

Countering Web Defacing Attacks with System Self Cleansing

Yih Huang

Computer Science Department
George Mason University
Fairfax, VA 22030
(703) 993 1540

huangyih@cs.gmu.edu

Arun Sood

Computer Science Department
George Mason University
Fairfax, VA 22030
(703) 993 1524

asood@cs.gmu.edu

Ravi K. Bhaskar

Information and Software
Engineering Dept.
George Mason University
Fairfax, VA 22030

rbhaskar1@gmu.edu

ABSTRACT

Web defacing is a form of system intrusion that aims to subvert the contents of a web site. In this paper, we present a defense mechanism that is based on *high availability computing*, whereby a backup server is available to immediately take over in the presence of server failures.

Our approach, called *Self-Cleansing Intrusion Tolerance (SCIT)*, pushes the concept of high-availability computing one step further. In the SCIT approach, a web server is periodically assumed to have "failed," namely, comprised by undetected intrusion/subversion. Consequently, the server is brought off-line for cleansing and the integrity checking of system files and web contents, while a backup takes over. Indeed, it is more appropriate to see a SCIT system as two mirror servers working alternatively than as a primary server and its backup. In this paper, we define the concept of SCIT and present our experiences in building a SCIT web server prototype. . Our prototype results show that self-cleansing cycles are in the range of minutes, restricting the attackers to a very short time window to breach the system and subvert web contents.

Keywords

high-availability computing, computer security, defacing attack, self-cleansing systems.

1. INTRODUCTION

In this paper, we address the issue of web defacing attacks through system self-cleansing. Web defacing is a form of cyber terrorism that subverts the contents of a web site. Typically, the contents are changed to

show the propaganda of the attacker. At the first glance, "losing face" hardly constitutes terror. However, the seriousness of the attack is twofold. First, since the users of the web site are deprived of the normal contents of the site, a defacing attack causes the denial of services. Thus, while the defacing of a charity site could be considered relatively harmless, the denial of access to important e-commerce sites can be disastrous. Second, a successful defacing attack reveals severe vulnerabilities in system security. The damage of the attack could be a lot more than what one sees on the surface. If the hacker is able to change web contents, Trojan horse codes could have been installed too, for instance.

Traditionally, the integrity of web contents (as well as that of the whole system) is protected by Intrusion Management Systems (IMS). IMS is based on intrusion prevention and detection followed by recovering and cleanup procedures [Bish99,CIDF]. Such an IMS approach relies heavily on the ability to detect intrusion events in the first place.

We however made the pessimistic but realistic assumption that not all intrusion activities can be detected and blocked (at least not in a timely manner to avoid significant damage) and seek technologies to build secure systems which constantly assume the compromise of the system and perform self-cleansing, regardless of whether intrusion alarms actually occur or not. We believe that such an assumption is appropriate given the sophistication and rapid evolution of information warfare.

One implementation of self-cleansing involves rebooting the subsystem from a trusted storage device followed by, if necessary, system recovery, checkpoint, rollback, and data integrity checking routines. (A trusted storage device can be either a

read-only storage device or any nonvolatile storage where information is cryptographically signed.) System availability is achieved by means of redundancy, that is, a backup system is brought online to provide services. In our previous work, we have designed and implemented a SCIT firewall [HuSo02]. In this paper, we apply the SCIT concept to the more sophisticated systems of web servers. Our goal is to protect the integrity of critical system files and a set of static web contents.

Our SCIT Web Server architecture is depicted in Figure 1. As seen in the figure, two server boxes called *w-boxes* are used to build one SCIT web server. Each server box is a complete computing device, including local storage devices for the operation system and application programs. The two boxes have access to a shared backend storage that is used to store contents of the web site. When one box is operating, the other will be performing self-cleansing by rebooting itself, followed by integrity checking of system files and web contents using digital signatures. Moreover, the life spans of these signatures are limited to one self-cleansing cycle, that is, new signatures are produced each time a server box is rebooted. As such, even if an enemy managed to break into a server box, its control over that box is limited to only one self-cleansing cycle, within which the enemy must break the signature keys in order to sign bogus contents. In our prototype, a cycle is typically shorter than 7 minutes.

