International Journal of Innovative Computing, Information and Control
Volume **13**, Number **5**, October **2017**

ICIC International ©2017 ISSN 1349-4198
pp. 1735–1749

IMPLEMENTATION AND EVALUATION OF AN INTRUSION-RESILIENT SYSTEM FOR A DNS SERVICE

Takeshi Okamoto[1], Fumikazu Sano[1], Idris Winarno[2]
Yoshikazu Hata[2] and Yoshiteru Ishida[2]

[1]Department of Information Network and Communication
Kanagawa Institute of Technology
1030 Shimo-ogino, Atsugi, Kanagawa 243-0292, Japan
take4@nw.kanagawa-it.ac.jp

[2]Department of Computer Science and Engineering
Toyohashi University of Technology
Tempaku, Toyohashi 441-8580, Japan
{ idris; hata }@sys.cs.tut.ac.jp; ishida@cs.tut.ac.jp

Received April 2017; revised July 2017

Abstract. *The Internet has increased its inclusion of highly available servers. These servers require not only fault tolerance but intrusion resilience. DNS servers are particularly critical components on the Internet, since a DNS service is an infrastructure for the Internet. This paper proposes an implementation of an intrusion-resilient system for the DNS service using an active redundancy approach with diversification. This system is composed of heterogeneous virtual machines running different operating systems and different server implementations of the same server protocol specification. The approach is founded on the observation that not all implementations will be affected by the same vulnerability, apart from vulnerabilities in the specifications and shared libraries. The intrusion-resilient system needs to run only two application servers in parallel whereas others, such as intrusion-tolerant systems, require many physical resources to diversify an application server and operating systems. Tests of the intrusion-resilient system demonstrated that stab resolvers did not fail name resolution even during downtime due to cyberattacks. In addition, the overhead of the intrusion-resilient system had negligible effect on DNS service performance.*

Keywords: Intrusion-tolerant system, Fault-tolerant system, Diversity, Redundancy, Cyberattack

1. **Introduction.** The Internet has increased its use of highly available servers whose key feature is fault tolerance. The most important characteristic of fault tolerance is to avoid single point failures in highly available servers, i.e., highly available servers must continue providing services even if they experience a failure. Single point failures can be addressed by passive and/or active redundancy approaches [1] using multiple identical instances of one server. In the passive approach, all of instances need to provide their service simultaneously. This approach requires a mediator that forwards a request to all of instances in parallel, following which the mediator selects a correct response based on quorum, e.g., majority voting, and forwards it to a client. The active approach delivers a request to only one instance using a load-balancer appliance or round-robin DNS technique, following which the instance sends a response to a client. In the active approach, not all of instances need to provide their service simultaneously, i.e., only a primary server provides its service. The active approach switches from a primary server to a backup server when the former fails. This approach requires no mediator, but failure may involve downtime

during failover from the primary to backup server, while the passive approach can continue providing a service without downtime.

However, these approaches do not cover threats by cyberattacks exploiting vulnerabilities in server applications or operating systems. If attackers find a vulnerability, all of instances can be compromised since both approaches use one server application and one operating system. To mitigate threats, a diversification approach can be combined with these approaches. The diversification approach uses multiple different implementations of the same specification, e.g., Microsoft DNS on Windows, unbound on Linux, and ISC BIND 9 on NetBSD. The diversification approach is based on the observation that not all implementations are affected by the same vulnerability, because they depend on the implementation, except for vulnerabilities in specifications and shared libraries such as libc and OpenSSL. This observation is supported by reports of vulnerabilities of database management systems [2] and operating systems [3].

A passive redundancy approach with diversification has been applied to intrusion-tolerant systems (ITSs) that aim to tolerate cyberattacks on a server. The ITSs can be grouped into N-version programming- and proactive recovery-inspired systems. The former comprises redundant proxies and diversified application servers [4, 5, 6] with the advantage of being able to continue to provide their service without downtime. The disadvantage of the former is that the proxies risk a potential single point of failure (SPOF), because modules running on the proxies are not diversified. The latter comprises a control server and diversified application servers [7]. The control server periodically rejuvenates the application servers by reverting them to their initial states, e.g., by restarting them, regardless of their integrity. These systems are based on the principle that high frequency rejuvenation hinders attackers from simultaneously compromising sufficient application servers to break down the whole system. The advantage is that they pose no SPOF risk because the control server is not exposed to the Internet. However, passive redundancy approaches require a large number of physical resources to run many application servers in parallel, including virtual application servers on a hypervisor. In addition, the hypervisor may have a potential risk of SPOF, since vulnerabilities that allow attackers to cause denial of service (DoS) have been found on various hypervisors, e.g., CVE-2014-8370 and CVE-2016-1571.

Many ITSs [5, 6, 7, 8] have been applied to Web service, but they have two limitations. One is vulnerability to zero-day attacks on Web applications, such as from cross-site scripting and SQL injection attacks, because of lack of diversity of Web applications. The other is lack of mechanism to share volatile information, e.g., cookies, among different Web servers. Although Terracotta, a network-attached memory, has been used to share volatile information with multiple Jakarta tomcat Web servers [9], Terracotta is Java-based, significantly limiting the diversity of server applications to which it is applicable.

An active redundancy approach with diversification has not been applied to ITSs, although it requires less resource than the passive redundancy approach with diversification, i.e., it needs to run only one primary and backup server in parallel. Therefore, we have applied the active redundancy approach with diversification to an ITS [8, 10], which we refer to as an intrusion-resilient system (IRS). The active redundancy approach with diversification is more "resilient" than "tolerant" of cyberattacks since it may involve downtime to recover its service. Although there may be downtime during failover, Internet services are not mission-critical because the Internet protocol is based on best-effort delivery. For example, Linux allows a timeout of 5 sec for name resolution hence the active redundancy approach with diversification should reduce the downtime to within 5 sec. In this paper, we propose implementation of an IRS for a DNS service that completes the failover process within 5 sec. Since it is part of the infrastructure, resilience of a DNS service should enhance resilience of the Internet. In performance evaluations, we show that the overhead of the IRS has negligible effect on the performance of its service.

