Quantitative Approach to Tuning of a Time-Based Intrusion-Tolerant System Architecture

Quyen Nguyen and Arun Sood
International Cyber Center and Department of Computer Science
George Mason University, Fairfax, VA 22030
qnguyeng@gmu.edu *asood@gmu.edu*

Abstract

Many institutions rely on open systems to provide services to the public via the Internet. Unanticipated software vulnerabilities expose such services to malicious actors, and make them susceptible to attacks. Therefore, security is critical in order to ensure confidentiality, integrity, and availability for system data and services. The fact that security attacks have become increasingly sophisticated makes the protection of open systems more challenging. Current intrusion prevention and detection are reactive, and the bad guys are always one step ahead. In this paper, we will present a quantitative analysis of Self-Cleansing Intrusion Tolerance (SCIT), a time-based intrusion tolerance architecture. Using quantitative techniques we show that it is possible to tune a SCIT system based on its exposure window in order to achieve the required degree of intrusion tolerance.

1. Introduction

Security is critical in today's system design and architecture in order to ensure availability, integrity, and confidentiality. As computer systems are opened to large population of users via web services on the public Internet, the task of protecting them becomes more daunting. The vulnerabilities of such systems expose them to malicious attacks. Adding to the challenge is the fact that security attacks have become more sophisticated, so that a system cannot rely solely on intrusion prevention and detection for its security protection. Therefore, intrusion tolerance systems (ITS) should be part of the solution for securing computer information systems. As distinct from the intrusion avoidance of current systems, ITS systems focus on containing the losses. In this paper we present, model and analyze a new ITS system called Self Cleansing Intrusion Tolerance (SCIT). SCIT is a time-based ITS, and does not rely on intrusion detection.

This paper is organized as follows. In section 2, we present an overview of the SCIT architecture. Section 3 discusses briefly the strengths and weaknesses of the approach in terms of effectiveness, performance and integration into an application system. Related work is surveyed in section 4. The control parameters of the SCIT architecture are discussed in Section 5. In section 6, we derive a mathematical expression for assessing the intrusion tolerance of SCIT, and show how to quantitatively tune the tolerance level based on the SCIT exposure window. The paper ends with a conclusion and discussion of future work in section 7.

2. SCIT Overview

Goal: SCIT architecture is a time-based approach [1]. Since there is no perfect prevention and detection of intrusions, the guiding principle of SCIT design is that security compromises are inevitable. The SCIT design goal is to minimize the losses incurred because of an intrusion.

Basic Concepts: The main idea is to have a group of servers with identical functionalities and services, but with some diversity. The diversity, based on the operating system platform for example, is aimed at making intrusion more difficult. Within this group of servers, round-robin cleansing will be performed.

At any time, a server in a SCIT cluster is in one of the four states:

- Live Spare (LS) server is in a known good state but is offline,
- Active (AC) server is online and accepts and processes requests,
- Grace Period (GP) server finishes processing of pending requests but accepts no new requests,
- Inactive (IN) server is offline and being restored to a known good state.

These four states form a continuous cycle LS→AC→GP→IN→LS during the operational life of a server. Since cleansing a server takes time, the exposure time of an online server depends on when an offline server has completed its cleansing, and is ready

to come back online. The SCIT model has been proven to be applicable to servers that are critical to today's large systems that are exposed to the Internet, such as DNS, and Web servers [3][4].

Architecture Components: The SCIT architecture is simple, and does not require intrusion detection. On one hand, there is a group of servers to be protected. On the other hand, the core component of SCIT is the Central Controller which manages server rotation cycle. The server rotation is performed according to an algorithm, which takes into account the cardinality of the group, the cleansing cycle time of a server, and the number of required online servers at one time. Note that the cleansing time depends on the specific server type. In case that we need more redundancy to achieve higher availability, the number of servers in the SCIT cluster can be increased. The interfaces between the Controller and the group of servers to be protected are trusted and one-way from the Controller to the server group. In [4], the authors have reported that the implementation of SCIT is based on virtualization technology, which offers advantages. First, virtualization allows the instantiation of multiple servers with diverse guest operating systems on a single host machine. Second, logical servers deployed on virtual machines can be brought online and offline more easily, thus potentially speeding up the rotation time.