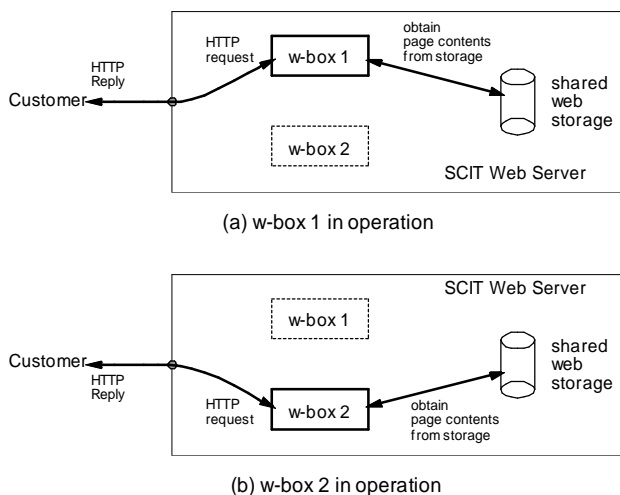


Figure 1. SCIT Web Server

We point out that SCIT complements and strengthens existing intrusion prevention and detection

technologies; we do not exclude the use of the current intrusion management systems, but rather add another layer of defense, extending the idea of system "defense-in-depth" through periodic system cleansing. The effectiveness of SCIT web servers depends on fast self-cleansing cycles, restricting the attackers to very short time windows to breach the system and subvert web contents.

The remainder of the paper is organized as follows. In Section 2, we discuss the relation between SCIT and high-availability computing. The design and performance of our prototype are presented in Section 3. We discuss in Section 5 the application of our approach to web sites that maintain session state. Conclusion is given in Section 6.

2. HIGH-AVAILABILITY COMPUTING

A high-availability system typically uses backup systems to ensure continued customer services in face of primary server failures [Weyg96, Linux-HA, BlNg01, RaMB01]. SCIT servers share many design challenges with high-availability systems, such as the seamless takeover by the backup, sharing of server identities (IP and/or hardware addresses) between the primary and backup, and monitoring the primary servers. Indeed, SCIT can be considered as an extension to high-availability systems so that artificial failure events are introduced to force periodic takeover of the backup server and the self-cleansing of the primary server. This view of SCIT justifies its technical feasibility: every high-availability product on the market gives confidence to a corresponding SCIT system following similar designs. Examples of existing high-availability systems include web servers, NFS servers, authentication services, firewalls, and IPsec gateways.

3. PROTOTYPING SCIT WEB SERVERS

Our testbed is based on the Virtual Machine software from VMWare, Inc. [VMware]. The virtual machine technology enables multiple *guest operating systems* to be installed and executed on top of a *host operating system*. In this environment, the two *w-boxes* are virtual machines running the Apache HTTP server and RedHat 7.2 Linux. The shared backend storage is provided by an NFS server running on a third RedHat 7.2 virtual machine. The

underlying host machine is a Pentium 4 PC also running RedHat 7.2. VMWare supports not only virtual machines but also *virtual networks*, which are emulated switched Ethernet networks. As seen in Figure 2, we use this feature to build two virtual networks, one with subnet ID 192.168.181/24 and the other 192.168.225/24. External users access the web server through the latter subnet. A third subnet, 192.168.88/24, is used by the two w-boxes to probe each other.