The security of the IRS is enhanced by the greater diversity of combinations of operating systems and server applications. Unfortunately, this will increase their operating costs, including those involved in the setup and maintenance of all operating systems and server applications. However, even an IRS with low diversity can enhance security, unless it includes shared libraries common to all server applications and operating systems. Thus, the operating costs can be reduced by decreasing the number of combinations.

This paper focuses on a practical implementation of the IRS for a DNS service rather than the theoretical concept of the IRS, i.e., what software may be used for the IRS, and how the IRS is implemented to reduce downtime during failover. The rest of this paper is organized as follows. Section 2 summarizes related work on the IRS. Section 3 explains some threats addressed by the IRS. Section 4 introduces the system architecture. Section 5 describes an implementation of the system. Section 6 provides performance evaluations of the IRS. Section 7 is the conclusion.

2. Related Work.

The concept of diversity for fault tolerance was introduced in 1970s [11]. Diversity in operating systems was first introduced in the Tempo-C specializer of the Synthetix project [12]. This specializer focuses on the diversity of kernel modules in an operating system by modifying these modules, whereas our IRS focuses on the diversity of operating systems and server implementations. The IRS is more resilient than the Tempo-C specializer, because the IRS can use multiple operating systems with different specifications.

N-version programming was proposed to generate functionally equivalent programs [13], but it is too costly for practical use. With the growth of the Internet, many different server implementations of the same specification have been available for Internet services frequently used on the Internet. These implementations have been used in ITSs for diversity without implementing N-version software from scratch.

Wang et al. proposed a scalable intrusion-tolerant architecture (SITAR) based on passive redundancy with diversification [4]. SITAR has five critical components: proxy servers, acceptance monitors, ballot monitors, adaptive configuration module, and audit control. These components risk SPOF due to lack of diversity, i.e., they are specific to SITAR, as all the proxy servers are exposed to the Internet while some components handle responses from application servers.

Deswarte et al. proposed a generic intrusion-tolerant architecture (GITAR) for Web servers that is similar to SITAR, except that their architecture has the diversity of intrusion detection systems [6, 14]. They enhanced the intrusion tolerance for their architecture by diversifying the operating systems and CPU architectures for proxy servers [6]. However, their architecture still has a risk of SPOF due to no diversity in modules for proxy servers similar to SITAR. The overheads of SITAR and GITAR are larger than that of our IRS, because they must wait for all responses from application servers to choose the correct response to return to a client based on quorum. For example, their duplex configuration was 2.3 times slower to receive a 1-Mbit file than their direct access configuration, while our quadplex configuration was 1.3 times slower than our direct access configuration. In addition, these architectures require a large number of physical resources to run many application servers in parallel, while our IRS only needs to run one primary and one backup server.

The passive redundancy approach with diversification has been applied to intrusion detection systems [15, 16]. These systems have a proxy server similar to that of SITAR, which compares all responses from diversified application servers. If there is a mismatch between any two responses, the proxy server raises an alarm. The intrusion detection system has the potential of detecting known cyberattacks and those unknown without patterns (a.k.a. signatures), but intrusion detection systems do not focus on the availability of their service.

In addition, there are two approaches related to our IRS, proactive recovery and self-repair network. Sood et al., proposed a proactive recovery-based ITS comprising a control server and diversified application servers [9, 17]. The control server mandatorily executes periodic rejuvenations, reverting the application servers to their initial state. Their ITS has no potential SPOF because there is no proxy server. However, their ITS is at risk of DoS or information leakage due to cyberattacks, because their ITS does not include intrusion detection. Sousa et al., proposed a highly available ITS with proactive-reactive recovery that executes rejuvenations not only periodically but also on detection of cyberattacks, and their experiments showed their proactive-reactive recovery was effective in the presence of powerful DoS attacks [18]. However, these proactive recovery approaches require a large number of physical resources to run many application servers in parallel.

Winarno et al., proposed a resilient server with a self-repair network (SRN) model that consists of an SRN manger and diversified application servers on a hypervisor [8, 19]. The SRN manger repairs abnormal virtual machines by adaptively selecting a repair model from four models: self-repair, mutual-repair, mixed-repair, and switching-repair [20]. The selection of the repair model is based on the immunity-based diagnosis model [21], which identifies faulty servers by mutually testing servers. However, their server will be unable to recover a service within the timeout time of the service when the self-repair or the switching-repair is performed due to failures, because these repairs require more than 17.5 sec.

3. Threats Addressed by an Intrusion-Resilient System.

ISO/IEC TR 13335-1 [22] states that threats may be of environmental or human origin. Environmental threats are generally accidental and include earthquakes, floods, and fires, whereas those of human origin may be deliberate. Accidental threats may involve hardware and power failure whereas deliberate may involve activity by people inside or outside an organization such as employee sabotage or malicious hacking.

Environmental threats may be mitigated by running two or more redundant IRSs at physically distant sites. Accidental threats may be addressed by passive and/or active redundancy approaches to the IRS with an uninterruptible power-supply system (UPS).

Deliberate threats from inside range from malicious power-off to physical destruction of the IRS. These may be addressed by isolating the IRS in a secure server room that requires simultaneous authentication of multiple administrators for access. Deliberate threats from outside may involve cyberattacks, including exploits or DoS attacks. DoS attacks by flooding the network with requests can be mitigated using anycast addressing, although the IRS cannot stop attackers from sending massive numbers of requests. The other attacks may be addressed by security modules of the IRS and is described in Section 5. Although many types of cyberattacks may be blocked by a firewall equipped with an application gateway, some firewalls are at risk of SPOF from cyberattacks such as BlackNurse [23], which is a special type of ICMP flooding attack against some well-known firewall products.

