To appear in Proc The Second International Conference on Dependability (DEPEND 2009)June 18-23, 2009 - Athens/Vouliagmeni, Greece

# Securing Web Servers Using Self Cleansing Intrusion Tolerance (SCIT)

Anantha K. Bangalore and Arun K Sood
George Mason University, Fairfax, Virginia
{bangondrum@gmail.com, asood@gmu.edu}

*Abstract*— **The number of malware attacks is increasing, Companies have invested millions of dollars in intrusion detection and intrusion prevention (ID/IP) technologies and products, yet many web servers are hacked every year. The current reactive methods of security have proven to be inadequate because the "bad guys" are always one step ahead of the Intrusion Detection/Intrusion Prevention community. Our research seeks to prove the feasibility of a completely new and innovative theory of server security called "Self-Cleansing Intrusion Tolerance" (SCIT). SCIT shifts the focus from detection and prevention to containing losses. SCIT uses virtualization technology in a new and unique way to make it more difficult for attackers to do damage/acquire data by reducing a server's exposure time from several months to less than a minute. In this way we increase the dependability of the server and provide a new way to balance the trade-off between security and availability.**

**We have applied SCIT to multiple types of servers (DNS, SSO and Web), in this paper we will focus on securing web servers using SCIT. Based on the results of load testing of a web application for various load scenarios under both scit and non-scit environments, we will clearly show that SCIT provides a high degree of security with little degradation in overall response time of the application.**

***Keywords:SCIT,exposuretime,virtualization,vmware,persistence,pro-active,web server,response time***

## I. INTRODUCTION

The complexity of modern information services, and the sophistication, pace, and variety of hacking and malware attack techniques requires a new approach to the challenge of server security. Despite large investments in computer security infrastructure, attackers continue to evade and outperform the most advanced intrusion prevention and detection systems. The problem stems, in large part, from: (1) constant innovation and evolution of attack techniques, (2) rapid development of exploits based on recently discovered software vulnerabilities, and (3) reliance of most defense approaches on detecting "signatures" of an attack to mount an effective defense. The current intrusion prevention or detection approaches require prior knowledge of all potential attack modalities and their own software vulnerabilities. These approaches are good at fighting yesterday's wars, but are totally ineffective against serious current and future threats.

Today's servers are on-line for extended periods – often several months at a time. In general, servers are brought off-line only for patch application or upgrades. Thus, attackers have ample time to explore, experiment and understand target server configurations. In this sense, the servers are sitting ducks, making easy targets for hackers. The SCIT approach is tailored to make it more difficult for attackers to do damage/acquire data by reducing a server's exposure time from several months to less then a minute. The key differences between current approaches (firewall, IDS, IPS) and the SCIT approach can be summarized as follows:

1. Current approaches are reactive and motivated by risk prevention; SCIT is a proactive risk management approach.
2. IPS and IDS depend on a priori information, like attack models and software vulnerabilities. SCIT requires the selection of an exposure time[1] and specification of maximum transaction size. We utilize exposure time as a metric that defines the security – availability tradeoff. For example, higher exposure time leads to less security and more availability.
3. In current approaches, the goodness of the packet requires packet examination. SCIT does not require packet inspect. We emphasize that unlike other intrusion tolerance approaches (see Section ) SCIT does not require intrusion detection – we make no attempt to determine if an intrusion has occurred or not.
4. With reactive approaches, patches for vulnerabilities need to be applied immediately; there is no protection between the detection of the vulnerability and the application of the patch. SCIT provides protection while patches are developed, tested and applied.

Two trends impact on the performance of IDS and IPS – (1) the increasing bandwidth increases the

[1] Exposure time is defined as the time that the server is continuously connected to the internet.

number of packets that must be examined; (2) the number of threats is increasing and this implies that the black list of signatures is increasing. Thus for current IDS and IPS more and more cycles need to be expended for performing packet inspections and comparisons with a black list of threats. The SCIT approach is based on measuring exposure time, and hence is independent of packet inspection time.