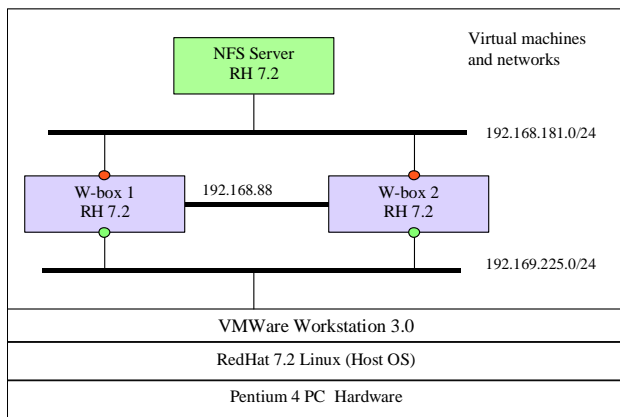


Figure 2. SCIT web server testbed

We use the Virtual Router Redundancy Protocol (VRRP) [RFC2338] to allow the two w-boxes to share the same IP and MAC (Media Access Control) addresses, collectively referred to as a *server identity* hereafter. In our prototype, there are two server identities, one in subnet 192.168.181/24 shared by the “red” interfaces in Figure 2 and the other in 192.169.225/24 shared by the green interfaces. With VRRP, one w-box is the primary and owns the server identities while the other w-box acts as the backup. When the primary reboots, the backup takes over the server identities. In this way, there is no need to use DNS servers or workload distributors to redirect traffic to the currently running server.

To protect against defacing, system files and (static) web contents are protected by digital signatures. While digitally signing critical information is not new, we further enhance its effectiveness by integrating it with SCIT self-cleansing cycles. In our prototype, new signatures are generated at the beginning of every self-cleansing cycle. Thus the enemy has only the time window of one cycle to break the current key. Even if the enemy succeeds in

the task, the time window to inflict damage to the system is limited to one cleansing cycle.

The lifetime of a SCIT web server is given below.

1. Check signature integrities, using previous keys.
2. Generate new keys and re-sign system and web page directories using the new keys.
3. Claim the web IP and MAC addresses by VRRP.
4. Probe/ping the other w-box, until a response is received.
5. Reboot and return to step 1.

We use the Tripwire package [HoDu98,KiSp94] to generate digital signatures and perform integrity checking. Tripwire keys are generated by an Expect [Expect] script. We replicated on the (virtual) NFS server the 40 MB contents of the GMU Center for Image Analysis home page at <http://cfia.gmu.edu>. The digital signatures generated by each w-box cover its local system directories (`/etc`, `/usr/bin`, and so forth) and the web contents on the NFS server. Signatures are generated only for directory contents (recursively), not for individual files. This allows for fast signature generations and the verification of following properties for each file under protected directories:

- Access permissions and file mode settings,
- Inode number
- Number of links
- User ID and Group ID
- Size
- The creation and last modification times

A screen capture of our prototype is shown in Figure 3. In the figure the top left and top right windows are the consoles of the two w-boxes. The top middle window is the console of the NFS server. The top left window shows that the presently running w-box is probing (unsuccessfully) the other w-box (Step 4 above). In the meantime, the top right w-box is generating new keys (Step 2). The left w-box will reboot itself once the right w-box finishes generating new signatures and is ready for operation. The bottom window is a virtual machine that functions as a netscape client attached to subnet 192.169.225/24.