4. Intrusion-Resilient System.

4.1. Overview of an intrusion-resilient system.

An IRS can include three configurations (Figure 1). The first is a standalone IRS to enhance resilience against cyberattacks and is described in detail in Section 4.2. The second is a fault-tolerant system consisting of

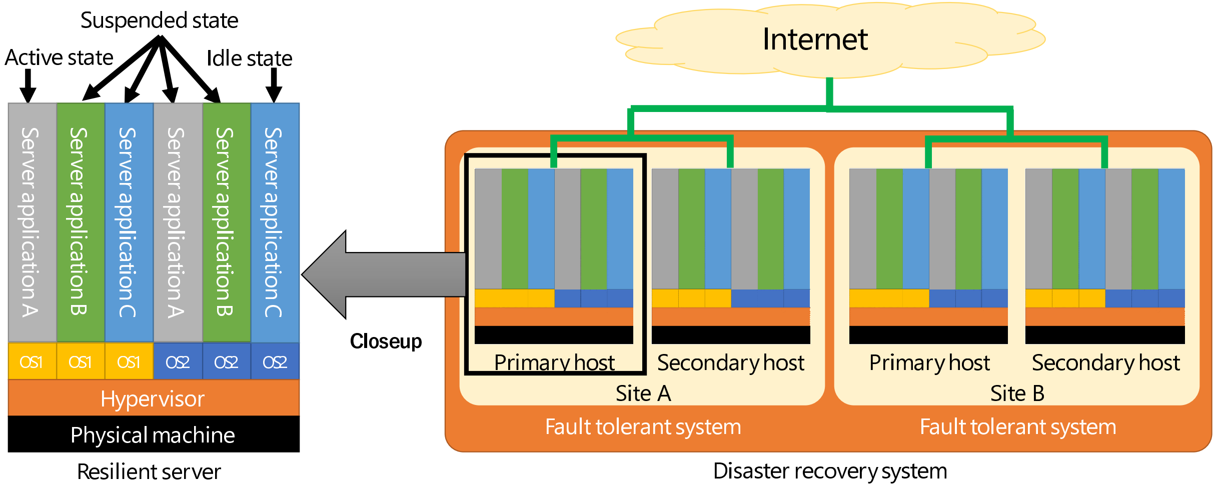

FIGURE 1. Overview of an IRS

two or more redundant standalone IRSs to tolerate accidental failures. It is recommended to design the fault-tolerant system with two or more different hardware specifications to avoid failure caused by the hardware specification. Note that it is necessary to ensure compatibility between the different hardware. The third configuration is a disaster recovery system consisting of two or more redundant fault-tolerant systems at physically distant sites to recover from a natural disaster such as an earthquake or a flood.

4.2. **Standalone intrusion-resilient system.** A standalone IRS utilizes heterogeneous virtual machines running different operating systems and different server implementations of the same server protocol specification. This system enhances resilience against cyber-attacks, because not all implementations will be affected by the same vulnerability, unless it is one in specifications or shared libraries. Therefore, the standalone IRS requires at least four virtual machines comprising a combination of two different operating systems and two different server implementations of the same server protocol specification. 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 them, the maximum combinations of operating system and server implementation is 80, a number thought to make a server significantly more resilient. Note that the combination must not include a common shared library, because the library may have a vulnerability posing a security hole. In fact, the "HeartBleed" vulnerability in the OpenSSL shared library, i.e., CVE-2014-0160, affected many implementations on a Web service.

4.2.1. *Virtualization.* Server virtualization can be divided into machine virtualizations and application virtualizations. Well-known software for machine virtualization includes VMware ESXi, Xen, and KVM, and that for application virtualization includes Docker on Linux and jail on FreeBSD.

Machine virtualization provides a virtual machine capable of running various operating systems. This enables the IRS to diversify operating systems at the expense of resources, such as memory and computing power.

The application virtualization provides a virtual runtime environment comparable to a runtime environment on a physical machine but requires considerably fewer resources and a smaller overhead. The application virtualization only runs a program with its own network stack, with its process of the program isolated from a host system with security constraints. For example, our previous work showed that Docker could start a server application approximately ten times more quickly, while consuming 63.2% less memory, than Xen [24]. However, the application virtualization cannot diversify operating systems, since application virtualization depends on features provided by an operating system. Therefore, the IRS applies a hybrid virtualization (Figure 2), which uses both machine and application virtualizations. Machine virtualization diversifies the operating systems, while application virtualization diversifies server implementations on every virtual machine. Hybrid virtualization provides the best performance with minimum resources in a standalone IRS.

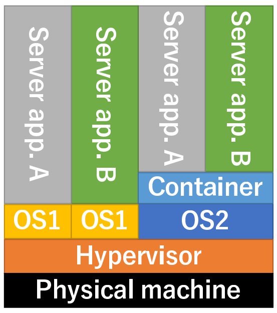

Figure 2. Configuration of virtual machines on a hypervisor

4.2.2. *State transitions of virtual machines.* Each virtual machine can be in any of three states: active, idle, or suspended. An active machine is identical to a primary server, while the idle and suspended machines resemble backup servers. While the active and idle machines run on the standalone IRS only the former provides its service. The suspended machines are queued on standby.

Each virtual machine includes a security module for intrusion protection that inactivates its service if a cyberattack is detected, following which the idle machine is promoted to active, resuming the service.

