifying these modules, whereas the attack-resilient server focuses on the "macro diversity" of operating systems and server implementations. Therefore, our attack-resilient server is more diverse than the Tempo-C specializer, because the Tempo-C specializer can diversify individuals (i.e., kernel modules), but cannot diversify species (i.e., operating systems).

Another related work is address space layout randomization (ASLR), which was designed and implemented by the Linux PaX project[25]. ASLR randomizes the address space positions of key data areas of a process, including the positions of the stack, heap and modules. These features of ASLR belong to the category of "micro diversity," because ASLR focuses only on the address space positions of key data areas without modifying the executable code of a server application. These techniques can be incorporated into the attack-resilient server.

7.2. Enhancement of server attack-resilience

The approaches we have described use virtualization to make servers more resilient to faults and cyber attacks[26, 27]. One of these approaches applies a self-repair network model to the resilient server to recover from fault states[27]. Another approach makes a server application more resilient by attaching a security module to the server application[28]. This approach has the advantages of not requiring multiple operating systems and server implementations, while having the disadvantages of false positive detection and of leaving a vulnerability open. Therefore, this approach is considered effective only when all virtual machines on the attack-resilient server are run out.

8. Conclusions

We have proposed a cyber attack-resilient server using hybrid virtualization to reduce server downtime and to enhance the diversity of operating systems by adding a Linux virtual machine. Docker on Linux makes starting a server application faster while requiring fewer resources such as memory and storage. We built and tested a prototype

system to evaluate the continuity of the service. The prototype system was found to reduce the downtime of the DNS service by exploiting a vulnerability with no false positives, compared with our previous work.

We are currently constructing an attack-resilient server with fault tolerance using VMware vSphere FT. In future work, we will enhance the diversity of hypervisors on the attack-resilient server, using not only VMware ESXi but various hypervisors such as Xen, KVM, and Hyper-V. In addition, we will enhance resilience against disasters using VMware vCenter Site Recovery Manager Enterprise.

References

1. Fratric I, Runtime prevention of return-oriented programming attacks. University of Zagreb. 2012. https://code.google.com/ p/ropguard/
2. Cheng Y, Zhou Z, Miao Y, Ding X, Deng H, ROPecker: A generic and practical approach for defending against ROP attack; 2014
3. Okamoto T, SecondDEP: Resilient computing that prevents shellcode execution in cyber-attacks. *Procedia Computer Science* 60:691-669; 2015
4. Sano F, Okamoto T, Idris W, Hata Y, Ishida Y, A cyber attack-resilient server inspired by diversity, *In: Proc. of the 21st International Symposium on Artificial Life and Robotics*; 2016, p. 31-35.5. CVE-2014-0160, http://www.cvedetails.com/cve/2014-0160/
6. Microsoft Corporation, TCP/IP Registry Values for Microsoft Windows Vista and Windows Server 2008, Microsoft Windows Server 2008 White Paper. 2011.
7. CVE-2012-1667, http://www.cvedetails.com/cve/2012-1667/
8. CVE-2015-5722, http://www.cvedetails.com/cve/2015-5722/9. Amazon Web Services, AWS Best Practices for DDoS Resiliency. 2015. https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf
10. Microsoft Corporation, Microsoft Azure Network Security, 2015. http://download.microsoft.com/download/C/A/3/CA3FC5C0-ECE0-4F87-BF4B-D74064A00846/AzureNetworkSecurity_v3_Feb2015.pdf
11. Berrangé D P, Getting started with LXC using libvirt, 2011, https://www.berrange.com/posts/2011/09/27/getting-started-with-lxc-using-libvirt/
12. LXC, gentoo linux wiki, https://wiki.gentoo.org/wiki/LXC
13. Jay T, Container Security: Isolation Heaven or Dependency Hell, Security Blog, Red Hat, 2014, https://securityblog.redhat.com/2014/12/17/container-security-isolation-heaven-or-dependency-hell/
14. Petazzoni J, Containers & Docker: How Secure Are They?, Docker Blog, Docker, 2013, https://blog.docker.com/2013/08/containers-docker-how-secure-are-they/
15. Frazelle J, Docker Engine 1.10 Security Improvements, 2016, https://blog.docker.com/2016/02/docker-engine-1-10-security/
16. Docker security, Docker, https://docs.docker.com/v1.9/engine/articles/security/
17. Hayden M, Securing Linux Containers, InfoSec Reading Room, SANS Institute, 2015, https://www.sans.org/reading-room/whitepapers/linux/securing-linux-containers-36142
18. CVE-2014-6407, http://www.cvedetails.com/cve/2014-6407/
19. CVE-2014-6408, http://www.cvedetails.com/cve/2014-6408/
20. CVE-2008-0009, http://www.cvedetails.com/cve/2008-0009/
21. CVE-2016-1571, http://www.cvedetails.com/cve/2016-1571/
22. CVE-2015-1044, http://www.cvedetails.com/cve/2014-8370/
23. Winarno I, Okamoto T, Hata Y, Ishida Y, Increasing the diversity of resilient server using multiple virtualization engine technology, *Procedia Computer Science*, 2016. (To be appeared)
24. Pu C, A specialization toolkit to increase the diversity of operating systems. PhD Thesis. 1996. Portland State University.
25. Spengler B, PaX: The Guaranteed End of Arbitrary Code Execution. 2003. https://grsecurity.net/PaX-presentation.ppt
26. Idris W, Ishida Y, Simulating Resilient Server using XEN Virtualization, *Procedia Computer Science* 60: 1745–1752; 20
27. Winarno I, Okamoto T, Hata Y, Ishida Y, A resilient server based on virtualization with a self-repair network model. *International Journal of Innovative Computing, Information and Control*. 2016. (To be apeared)
28. Tarao M, Okamoto T, Toward an artificial Immune server against cyber attacks. *In: Proc. of the 21st International Symposium on Artificial Life and Robotics*; 2016, p. 36-39.