3. Discussion

Effectiveness: SCIT strategy is based on the assumption that there would be vulnerabilities in the system, and attacks are unavoidable. The effectiveness of SCIT architecture cannot be analyzed in terms of prevention features. The degree of intrusion tolerance should be evaluated in the context of how much the system can limit damages caused by malicious attacks and the service availability provided to the users. For SCIT, quantitative evaluation provides guidance for parameter selection (Section 6). SCIT-based systems add architectural constructs to adapt to the particular requirements of applications. SCIT architecture for various servers, including DNS and Web Servers [3][4], and Grid Servers based on virtual server rotation can be found in [5]. With the combination of a hardware solution in SCIT/HES [2], the architecture can achieve two things: (a) "incorruptible" SCIT components to perform intrusion tolerance; and (b) increased system dependability.

Performance: A SCIT-based system does not require any extra hops or processing of an application transaction. The operations of switching a node from state to state and of self-cleansing do need extraneous processing and computing resources. But, these SCIT primitives happen "out-of-band", and do not interfere with the main data flow of the online node serving the applications. Therefore, in theory, there should be no additional latency, provided that allocation of computing resources to the application is adequate.

Integration: In order to SCIT-ize an application system, we have to put the application server under the control of the Central Controller in charge of the rotation management. Conceptually, this integration is analogous to configuring a node under a system management server in the SNMP management world. Thus, from the implementation perspective, SCITizing an application does not require changing the application software. Applications with short processing time are most suited for the SCIT approach. Applications involving in-memory storage of session information have been shown to be SCITizable; such systems include Web servers, and SSO servers.

4. Related Work

There has been work done to analyze intrusion-tolerant systems. For example, MAFTIA is a European project, while SITAR was part of the DARPA funded program called OASIS (Organically Assured and Survivable Information Systems) [8]. In [6], Stroud et al. provided a qualitative analysis of the MAFTIA architecture. Via a case study of an Internet application using MAFTIA, the authors have shown how the services of the MAFTIA middleware [10] make the system survivable under typical attack scenarios. These malicious attacks are systematically dissected by means of fault-tree analysis. The authors of [7] took the approach of using transition state diagram to study the behavior of the SITAR architecture in the face of security compromises. Quantitative analysis based on Semi-Markov Process was provided, which led to a closed form for the Mean-Time To Security Failure (MTTSF) in order to characterize an intrusion-tolerant system. . Software rejuvenation was proposed in [9] to limit impact of software faults and aging; thus cluster survivability is increased. The authors demonstrated the benefits of this approach, and analyzed the rejuvenation rate by means of stochastic process modeling. Sousa et al. [11] proposed a scheme that combines periodic system rejuvenation with a "reactive recovery". This recovery is triggered when the perceived threat goes beyond a tolerable threshold that can affect the correct working of the system.

5. SCIT Parameters and Metric

SCIT has three tunable parameters. The main one is the exposure time window W, which is the time that a node in the SCIT cluster is online to serve clients. The exposure window is composed of the online window W_o, when the server accepts transaction requests from the network, and the grace period W_g, when the server stops accepting new requests and tries to fulfill outstanding requests already in its queue:

$$W = W_o + W_g. \quad (1)$$

The second parameter is the number N_{online} of redundant online nodes to enhance the fault-tolerance of the architecture. The third one is the number N_{total} of total nodes in the cluster. Actually, N_{total}, W, and the cleansing-time $T_{cleansing}$ are inter-related. If we want to satisfy a fixed value for W, then a longer cleansing time will require more nodes to rotate. In this paper we will use MTTSF proposed in [7] to evaluate the intrusion tolerance of SCIT. Specifically, we will show how we can tune the $MTTSF_{SCIT}$ metric by means of the parameter W_o.