Our experiments have shown that SCIT web servers recover very quickly (minute or so) from defacement and software deletion attacks. A video of a demo showing the launching and recovery from such attacks has been posted at [10]. SCIT web servers provide the following protections:

- Delete malware every minute.
- Restore defaced website to pristine state every minute.
- Recover from software deletion attacks every minute.
- Work with IDS to reduce data ex-filtration.

As in the case of any security system, there is a cost associated with using SCIT. A study of the SCIT performance is the focus of this paper. In this paper we will demonstrate that the overhead cost incurred in terms of slower response times by a "*scitized*" system is quite small. The results will also show that lower exposure times will result in slightly higher response times, but yields higher security. Higher exposure times result in lower response times at the cost of security.

The rest of the paper is divided into 6 sections. In Section II we give a quick introduction to how SCIT works. We describe the core components of SCIT architecture relevant to Web servers in Section III. The design of the SCIT Web server is discussed in Section IV. We choose a simple persistence storage based web application running on a tomcat server. Short term persistence based storage is commonly used in web applications to implement shopping cart type functionality. Section V discusses our test methodology, and the test results are in Section VI. We finally discuss the results and related and future work.

## II. How SCIT Works

When a server is booted up, SCIT software launches a pristine, malware-free copy of the server's operating system (OS) into a Virtual Machine. After a certain, potentially random, exposure time to the Internet (usually less than a minute) the virtual server is taken offline and a new, pristine virtual server replaces the prior one. The decommissioned virtual server is wiped clean, loaded with a pristine copy of the OS and placed in a queue for re-activation.

The SCIT research has mainly focused on those servers which are most exposed to malicious intruders. Such servers are located in a network's Demilitarized Zone (DMZ). SCIT focuses on containing any losses resulting from an intrusion without knowing that an intrusion has occurred, i.e. unlike other intrusion tolerant architectures SCIT does not require the intrusion detection step — it just assumes attacks to be continually in progress.

Using virtualization technology, SCIT rotates pristine virtual servers and applications every minute, or less. In Figure 1, we show 3 different time period. At any given time, there are 5 servers online and 3 servers being wiped clean. In each case a different set of servers is being cleaned. Eventually every server will be taken offline, cleaned and restored to its pristine state. [1,2]

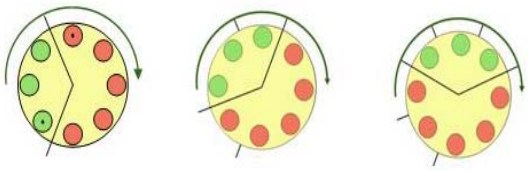

**Figure 1: SCIT Software commissions and decommissions virtual servers at sub minute intervals**.

We emphasize two scenarios that are relevant to the SCIT research.

(1) Single function SCIT servers have at least one virtual server online (red) that is receiving incoming messages, processing these messages, and sending the results; another virtual server (red) not receiving any more incoming messages but finishing up unprocessed requests before this server is cleaned; and a third virtual server (green) has been restored to pristine state and is ready to come on-line, in effect a live spare.

(2) Multiple function SCIT servers have each of the on-line virtual servers performing a different function. Since each of the virtual servers may have different computational loads, we need special algorithms to decide which server will be swapped next.

The key idea of the SCIT approach is to, at a minimum, contain any losses that occur because of an intrusion. SCIT achieves this goal by reducing the exposure time of the server to the Internet.

## III. SCIT Architecture

The 3 core components of the SCIT Architecture are:

1. Virtualization layer - VMware
2. Persistent short term (session) memory
3. SCIT controller

*3.1 Virtualization layer – VMware*

In the past 5 years, the virtualization technology has matured to the point where it is widely being adopted commercially. So far VMware seems to be industry leader in the Virtualization product space. We have implemented and validated SCIT using multiple VMware products. We emphasize that SCIT technology is independent of the virtualization platform.