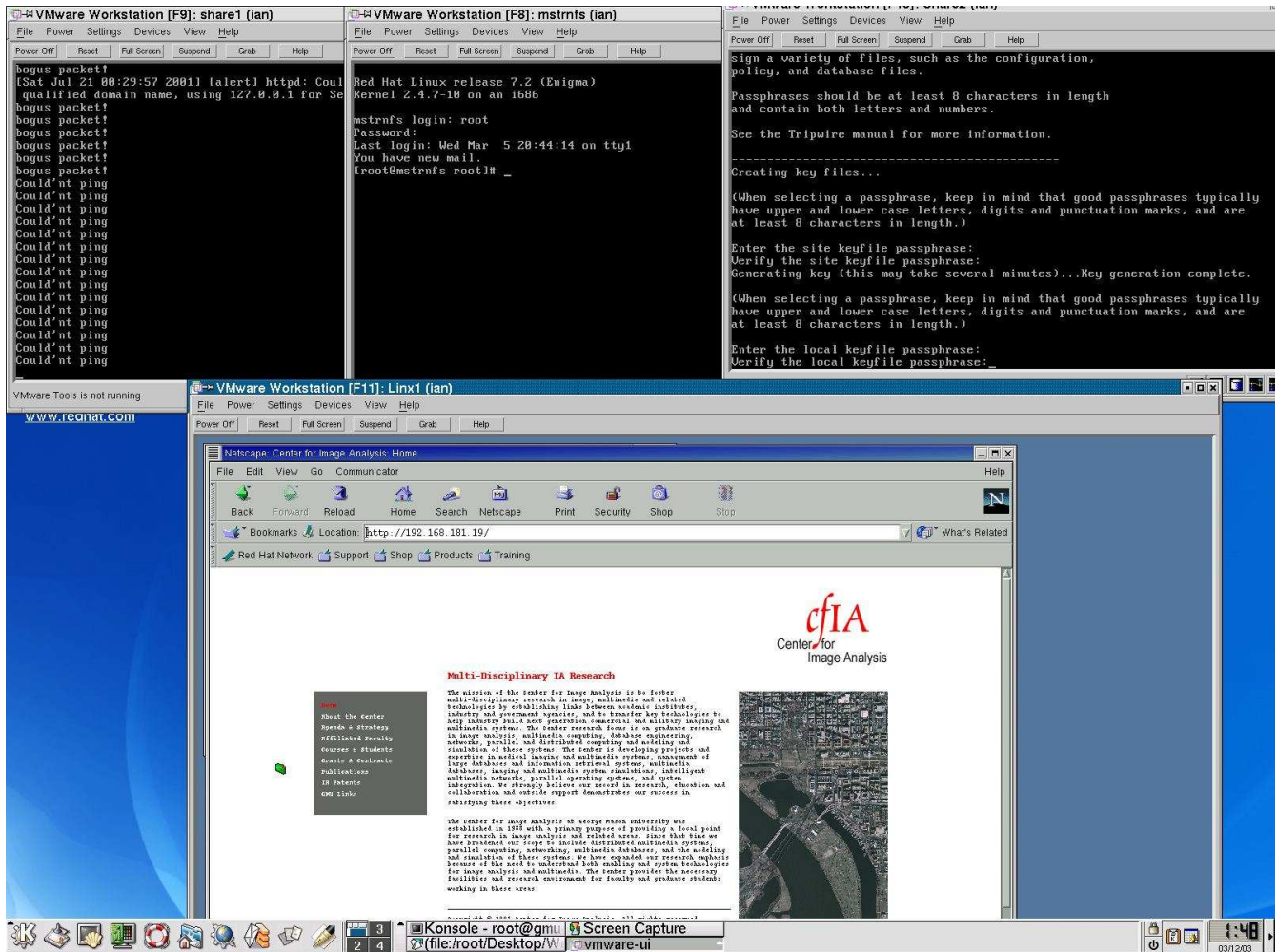


Figure 4. A screen capture of the SCIT web server prototype

With the setup given above, the lengths of self-cleansing cycles are determined by the sum of rebooting, signature checking, key generation, and new signature computation times. In our prototype, self-cleansing cycles are in the range of 6 to 7 minutes. That is to say, to deface the server an enemy has less than 7 minutes to breach the system and break the current signature keys. Notice that the results are achieved by using one PC to sustain the workload of four virtual machines. We believe that systems built with real machines will achieve even shorter cycles.

One concern of SCIT web servers is potential service disruptions caused by server switching. In particular, a client currently being served by a w-box will lose

its connection if the box is rebooted before it finishes the connection. We used web performance evaluation tool httpperf [MoJi98] to investigate this issue. We first determined the maximum number of HTTP calls our prototype could handle per second during periods when there is no server switching (a call consists of an HTTP request and the ensuing reply). We obtained the result of 100 to 105 calls per second. This low number is due to the heavy use of virtual machines in the testbed. Next, we created httpperf sessions that contain server switching (100 calls per second, one session one switching) and obtained the average loss rate of 1.833 per server switching. We believe that this represents a very low level of service disruption to end users.