4.2.3. *Control of virtual machines.* Either a hypervisor or a virtual appliance manages virtual machines using a service monitor and a virtual machine changer. The service monitor periodically checks the service status of the active machine. If the service monitor detects an outage, it notifies the virtual machine changer, which in turn suspends the active machine labeling it as compromised. Meanwhile, the idle machine is promoted to active and it starts providing its service while a suspended machine moves up the queue to become the idle machine. Finally, the virtual machine changer notifies administrators and/or security analysts about the cyberattack. Consequently, the standalone IRS safeguards its service even if a vulnerability in its server applications is compromised. Figure 3 shows the switching flow when the active machine detects a cyberattack.

When administrators and/or security analysts receive notification of the cyberattack, they can investigate the compromised machine and analyze the process memory of the suspended server application. If they can identify the vulnerability and find a security update, they can apply it and add the machine to the queue. If all virtual machines were compromised, the standalone IRS would suspend the service until administrators and/or security analysts solved the problem.

5. **Implementation of an Intrusion-Resilient System for a DNS Service.**

An IRS for a DNS service was implemented to evaluate continuity of the service (Figure 4). Since the active redundancy approach may cause downtime during switchover from the

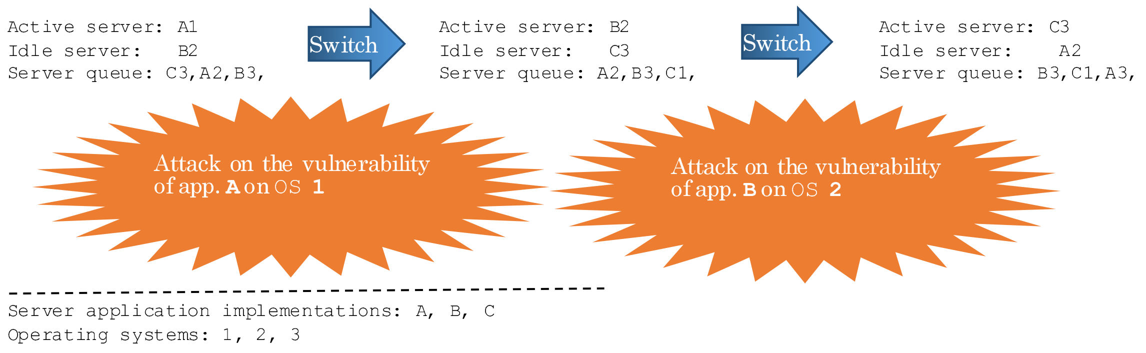

FIGURE 3. Switching flow of virtual machines

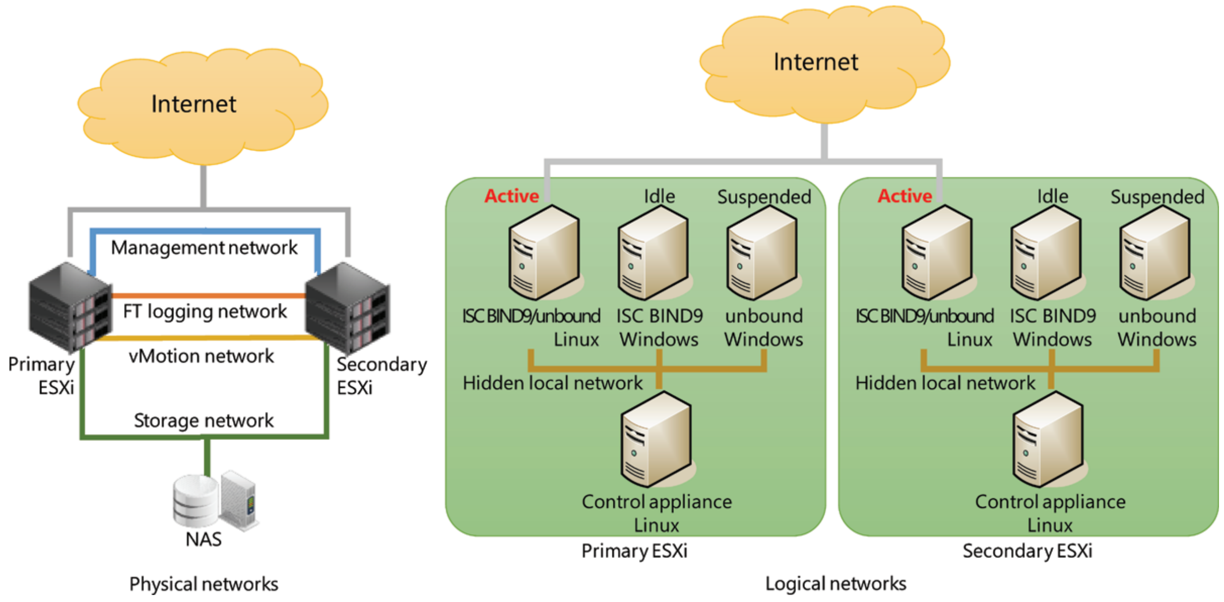

FIGURE 4. An IRS for a DNS service: physical (left) and logical (right) network topology

primary to backup server, the innovations of this implementation include configurations and methods to perform high-speed switching between the active and idle machine that enables stab resolvers to complete name resolution without failure during switchover. This section describes the configurations of physical machines and virtual machines, and the methods to perform high-speed switching between application servers without failure.