VMware software provides a completely virtualized set of hardware to the guest operating system. VMware software virtualizes the hardware for a video adapter, a network adapter, and hard disk adapters. The host provides pass-through drivers for guest USB, serial, and parallel devices. In this way, VMware virtual machines become highly portable between computers, because every host looks nearly identical to the guest. In practice, a systems administrator can pause operations on a virtual machine guest, move or copy that guest to another physical computer, and there resume execution exactly at the point of suspension. VMware Workstation, Server, and ESX take a more optimized path to running target operating systems on the host than emulators. VMware ESX (formerly called "ESX Server"), an enterprise-level product, can deliver greater performance than the freeware VMware Server, due to lower system overhead. In addition, VMware ESX integrates into VMware Virtual Infrastructure, which offers extra services to enhance the reliability and manageability of a server deployment.[4]

SCIT prototype servers have been ported to the ESX platform, and our tests have been performed with this system. This effort was necessary to ensure that SCIT technology can be readily applied in an enterprise environment. The ESX server used in our experiments is an Intel Xeon based Dell server with 8CPU's and 4GB of memory.

*3.2 Persistent short term (session) memory*

In large scale systems, e.g. e-commerce, two kinds of "persistent" data is handled. First, the long-term persistent data, like customer order information, is stored for long-term retention on devices like disks, tapes, etc. Second, the short-term session information is retained for the duration of the customer session. Putting this data on the disk is computationally expensive involving disk access related delays, so typically this is retained in main memory.

Handling the persistent session information in SCIT is a challenge. Typically, session information is small and the persistence is limited to the session duration. The difficulty is that in the SCIT servers, the exposed virtual machine is destroyed and a new virtual machine is exposed every minute. In this process the temporary memory is lost. Typically, session information is shared between tomcat servers either by using the multicast protocol or by using an external server which can very quickly store and retrieve short term transient session information associated with a web application. We used an open source version of a tool called Terracotta for our purpose.

Terracotta is based on the concept of Network Attached Memory (NAM). NAM is best suited for storing short term memory, like session data. A NAM implementation as defined by terracotta must meet 2 requirements: (i) NAM must look just like RAM to the application - Constructors, Wait / Notify, synchronized(), == and .equals() should all work as expected. (ii) NAM must work as an infrastructure service, i.e. NAM must run as a driver inside the JVM but also as a separate process apart from the application cluster. This is because, like networked file systems, the memory must survive whether or not your application is running. [5]

*3.3 SCIT Controller*

The SCIT controller is the central component of the SCIT architecture. The controller is a java program that controls the rotation and exposure times of the Virtual Machine. The controller is installed on a secure machine within the internal network of an organization. In our current implementation, during a single cycle of rotation each of the virtual machines are in one of the following states:

1. Active: virtual machine is online and accepts/processes any incoming requests.
2. Grace Period: virtual machine processes any existing requests, but does not accept any new requests.
3. Inactive: virtual machine is offline.
4. Live Spare: virtual machine has been restored to pristine state and is ready to come on-line.

The transitions between the states are shown in the following state diagram.

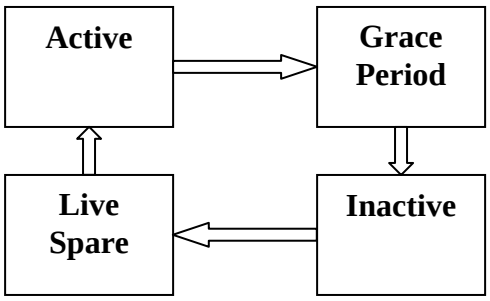

**Figure 2 SCIT virtual server state diagram**

In each rotation only one VM is on-line accepting queries. The state of this VM is considered Active. The other VM's will be in one of the above mentioned states except the load balancer which will at any given time be pointing to only one VM in Active state.

## IV. SCIT Persistent Session Webservers

Persistent session tracking enables web servers to track a user's progress over multiple servlets or HTML pages, which, by nature, are stateless. A session is defined as a series of related browser requests that come from the same client during a certain time period. In a typical e-commerce application, a user adds or removes items from a shopping cart, while browsing through the site's inventory. This shopping cart is active only during this session. and is considered transient. i.e. only valid during the lifetime of the session. Once the user is done shopping, he or she checks out . At this point the user is billed for the contents of the shopping cart.

In order to "scitize" an application that uses persistent sessions, the session information has to be shared between the multiple copies of the web servers running on each of the virtual machines that are part of the SCIT cluster. We use a simple web application to demonstrate that any persistent session based web application can be scitized.