6. Quantitative Analysis

6.1. State Transition Diagram

For the subsequent analysis, the state transition diagram only encompasses the active period of a SCIT server, i.e. when it enters the Active state and exits the Grace Period state. Since SCIT does not contain any detection component, the state diagram of such system is much simpler than that of SITAR [7] or similar ITS, and can be modeled with four states:

- G (Good): System is functioning normally.
- V (Vulnerable): System has vulnerabilities that a hacker is trying to study.
- A (Attacked): System is under attack, as vulnerabilities are exploited.
- F (Fail): System is in failure mode. In the SCIT architecture, this Fail state has a low chance of occurring. However, in the general case, it can still happen. For instance, the attack on the active node might be such that the virtual machine and/or the host machine no longer respond to the signal of the Controller. Another possibility is the case where the Controller itself fails due to a hardware fault. Since the controller presents a single point of failure, there is no other mechanism to trigger the cleansing mode to the server under attack. In these cases, the state F is an absorbing state.

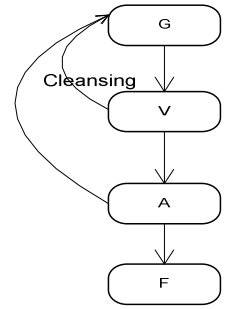

Figure 1. SCIT State Transition Diagram.

The cells of the following table give us the values for the transition probabilities from one state to the other, with P_a being the probability that an attack occurs, and P_c the probability that the live node goes to cleansing mode when the system is in Attack state. Note that for the transition probability from state G to state V, we give a value of 1, that is, we assume the worst case that the system is certain to contain exploitable vulnerabilities.

Table 1. Transition Probabilities with Absorbing State.

	G	V	A	F
G	0	1	0	0
V	$1-P_a$	0	P_a	0
A	P_c	0	0	$1-P_c$
F	0	0	0	1

6.2. Computational Steps of MTTSF

For computing the metric $MTTSF_{SCIT}$, we apply the computational steps described in [7], which is based on Semi-Markov Process model.

Step 1. From the above transition diagram, we can determine the sets X_a and X_t of absorbing states and transient states respectively as:

$$X_a = \{F\} \text{ and } X_t = \{G, V, A\}.$$

Step 2. The probability transition matrix **P** can be built from the three left columns of Table 1.

Step 3. Let **x** be the row vector composed of number of visit times x_i of the states in X_t: $\mathbf{x} = (x_0, x_1, x_2)$, where i=0,1,2 correspond to the states G, V, and A respectively. Let **q** be the row vector of probabilities q_i that the process starts at state i. Since we start with state 0 = G, **q** is the vector (1, 0, 0). Solving the system of equations

$$\mathbf{x} = \mathbf{q} + \mathbf{xP} \quad (2)$$

would yield the values:

$$x_0 = \frac{1}{P_a(1-P_c)}$$

$$x_1 = \frac{1}{P_a(1-P_c)}$$

$$x_2 = \frac{1}{(1-P_c)}.$$

Step 4. Let h be the vector composed of mean sojourn times in each state i: $\mathbf{h} = (h_0, h_1, h_2)$,
where h_0, h_1, and h_2 correspond to the states G, V, and A respectively. Then, the metric $MTTSF_{SCIT}$ can be computed as the scalar product of **x** and **h**:

$$MTTSF_{SCIT} = \mathbf{x} \cdot \mathbf{h} \qquad (3).$$

Applying the solution for **x** found above to (3), we can obtain the expression for $MTTSF_{SCIT}$:

$$MTTSF_{SCIT} = \frac{\frac{h_0 + h_1}{P_a} + h_2}{(1-P_c)} \qquad (4).$$

6.3. Exposure Window and MTTSF

Expression (4) reveals that:

- $MTTSF_{SCIT}$ is inversely proportional to the probability P_a that an attack has exploited some vulnerabilities of the system: lower values of P_a leads to higher MTTSF. So, if we want a more intrusion-tolerant system, we should find ways to decrease the value of P_a. This is in agreement with the conclusion found in [7].
- $MTTSF_{SCIT}$ is a monotone increasing function of the probability P_c that the currently live SCIT node goes to cleansing mode. The larger P_c is, so is $MTTSF_{SCIT}$. Therefore, if we want a more intrusion-tolerant system, we should find ways to increase the value of P_c. In SCIT the central controller enforces the transition to the cleansing state, thus ensuring a high P_c