5.1. **Overview of an intrusion-resilient system for a DNS service.** The IRS comprises a primary and a secondary hypervisor and a network attached storage (NAS). Two hypervisors powered by VMware ESXi 6.0, included in VMware vSphere 6 Enterprise Plus, run four virtual machines. One is a control appliance, and the others are application servers. All virtual machines have fault tolerance using vSphere FT, which is a feature included in VMware vSphere 6 Enterprise Plus. The NAS is powered by OpenMediaVault, which shares tiebreaker and configuration files of virtual machines between two ESXi hosts for fault tolerance. The specifications of a physical machine for the hypervisors and the NAS are:

- Hypervisor
  - CPU: AMD Opteron 6234 2.4 GHz × 2 CPUs
  - Motherboard: Supermicro H8DGi

  - Memory: 32 GB
  - Network: GbE × 2 and 10 GbE × 2
- NAS
  - CPU: AMD Opteron 2427 2.2 GHz × 2 CPUs
  - Motherboard: Supermicro H8DI3
  - Memory: 16 GB
  - Network: GbE × 2 and 10 GbE × 2

5.2. **Application servers.** The IRS has four application servers comprising a combination of two different operating systems and two different server implementations of a DNS service. One operating system is Windows Server 2012, and the other is Ubuntu 15.10. One server implementation is ISC BIND 9, and the other is unbound. Ubuntu 15.10 runs two server applications using Docker to reduce downtime. This combination has no vulnerability common to both operating systems and both server applications. These operating systems are run on ESXi hosts.

5.2.1. *Windows virtual machine.* One virtual machine runs ISC BIND 9 as a DNS server application and the other runs unbound. All server applications are equipped with a security module, called SecondDEP [25], which detects and prevents the execution of shellcode by cyberattacks, based on evidence that shellcode calls Windows APIs from a data area. If SecondDEP detects a cyberattack, it suspends the server application process to enable administrator and/or security analysts to analyze the attack. Simultaneously, the service monitor detects a service outage, because SecondDEP suspends the server application 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, is set to zero (default: 3), reducing the time required to connect a virtual network adapter to Windows.

The configuration of the Windows virtual machines is:

- OS: Windows Server 2012
- CPU: 1 CPU × 1 Core
- Memory/Disk: 4 GB/40 GB
- SCSI controller: VMware paravirtual
- Disk provisioning: Thick provisioning
- NIC: VMXNET3 enabled RSS × 2 (One connected to the Internet and the other to a local network)

5.2.2. *Linux virtual machine.* A virtual machine runs both ISC BIND 9 and unbound on Docker 1.10.3. Docker makes it faster to switch between containers and requires fewer resources, such as memory and storage. A disadvantage of Docker is deterioration in security strength, and hence all containers 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. Instead of AppArmor, SELinux and seccomp may be used to enhance the security of Docker containers. When AppArmor denies access, it logs the denied access to a log file, following which an `ommail` module emails it to administrators and/or security analysts. Simultaneously, an `icrond` daemon switches the current container to the next container (see Section 5.3.2), because the server application may lose its own normal execution control.

The configuration of a Linux virtual machine is:

- OS: Ubuntu 15.10 (Kernel: 3.19.0)
- CPU: 1 CPU × 1 Core

- Memory/Disk: 2 GB/40 GB
- SCSI controller: VMware paravirtual
- Disk provisioning: Thick provisioning
- NIC: VMXNET3 × 2 (One is connected to the Internet and the other is connected to a local network)

5.3. **Control appliance.** A control appliance is a Linux virtual machine on ESXi hosts, which is only connected to a local network to protect it from cyberattacks via the Internet. The control appliance runs a service monitor and a virtual machine changer that are implemented using vSphere SDK for Perl, which provides APIs to control virtual machines.

The service monitor checks whether the active machine is providing its service via the local network by executing the command `nanonslookup`, which we implemented to support an arbitrary timeout with a nanosecond resolution, specifying a timeout of 0.5 sec every 0.5 sec. If the command times-out three times consecutively, the service monitor determines that the active machine is failing and notifies the virtual machine changer. In addition, the service monitor subsequently checks whether the number of threads per second is equal to the maximum number of threads. If the number is equal to the maximum, the service monitor determines the active machine is under a DoS attack and notifies the virtual machine changer.

After receiving notification, 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 virtual machine changer saves two sessions of a primary ESXi host and an active machine using `Vim::save_session` and `ValidateCredentialsInGuest`, which are functions included in vSphere SDK. These sessions are reused to reduce the time to establish a session.

The configuration of a control appliance is:

- OS: Debian GNU/Linux 8.5 (Kernel: 3.16.0)
- CPU: 1 CPU × 1 Core
- Memory/Disk: 512 MB/4 GB
- SCSI controller: VMware paravirtual
- Disk provisioning: Thin provisioning
- NIC: E1000 × 1

5.3.1. *Switchover between virtual machines.* When switching between virtual machines, the virtual machine changer disables the active machine and promotes an idle one to active. Switchover between virtual machines is divided into three combinations: Windows → Windows, Windows → Linux, and Linux → Windows.

In the first switchover, the active machine is disabled by changing the IP address of the active machine to that of an idle one. The IP address is changed by executing the following command with administrator privilege on the active machine though the `StartProgramInGuest` API included in vSphere SDK.

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

An idle machine is promoted to active by changing the IP address of the idle machine to that of the active one. The IP address is changed by executing the following command with administrator privilege on the idle machine though the `StartProgramInGuest` API.

```
> netsh interface ip set address "Ethernet0 2" static
  ACTIVE-MACHINE-ADDRESS SUBNETMASK GATEWAY-ADDRESS
```

In the second switchover, the active machine is disabled by executing the same command as the first switchover, following which the idle machine is enabled by bringing a network interface up (changing the IP address prevents a Docker container from receiving queries). The network interface is brought up by executing the following command with root privilege on the idle machine through the `StartProgramInGuest` API.

```
# /sbin/ifup eth0
```

In the third switchover, the active machine is disabled by taking a network interface down. The network interface is taken down by executing the following command with root privilege on the active machine though the `StartProgramInGuest` API.

```
# /sbin/ifdown eth0
```

The idle machine is promoted to active by executing the same command as in the first switchover. After that, the virtual machine changer updates an ARP entry for the new active machine on gateways and the control appliance to ensure that it receives queries. The ARP entry on gateways is updated by sending a gratuitous ARP request using `scapy`, which is an interactive packet manipulation tool with a Python interpreter. The ARP entry on the control appliance is updated by executing the following command with root privilege.

```
# arp -s ACTIVE-MACHINE-MACADDRESS GLOBAL-IP-ADDRESS-FOR-SERVICE
```

At this time, the idle machine is promoted to active machine and commences service, following which a queued suspended machine is resumed in the background using the `PowerOnVM` API included in vSphere SDK and promoted to idle machine. Finally, the compromised machine is suspended in the background using the `SuspendVM`, and labeled as compromised.