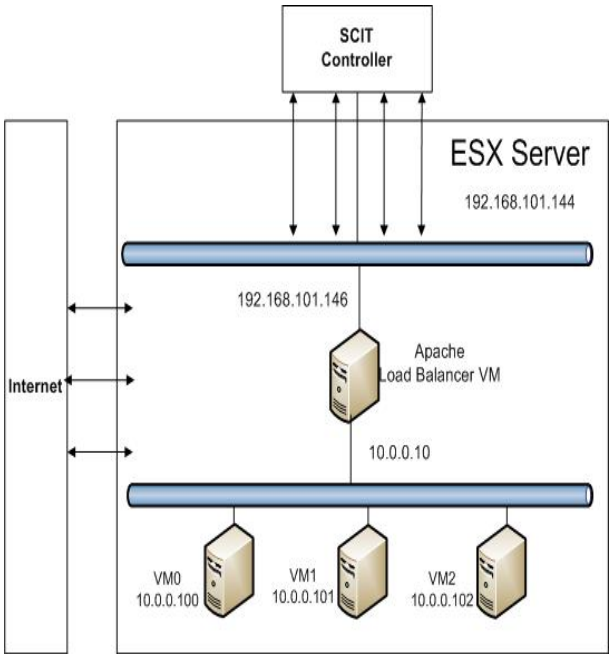


**Figure 3 SCIT Web Server**

As shown in figure 3, in our SCIT set up we have 3 application virtual machines VM0, VM1 and VM2. Each of these 3 virtual machines is running Slackware11 as the Operating System. Each of the VM's are also running Jakarta tomcat 5.5.12 as the web server. The web application is running under tomcat web server. The Load Balancer VM LB is running Cent OS as the operating system. It is also running the terracotta server. Ideally the terracotta server should be running in a separate virtual machine on the host only network. An Apache server running on the LB VM performs the function of load balancing across the 3 tomcat servers. The tomcat servers on startup connect to the terracotta server. All the VM's are running on a stand alone VMware ESX server. VM0, VM1 and VM2 are connected to a host only network. This ensures that these VM's that are hosting the application cannot be directly accessed from the internet. The LB VM is connected to a bridge network as well as to the host network to which the application VM's are connected. The LB VM on the bridge network can be accessed from the internet. During each rotation cycle the load balancer points to only one of the application VM's. This VM is considered to be in the active state. After the active VM is exposed for a predetermined exposure time, it is taken off-line to be cleaned and returned to the previous clean state of the VM. When the VM is taken off-line, it is said to be in the inactive state. During the same period, the LB is made to point to what was the live spare VM, which at this point becomes the active VM.

## V. Performance Tests

The average query response time[2] is used as the performance measure in our tests. We employ an open source load generation tool called Open System Testing Architecture (OpenSTA) for workload generation. OpenSTA is a distributed software testing architecture designed around CORBA. The current toolset of OpenSTA has the capability of performing scripted HTTP and HTTPS heavy load tests with performance measurements from Win32 platforms. To simulate a realistic test, we model the typical scenario including "think" time necessary in between transactions.

OpenSTA remotely generates a work load of virtual users whose life cycle is to complete a script of HTTP requests and responses. The start and completion time of the test script is recorded for each user by OpenSTA for later analysis.

[2] In our analysis the response time is the time taken for completing each session. Details of the activities in a session are described below.

In each of our tests we use the following framework:

1. The workload is measured in terms of the number of users per minute. In each experiment we use 3 levels for users per minute (U). We tested for scenarios involving 50, 100, 125 users/min.
2. Each user session includes a series of requests and responses from the server. We model the "think" time that is required between each of the requests. The "think" time for the web server averages 2 seconds between requests Each session involves selecting an item from a drop down list and adding it into the persistent storage. For our test, in each session we add three items to the persistent storage.
3. OpenSTA is used to generate workload for each of the scenarios. To minimize the impact of the random behavior for each case we conducted 3 runs. The duration (D) of each run is three times the exposure time (E), D = 3 * E. This choice makes sure that each virtual server is tested at least once. Higher values of D are advisable for future testing. The workload is generated in batches. Each batch consists of N user requests, and a new batch is released every 10 seconds. Thus U = N * 6, and total number of requests in a scenario (T) is given by T = U * D.
4. We choose an exposure time – minimum exposure time is 1 min, but as the server computational complexity increases a higher exposure time is necessary. The exposure time for tests was 2, 3 and 4 minutes. We also conducted a test in which there was no rotation of the servers. This can be viewed as a baseline test, and the performance can be compared to this baseline.
5. A total of 36 runs were performed.
6. For testing in a non-scit environment we set up a tomcat server running on a stand alone dual core machine with the same simple persistence storage based web applicationAs in the case of the tests conducted for the "scitized" web server, OpenSTA was used to generate workload for each of the scenarios. To minimize the impact of the random behavior for each case we conducted 3 runs. The workload is generated in batches. Each batch consists of N user requests, and a new batch is released every 10 seconds.
7. OpenSTA crashed for scenarios involving excess of 250 users/minute.

## VI. TEST RESULTS

**Table 1: Average Response Times for SCIT Persistent Web server tests**

| Exposure Time | Users | Average Response Time | STD. Dev. |
|---|---|---|---|
| 2 mins | 50 | 6.16 | 0.07 |
| 2 mins | 100 | 6.10 | 0.05 |
| 2 mins | 125 | 6.08 | 0.04 |
| 3 mins | 50 | 6.03 | 0.01 |
| 3 mins | 100 | 6.02 | 0.02 |
| 3 mins | 125 | 6.24 | 0.01 |
| 4 mins | 50 | 6.16 | 0.02 |
| 4 mins | 100 | 6.15 | 0.02 |
| 4 mins | 125 | 6.03 | 0.00 |
| NR | 50 | 6.02 | 0.01 |
| NR | 100 | 6.27 | 0.02 |
| NR | 125 | 6.31 | 0.05 |
| Standalone | 50 | 6.14 | 0.02 |
| Standalone | 100 | 6.04 | 0.00 |
| Standalone | 125 | 6.02 | 0.00 |

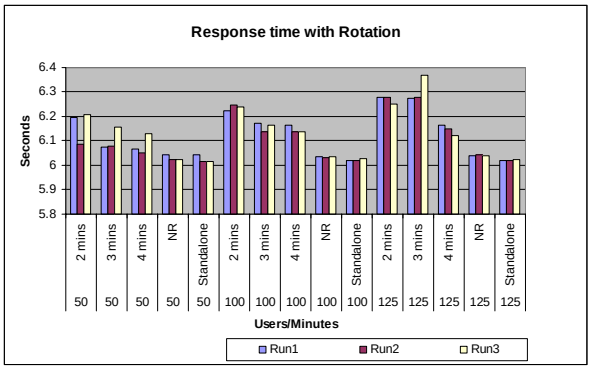


**Figure 4 Average User Session Response Time for different workloads**

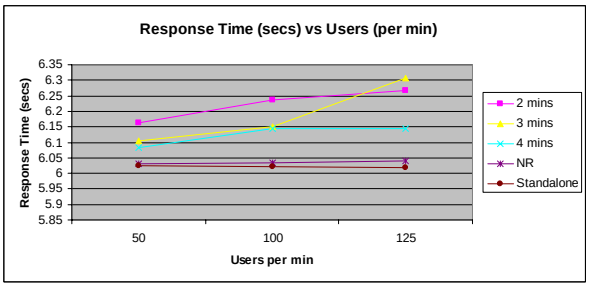


**Figure 5 Response Time for users/min**

Based on the results summarized in the table and graph above, for the exposure time of 2 minutes, the response time is slightly higher than exposure time of 3 minutes which in turn is higher than exposure time of 4 minutes. The No Rotation (NR) response time seems to be closer to the 4 minute exposure time as is the response times for the tests conducted on the stand alone server (SA) in a non-scit environment. There is little difference between the no rotation response times and the response times conducted under the non-scit environment. We note that the last column contains the standard deviation for each of the test scenarios. This supports our hypothesis that low exposure times lead to larger response times, but the low exposure times of the VM's lead to increased security of the scitized server. On the other hand, increasing exposure time, leads to lower response time, but the exposed servers are obviously more susceptible to being hacked under higher exposure times. Based on the results, we conclude, the impact of scit on performance is mild.