4. MAINTAINING SESSION STATE

HTTP is a stateless protocol. The handling of each request is independent of that of any other request. This property enables different boxes to serve the requests from the same client and constitutes the foundation of SCIT web servers. Many commercial sites however need to maintain client session state (for what is in a user's "shopping basket" for example). Not all such sites are compatible with SCIT. The evils are in the configuration details of particular servers. Here we consider the Apache server.

With the Apache server, session information can be stored 1) in the main memory of the server, 2) in the file system of the server, or 3) as cookies on client machines. The first approach is incompatible with SCIT, because the rebooting of one server box will lose all session information maintained in the main memory of that box. With the second approach, the file/database that stores session information must be on the backend storage (the NFS server in our prototype), so that both server boxes have access to all sessions. The implementation and testing of this

References

- [Bish99] M. Bishop, "Vulnerabilities Analysis," Recent Advances in Intrusion Detection, Sept. 1999.
- [BINg01] Steve Blackmon and John Nguyen, "High-Availability File Server with Heartbeat," *System Admin, the Journal for UNIX Systems Administrators*, vol. 10, no. 9, September 2001.
- [CIDF] Common Intrusion Detection Framework, <http://www.gidos.org/>.
- [Expect] The Expect Home Page: <http://expect.nist.gov/>.
- [HA-Linux] High-Availability Linux Project. Home page <http://linux-ha.org/>.
- [HoDu98] M. Hosmer and M. Duren, "Detecting subtle system changes using digital signatures," Proceedings of Information Technology Conference, pages 125—128, (Syracuse, NY) Sep. 1998.
- [HuSo02] 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.
- [KiSp94] Gene H. Kim and Eugene H. Spafford, "Writing, Supporting, and Evaluating Tripwire: A Publicly Available Security Tool," in *Proceedings of USENIX Applications Development Symposium*, (Toronto, Canada), April 1994. Also see <http://www.tripwire.com/>.
- [MoJi98] D. Mosberger and T. Jin, "httpperf – A tool for measuring web server performance." in Proceedings of Workshop on Internet Server Performance (WISP) '98, (Madison, Wisconsin), June 1998. Also available at http://www.hpl.hp.com/personal/David_Mosberger/httpperf.html.
- [RaMB01] Richard Rabbat, Tom McNeal and Tim Burke, "A High-Availability Clustering Architecture with Data Integrity Guarantees," Proceedings of IEEE International conference on Cluster Computing, pages 178 – 182, (Newport Beach, California) October, 2001.
- [RFC2338] S. Knight, D. Weaver, D. Whipple, R. Hinden, D. Mitzel, P. Hunt, P. Higginson, M. Shand, and A. Lindem, "Virtual Router Redundancy Protocol." Internet RFC 2338.
- [VMware] VMware Inc. Home page <http://www.vmware.com/>.
- [Weyg96] Peter S. Weygant, *Clusters for High Availability*, Prentice Hall, 1996.

approach in our prototype constitute part of our ongoing research. The third approach is automatically compatible with SCIT, because with the approach the servers are not responsible for maintaining session state.

5. CONCLUSION

We have presented a novel application of high-availability computing, namely, the containment of the web server defacing attacks. Our SCIT web server design uses multiple, identical servers to execute in turn, allowing off-line servers to be checked for the integrity of web contents. These self-cleansing activities occur periodically, regardless the presence/absence of attack alarms. Our prototype results show that self-cleansing cycles are in the range of minutes, restricting the attackers to a very short time window to breach the system and subvert web contents. Our future researches include applying the SCIT concept to other important servers in distributed computing environments, such as NSF servers.