5.3.2. *Switchover between containers.* A Linux virtual machine runs all Docker containers bound to different port numbers on the host to more quickly and reliably switch between containers. One container has the port number 53 and the other 60050. For example, the Docker container for ISC BIND 9 is run by executing the following bash script with non-root privilege though the `StartProgramInGuest` API.

```
$ 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 -td 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: "`name`" which assigns a name to the container; "`security-opt`" which specifies a profile of AppArmor for the daemon of the service; "`publish`" which publishes the container's port to the host; and "`td`" option which 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: "`start`" which starts the daemon specified by the "`exec`" option; "`oknodo`" which returns to exit status 0; "`quiet`" which only displays error messages; "`pidfile`" which checks whether a daemon process has created the file specified by its option; and "`--`" option which is a terminator of the `start-stop-daemon` command. 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 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, following which it stops and removes the active container. 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 executing the following bash script with root privilege though the `StartProgramInGuest` API.

```
# 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
# docker stop bind && docker rm bind
```

The first and second commands change the port number on the host to 53. The third command stops and removes the Docker container for ISC BIND 9. Finally, if all machines are compromised, the virtual machine changer stands by until administrators and/or security analysts apply a security update to one of the compromised machines.

5.4. **Administrators and/or security analysts.**

Administrators and/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. If the server application has been suspended or abnormally terminated, administrators and/or security analysts determine whether the server has been attacked, and then analyze the server application process to identify the exploited vulnerability. If they identify the vulnerability and find an appropriate security update, they apply it to the server application and add the virtual machine to the queue of suspended machines on a primary ESXi host.

5.5. **Fault-tolerant system using vSphere FT.**

The IRS has fault tolerance using legacy vSphere FT (a.k.a. vLockStep) (vLockStep requires less resource than vSphere FT of VMware vSphere 6 Enterprise Plus). Legacy vSphere FT enables the IRS to provide a service without downtime due to accidental failures. Legacy vSphere FT is configured by using VMware vCenter Server Appliance 6.0, which provides a centralized platform for managing VMware vSphere environments. Legacy vSphere FT, however, works without vCenter Server once its configuration is completed.

The configurations of ESXi hosts and virtual machines for legacy vSphere FT are:

- A cluster consists of two or more ESXi hosts with vSphere HA;
- A cluster has an FT log and a vMotion network as shown in Figure 4;
- A virtual machine has only a single virtual CPU;
- A virtual machine is located on a shared storage;
- A virtual machine has no snapshots.

The FT log network is used to synchronize the virtual machine between the primary and secondary ESXi host. The vMotion network is used for migration of a virtual machine using vMotion, which provides live migration between ESXi hosts without downtime.

Legacy vSphere FT requires one primary ESXi host and one or more secondary ESXi hosts. When a virtual machine on the primary ESXi host is protected by legacy vSphere

FT, the virtual machine on the primary and secondary ESXi host is synchronized through an FT log and a vMotion network as shown in Figure 4.

If the primary ESXi host experiences an accidental failure, the secondary ESXi host is automatically promoted to primary and one of the remaining ESXi hosts is promoted to secondary ESXi host. The new primary and secondary ESXi host start to synchronize the virtual machine.

6. **Performance Evaluations.**

Critical performance indices include continuity and responsivity of the service provided by the IRS. Continuity is evaluated by measuring the downtime of the DNS service. Responsivity is evaluated by measuring time required to complete name resolution.

6.1. **Downtime.**

Downtime may be caused by cyberattacks and hardware failures. We evaluated downtime due to cyberattacks and hardware failures, respectively.

6.1.1. *Cyberattack.*

We evaluated downtime due to cyberattack of a vulnerability in ISC BIND 9 (i.e., CVE-2016-2776), which allows remote attackers to cause DoS via a specially crafted DNS query. This test assumed the latest security update was not applied to ISC BIND 9. The vulnerability was attacked using a module under development provided by Metasploit framework [26].