Given that the exposure window W is the principal control parameter of a SCIT-based system, we would like to arrive to the conjecture that there exists a relationship between W and $MTTSF_{SCIT}$. The significance of this relationship would allow us to control $MTTSF_{SCIT}$ by means of adjusting W. For the sake of simplification, we don't consider attacks mounted in stages. Instead, we assume that attacks are independent of each other, in which case we can model attack arrival as a Poisson process. Thus, the number of attacks follows a Poisson distribution with parameter λ, which is the average number of attacks per unit time. Hence, the probability of k attacks per unit time is:

$$P(X=k) = (\lambda^k / k!)\, e^{-\lambda}$$

Let us consider the random variable Y to be the time between two successive attacks. It is well-known that Y has an exponential distribution with parameter λ:

$$f(t) = \lambda e^{-\lambda t}$$

Then, given t time units, we have the following probability: $P(Y \le t) = 1 - e^{-\lambda t}$. With t = W, this latter formula becomes: $P(Y \le W) = 1 - e^{-\lambda W}$.
This expresses the probability that the inter-arrival time between two successive attacks is less than the exposure window. Thus, the probability $P(Y \le W)$ can be considered as an upper bound for the probability P_a that there is an attack within W:

$$P_a \le 1 - e^{-\lambda W} \qquad (5).$$

We note that, $(1 - e^{-\lambda W})$ is a monotonically increasing function of W. This implies that if we make W smaller, then P_a will decrease, and $MTTSF_{SCIT}$ will increase according to (3).
The probability P_c that the live node goes to cleansing mode when it is in Attack state is an important characteristic of SCIT. We will show that P_c has a lower bound that depends on W.
Toward this end, we view the system as "serving" attacks, and the resident time of the attack modeled as a "service" time Z with rate μ. Thus, assuming an exponential distribution, we have:

$$P(Z \le z) = 1 - e^{-\mu z}.$$

Then, by our design, the probability P_c that the system moves out of state A due to the cleansing mode of the current server is more likely than the probability that the service time is greater than W:

$$P_c \ge P(Z > W) = e^{-\mu W}.$$

Since $\mu \le \lambda$ (the system cannot "serve" more than the arriving attacks), we deduce:

$$P_c \ge e^{-\lambda W} \qquad (6). \qquad \square$$

Since $e^{-\lambda W}$ is a monotonically decreasing function of W, $P(Z > W)$ increases as W decreases. This implies that if we make W smaller, then P_c will increase, and $MTTSF_{SCIT}$ will increase according to (4).
By combining (4), (5) and (6), we conclude that a smaller exposure window W for a SCIT-based system will on one hand limit the chance P_a of an attack, and on the other hand, increase the pristine state recovery likelihood P_c, thus increasing the system's tolerance and

$$MTTSF_{SCIT} \ge F(W)$$

where

$$F(W) = \frac{\frac{h_0 + h_1}{(1-e^{-\lambda W})} + h_2}{(1-e^{-\lambda W})} \qquad (7),$$

is a decreasing function of W.
Summary. For $MTTSF_{SCIT}$, we obtained expression (4) in terms of P_a and P_c. Next, we have shown via (5) and (6) that P_a and P_c can be controlled by the exposure window W. t Then, we were derived (7) which expresses that $MTTSF_{SCIT}$ depends on W. This conclusion is significant, because it means that we can engineer an instance of SCIT architecture by tuning the

window W in order to increase or decrease the value of $MTTSF_{SCIT}$.