## VII. Related and future work

Intrusion tolerance is a new approach that has slowly emerged during the past decade, and is steadily gaining momentum in the field of security. Some of the related work in this domain are:

1) MATIA: Malicious- and Accidental-Fault Tolerance for Internet Applications is a European project that aims to build conceptual models, mechanism and protocols for achieving tolerance.[9]

2) OASIS: Organically Assured and Survivable Information Systems was a DARPA project. A number of intrusion tolerant architectures were developed in this program[6]. DPASA was a result of this effort.

3) SITAR: Scalable Intrusion-Tolerant Architecture for Distributed Services was developed at Duke University aims to use redundancy to reconfigure systems to increase security.[7,8]

As mentioned before, along with Web servers, we have successfully applied SCIT to DNS[3] as well as Single-Sign-On systems. In the future we want to apply SCIT to other servers (Email, etc). We are also looking at methods by which the exposure time is reduced even further. One way of achieving this is perhaps by having multiple VM's in live spare mode at any given time. In our experiments to secure web servers, all the VM's run the same OS with the same configuration. Hackers could potentially exploit this fact. This can be mitigated by using diversity principals. In each rotation cycle the operating system, application or the memory image can be changed to further confuse the intruder. This approach also makes it more difficult for one virtual server to infect another virtual server on the host network. Another point of vulnerability is malicious alteration of the session information that is shared among all the virtual servers The SCIT solution does not provide any additional capability for the intruder to maliciously alter the session data. The constant rotation potentially reduces the scope of the alteration. We note that the session information is relatively small and well structured. This enables the validation of the data as it is transferred from one server to the other. We plan to address these issues in future implementations of SCIT.

## Acknowledgment

This research was partially supported by a contract from Lockheed Martin and a contract from Virginia Center for Innovative Technologies.

## References

[1] Yih Huang, Arun Sood, and Ravi K. Bhaskar, "Countering Web Defacing Attacks with System Self-Cleansing", Proceedings of 7th Word Multiconference on Systems, Cybernetics and Informatics, pp. 12?16, Orlando, Florida, July 2003.

[2] Yih Huang and Arun Sood, "Self-Cleansing Systems for Intrusion Containment", Proceedings of Workshop on Self-Healing, Adaptive, and Self-Managed Systems (SHAMAN), New York City, June 2002..

[3] Yih Huang, David Arsenault, and Arun Sood, "SCIT-DNS: Critical Infrastructure Protection through Secure DNS Server Dynamic Updates", Journal of High Speed Networking, vol 15 No 1, pp 5 19, 2006.

[4] VMware documentation, http://www.vmware.com

[5] Terracotta documentation, http://www.terracotta.org

[6] Foundations of Intrusion Tolerant Systems, Edited by Jaynarayan H. Lala, DARPA Organically Assured and Survivable Information Systems (OASIS), IEEE Computer Press, 2003.

[7] Feiyi Wang; Jou, F.; Fengmin Gong; Sargor, C.; Goseva-Popstojanova, K.; Trivedi, K.; "SITAR: a scalable intrusion-tolerant architecture for distributed services", Foundations of Intrusion Tolerant Systems, 2003 Page(s):359 – 367.

[8] Dazhi Wang, Bharat B. Madan, and Kishor S. Trivedi, "Security Analysis of SITAR Intrusion Tolerance System", Proceedings of the 2003 ACM workshop on Survivable and self-regenerative systems: in association with 10th ACM Conference on Computer and Communications Security, Fairfax, Virginia.

[9] I. Welch, J. Warne, and P. Ryan, and R. Stroud, "Architectural Analysis of MAFTIA Intrusion Tolerance Capabilities". Technical Report CS-TR-788, University of Newcastle upon Tyne. Feb 2003.

[10] http://www.youtube.com/watch?v=gIN6JWInuv8