Figure 5 shows a max-min-average chart of downtime resulting from 10 trials. The "C-C" label indicates switchover between two containers on a Linux virtual machine, "L-W" indicates switchover between a Linux and a Windows virtual machine, and "W-W" indicates switchover between two Windows virtual machines. The maximum downtime as shown in Figure 5 was 3.2 sec, which is faster than the timeout time of stub resolvers in Windows and Linux (the Linux stub resolver retransmits once after a 5 sec timeout, while the Windows stub resolver retransmits four times after 1 sec, 1 sec, 2 sec, and 4 sec timeouts). Consequently, both stab resolvers showed they did not fail name resolution even when the IRS caused downtime due to a cyberattack.

In addition, the downtime of the IRS was much faster than the downtime of an IRS without high-speed switching (i.e., the downtime required for shutdown of the active

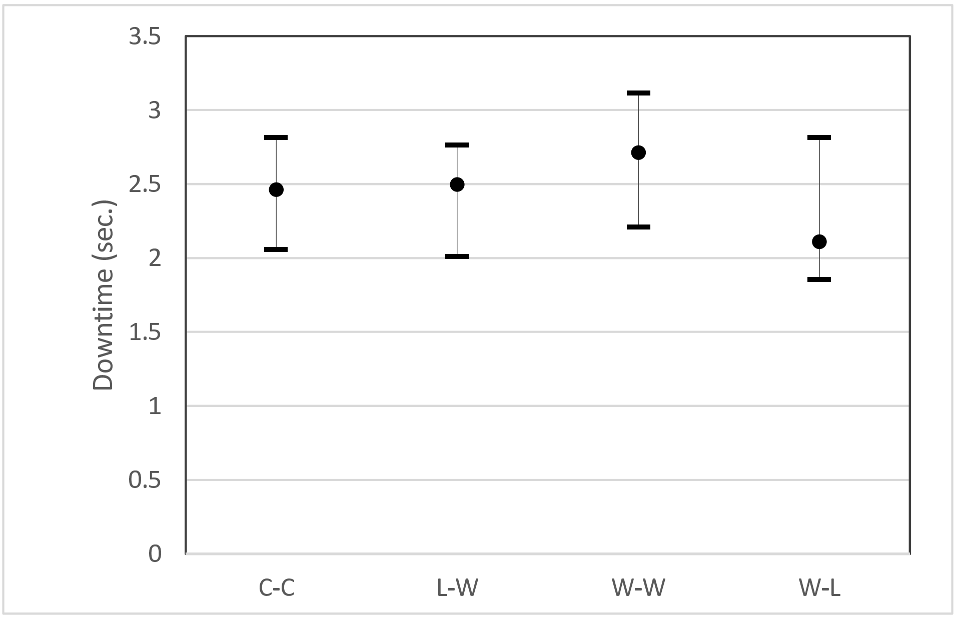

FIGURE 5. Max-min-average chart of downtime for all switchover combinations

machine and start-up of the idle machine [19]), because high-speed switching is performed by only changing their IP address or by bringing up and taking down their network interface.

To investigate the potential for downtime reduction, we investigated a breakdown of the downtime. Downtime comprises time spent checking whether the active machine is providing its service, time spent inactivating the active machine, time spent activating the idle machine, and the waiting time until a server application on a new active machine receives a request. The time spent checking whether the active machine is providing its service is at least 1.5 sec, because the service monitor checks it up to three times, specifying a timeout of 0.5 sec. The time spent inactivating the active machine or activating the idle machine is equivalent to that spent executing a script on their virtual machine. Only switchover between containers is performed by a single script that inactivates the active machine and activates the idle machine, since the script is executed on the same virtual machine. Table 1 shows the average time required for two scripts and the waiting time for four combinations of switchover. The waiting time was longer than the inactivation and activation time in three combinations: "C-C", "L-W", and "W-W". The waiting time of the "C-C" could be accounted for by Netfilter, which offers various functions for packet filtering, network address translation (NAT), and packet mangling, because it requires 0.5 to 1.1 sec to modify the table of NAT rules via the `iptables` command described in Section 5.3.2. The waiting time of the "L-W" and the "W-W" could be due to IP address modification, because a server application on Windows requires 0.2 to 1.2 sec to receive a request just after its IP address is modified. Therefore, it is difficult to reduce further the downtime.

TABLE 1. Average time required for two scripts and awaiting a service (sec)

Time	C-C	L-W	W-W	W-L
Inactivation time	0.115891	0.097500	0.081511	0.086430
Activation time		0.283627	0.301472	0.263776
Waiting time	0.846323	0.616284	0.830690	0.260291
Total time	2.462213	2.497410	2.713672	2.110497

6.1.2. *Hardware failure.* We evaluated downtime due to hardware failure by simulating a hardware failure on the primary ESXi host using VMware vSphere Client via VMware vCenter Server. The IRS has fault tolerance using vSphere FT, which seamlessly executes failover, with detecting hardware failures on the primary ESXi host. Therefore, the downtime due to hardware failure is the same as the time spent executing failover. Table 2 shows statistics of time spent executing failover over 10 trials. The maximum time was 0.442636 sec, but the IRS lost 24.24% of requests for approximately 3.2 sec just after failover. The total downtime was less than the timeout time of stub resolvers in Windows and Linux, suggesting that stab resolvers complete name resolution without fail during failover.

TABLE 2. Statistics of failover time (sec)

Average	Standard deviation	Minimum	Maximum
0.357494	0.040825	0.287857	0.442636

6.2. **Responsivity.** We evaluated responsivity of the DNS service with and without the IRS by measuring the time required to complete name resolution. Table 3 shows statistics of the name resolution time in cases of with and without IRS over 1,000 trials. The average time of the DNS service with IRS was 0.7663 ms per trial, while that without was 0.6167 ms per trial. The time difference between these average times is only 0.1496 ms, which has negligible effect on the performance of a DNS service.

Table 3. Statistics of name resolution time (ms)

Configuration	Average	Standard deviation	Minimum	Maximum
DNS service with IRS	0.766298	0.376761	0.447989	8.314848
DNS service without IRS	0.616717	0.148609	0.344992	2.341986

7. **Conclusion.** We proposed an implementation of an IRS for a DNS service using an active redundancy approach with diversification. Our IRS comprises heterogeneous virtual machines running different operating systems and diverse server implementations of the same server protocol specification. This approach is based on the observation that not all implementations are affected by the same vulnerability except for vulnerabilities in the specification and shared libraries. Other ITSs require a larger number of physical resources proportional to the number of diversified application servers, whereas ours needs to run only two application servers in parallel.