6.4. Grace Period Control

From (1), we know that the exposure window has two components, namely the online time W_o and the grace period time W_g. It is precisely during the window W_o that data traffic and transaction requests will be accepted into the server, so that all of the potential attacks can arrive only during that period. Note that the Central Controller is responsible for sending a signal to the online server node at the end of its grace period to go into cleansing mode. The communication between the Controller and a server node is one-way from Controller to nodes in order to protect the Controller from any security compromises coming from the server nodes. The Controller signals the transition of the server to the next state. The following discussion guides the grace period W_g estimation.

Let N be the average number of outstanding requests in the queue when the server enters the grace period. As in the previous section, we can also model the incoming traffic and transaction requests by using a Poisson process with parameter α. It is well-known that the attack traffic is a subset of the total traffic, and there exists a probabilistic relation between the two:

$$\lambda = p.\alpha, \text{ with } p \leq 1.$$

In turn, the average number of outstanding requests cannot exceed the average number of arrivals during the online window:

$$N \leq \alpha W_o. \qquad (8)$$

Let S be the service rate in terms of number of serviced requests per unit time. Then:

$$W_g = N/S \leq (\alpha /S) W_o.$$

If we assume that $(\alpha/S) < 1$, then clearly W_o represents an upper bound for

$$W_g: W_g < W_o \qquad (9)$$

This assumption $(\alpha/S) < 1$ is not unrealistic, since one always wants to engineer and design the server with an adequate service rate to serve the rate of incoming traffic. If this is not true, then the server will be overloaded. The result in (10) provides the SCIT Controller a computed estimate of the grace period allocated to the server nodes under its management.

By combining (1), (7) and (10), we can arrive at the lower bound of $MTTSF_{SCIT}$ that depends solely on the online window W_o:

$$MTTSF_{SCIT} \geq F(2W_o) \qquad (10).$$

Figure 2 depicts the trend of the lower bound $lb(MTTSF_{SCIT})$ in terms of W_o with example values of $h_i = (1/3)$ and $\lambda = 1$ (x), 2 (□), 3(○).

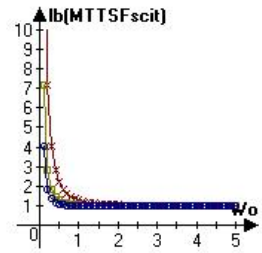

Figure 2. $MTTSF_{SCIT}$ trend as a function of W_o.

Now, that we know how to control $MTTSF_{SCIT}$ by W_o, we want to find if there exists any constraint on W_o. Applying Little's Law to (9), we can write:

$$\alpha /S \leq \alpha W_o. \text{ Hence: } (1/S) \leq W_o \qquad (11).$$

This constraint means that the online window has to be larger than the average service time of a request. Since we want to make W_o small to increase the effectiveness of the intrusion tolerance, (11) indicates that the SCIT architecture is more effective for short transactions.

6.5. Failure State in SCIT

During the analysis of $MTTSF_{SCIT}$, we consider state F (Failure) as an absorbing state. Actually, the SCIT architecture periodically activates a live spare server, and brings down a server being in grace period. Moreover, the chance that the intruder takes over the Controller is very minimal, due to the one-way data path as described in earlier section. In this context, one can argue that the cluster's state F is not really absorbing, since the system automatically recovers back to the G state. Thus, we can use Semi-Markov Process with embedded DTMC (Discrete-Time Markov Chain) to compute the steady-state *Availability* (state without security faults) of a SCIT system as outlined in [7]. The transition matrix **Q** used for this computation will get the values from Table 2, where the last row for F will have the values (1, 0, 0, 0) since F can go to G with probability P_c.

Table 2. Transition Probabilities for DTMC

	G	V	A	F
G	0	1	0	0
V	$1-P_a$	0	P_a	0
A	P_c	0	0	$1-P_c$
F	1	0	0	0

After solving the DTMC steady-state probabilities for all states denoted by $\mathbf{y} = (y_0, y_1, y_2, y_3)$, we can derive the SMP stead-state probability π_F for state F: $\pi_F = y_3h_3/\mathbf{y}.\mathbf{h}$, with $\mathbf{h} = (h_0, h_1, h_2, h_3)$ being extended to include the mean sojourn time h_3 for state F.