Our IRS completed failover within 3.2 sec, and hence stab resolvers did not fail name resolution during downtime due to cyberattacks. Response time degraded by our IRS was within 0.1496 ms, and hence its overhead has negligible effect on DNS service performance.

Future work includes the use of heterogeneous hypervisors in the IRS to eliminate SPOFs due to cyberattacks on vulnerabilities of hypervisors, and disaster recovery of the IRS using VMware vCenter Site Recovery Manager Enterprise.

REFERENCES

[1] E. Dubrova, *Fault-Tolerant Design*, Springer, 2013.

[2] I. Gashi, P. Popov and L. Strigini, Fault tolerance via diversity for off-the-shelf products: A study with SQL database servers, *IEEE Trans. Dependable and Secure Computing*, vol.4, no.4, pp.280-294, 2007.

[3] M. Garcia, A. Bessani, I. Gashi, N. Neves and R. Obelheiro, OS diversity for intrusion tolerance: Myth or reality?, *Proc. of the 41st International Conference on Dependable Systems and Networks*, pp.383-394, 2011.

[4] F. Wang, F. Gong, C. Sargor, K. Goseva-Popstojanova, K. Trivedi and F. Jou, SITAR: A scalable intrusion tolerance architecture for distributed server, *Proc. of the 2001 IEEE Workshop on Information Assurance and Security*, pp.38-45, 2001.

[5] J. Reynolds, J. Just, E. Lawson, L. Clough, R. Maglich and K. Levitt, The design and implementation of an intrusion tolerant system, *Proc. of International Conference on Dependable Systems and Networks*, pp.285-290, 2002.

[6] A. Saidane, V. Nicomette and Y. Deswarte, The design of a generic intrusion-tolerant architecture for web servers, *IEEE Trans. Dependable and Secure Computing*, vol.6, no.1, pp.45-58, 2009.

[7] H. Seondong, L. Soojin, J. Bumsoon and Y. Hyunsoo, Designing and implementing a diversity policy for intrusion-tolerant systems, *IEICE Trans. Information and Systems*, vol.E100-D, no.1, pp.118-129, 2017.

[8] I. Winarno, T. Okamoto, Y. Hata and Y. Ishida, A resilient server based on virtualization with a self-repair network model, *International Journal of Innovative Computing, Information and Control*, vol.12, no.4, pp.1059-1071, 2016.

[9] A. K. Bangalore and A. Sood, Securing Web servers using self-cleansing intrusion tolerance (SCIT), *Proc. of the 2nd International Conference on Dependability*, pp.60-65, 2009.

[10] F. Sano, T. Okamoto, I. Winarno, Y. Hata and Y. Ishida, A cyber attack-resilient server using hybrid virtualization, *Procedia Computer Science*, vol.96, pp.1627-1636, 2016.

[11] B. Randel, System structure for software fault tolerance, *Proc. of the International Conference on Reliable Software*, pp.437-449, 1975.

[12] C. Pu, A. P. Black, C. Cowan, J. Walpole and C. Consel, A specialization toolkit to increase the diversity of operating systems, *Proc. of ICMAS Workshop on Immunity-Based Systems*, pp.1-10, 1996.

[13] L. Chen and A. Avizienis, N-version programming: A fault-tolerance approach to reliability of software operation, *Proc. of FTCS-8: The 8th Annual International Conference on Fault Tolerant Computing*, pp.113-119, 1978.

[14] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saidi, V. Stavridou and T. E. Uribe, An architecture for an adaptive intrusion-tolerant server, *Proc. of International Workshop on Security Protocols*, pp.158-178, 2002.

[15] F. Majorczyk, E. Totel and L. Mé, COTS diversity based intrusion detection and application to Web servers, *Proc. of the International Workshop on Recent Advances in Intrusion Detection*, pp.43-62, 2005.

[16] L. Nagy, R. Ford and W. Allen, N-version programming for the detection of zero-day exploits, *Proc. of the IEEE Topical Conference on Cybersecurity*, 2016.

[17] Y. Huang and A. Sood, Self-cleansing systems for intrusion containment, *Proc. of Workshop on Self-Healing, Adaptive, and Self-Managed Systems*, 2002.

[18] P. Sousa, A. N. Bessani, M. Correia, N. F. Neves and P. Verissimo, Highly available intrusion-tolerant services with proactive-reactive recovery, *IEEE Trans. Parallel and Distributed Systems*, vol.21, no.4, pp.452-465, 2010.

[19] I. Winarno, T. Okamoto, Y. Hata and Y. Ishida, Distributed SRN manager on a resilient server with multiple virtualization engines, *International Journal of Innovative Computing, Information and Control*, vol.13, no.2, pp.365-379, 2017.

[20] Y. Ishida, *Self-repair Network: A Mechanism Design*, Springer, 2015.

[21] Y. Ishida, *Immunity-Based System: A Design Perspective*, Springer, 2004.

[22] *ISO/IEC TR 13335-1:2001: Information Technology – Guidelines for the Management of IT Security – Part 1: Concepts and Models for IT Security.*

[23] L. Hansson, P. Høgh, B. Bachmann, K. B. Jørgensen and D. Rand, *The BlackNurse Attack*, TDC Security Operation Center, http://soc.tdc.dk/blacknurse/blacknurse.pdf, 2017.

[24] I. Winarno, T. Okamoto, Y. Hata and Y. Ishida, Implementing SRN for resilient server on the virtual environment using container, *Advances in Smart Systems Research*, vol.4, no.1, pp.32-37, 2015.

[25] T. Okamoto, SecondDEP: Resilient computing that prevents shellcode execution in cyber-attacks, *Procedia Computer Science*, vol.60, pp.691-699, 2015.

[26] M. Rocha, E. Tarella, A. Parodi and Infobyte Research Team, *DoS Exploit of CVE-2016-2776*, modules/auxiliary/dos/dns/namedown.rb, Rapid7, https://github.com/rapid7/metasploit-framework/pull/7382/files.