From *Availability* = $1 - \pi_F$, we arrive at the expression:

$$Availability = \frac{h_0 + h_1 + P_a h_2}{h_0 + h_1 + P_a h_2 + P_a(1 - P_c)h_3} \quad (12).$$

By considering the expression for *Availability* as a function of P_a and P_c, and using partial differentiation, we find that *Availability* monotonically decreases with P_a but increases with P_c. Using the same line of reasoning and the assumption of Poisson attack arrival process as for $MTTSF_{SCIT}$ above, we can also conclude that decreasing the exposure window will increase the system availability.

7. Conclusion

The SCIT approach has been discussed from the perspectives of effectiveness, tunable parameters, performance impact, and integration to application systems. From the derived expression for $MTTSF_{SCIT}$, we were able to conjecture mathematically that decreasing the exposure time window will improve the intrusion tolerance of a SCIT-based system. The result is quite powerful in the sense that it offers to system engineers the capability to utilize SCIT exposure window to tune the system to the required level of intrusion tolerance specified by MTTSF, just as such tuning has been possible with the availability of a fault-tolerant system.

To increase $MTTSF_{SCIT}$ would require shrinking the exposure window; hence the cycle that a SCIT server has to go through will become shorter. Therefore, we would like to evaluate the impact and cost trade-off between the $MTTSF_{SCIT}$ tuning and the computing resource utilization in a future work. Furthermore, we plan want to study more complex attack models and their distribution functions.

8. Acknowledgement

Dr. Sood's research is partially supported by a research grant made to George Mason University by Lockheed Martin Corporation. The authors are grateful to the anonymous reviewers for their comments, which have contributed to improvements in the final version of the paper.

9. References

[1] Yih Huang, David Arsenault, and Arun Sood. "Incorruptible System Self-Cleansing for Intrusion Tolerance". *Performance, Computing, and Communications Conference, IPCCC 2006*.

[2] Yih Huang, David Arsenault, and Arun Sood. *Secure,* "Resilient Computing Clusters: Self-Cleansing Intrusion Tolerance with Hardware Enforced Security (SCIT/HES)". *The Second International Conference on Availability, Reliability, and Security, ARES 2007*.

[3] Yih Huang, David Arsenault, and Arun Sood. "Securing DNS services through system self cleansing and hardware enhancements". *The First International Conference on Availability, Reliability, and Security, ARES 2006*.

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

[5] Matthew Smith, Christian Schridde and Bernd Freisleben. "Securing Stateful Grid Servers through Virtual Server Rotation". *HPDC'08,* June 23–27, 2008, Boston, Massachusetts, USA.

[6] Robert Stroud, Ian Welch1, John Warne, and Peter Ryan. "A Qualitative Analysis of the Intrusion-Tolerance Capabilities of the MAFTIA Architecture". *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, 2004.

[7] Bharat B. Madan, Katerina Goseva-Popstojanova, Kalyanaraman Vaidyanathan, and Kishor S. Trivedi. "A Method for Modeling and Quantifying the Security Attributes of Intrusion Tolerant Systems". *Dependable systems and networks-performance and dependability symposium (DSN-PDS) 2002*.

[8] *Organically assured and survivable information systems*. http://www.tolerantsystems.org

[9] Khin Mi Mi Aung, Kiejin Park and Jong Sou Park. "A Rejuvenation Methodology of Cluster Recovery". *CCGrid 2005, IEEE International* Symposium Vol. 1, pp. 90 - 95, May 2005.

[10] Paulo E. Veríssimo, Nuno F. Neves, Christian Cachin, Jonathan Poritz, David Powell and Yves Deswarte, Robert Stroud, and Ian Welch. "Intrusion-Tolerant Middleware: The Road to Automatic Security". *IEEE Security & Privacy*, 2006.

[11] Paulo Sousa, Alysson Neves Bessani, Miguel Correia, Nuno Ferreira Neves, Paulo Verissimo. "Resilient Intrusion Tolerance through Proactive and Reactive Recovery". *13th IEEE International Symposium on Pacific Rim Dependable Computing*, 2007.