Attack-Potential-Based Survivability Modeling for High-Consequence Systems*

J. McDermott
Center for High Assurance Computer Systems
Naval Research Laboratory
Washington, DC 20375, USA
John.McDermott@NRL.Navy.mil

24 March 2005

Abstract

Previous quantitative models of security or survivability have been defined on a range of probable intruder behavior. This measures survivability as a statistic such as mean time to breach. This kind of purely stochastic quantification is not suitable for high-consequence systems. For high-consequence systems the quantified survivability should be based on the most competent intruders the system is likely to face. We show how to accomplish this with a contingency analysis based on variations in intruder attack-potential. The quantitative results are then organized and presented according to intruder attack potential. Examples of the technique are presented using stochastic process algebra. An interesting result for diverse replication is included in the examples.

1 Introduction

Recently, intrusion-tolerance has been seen as a potential approach to increasing the survivability of an information system. An important part of this approach has been to quantitatively model intrusion-tolerance or survivability. The trend in these models has been to move away from characterizing security or survivability as simply a small collection of statistics, such as mean time to security breach or mean effort to security breach. More recent approaches have included detailed models of intruder behavior including relatively complex interaction between the intruder and the system under attack. However, the end result is still presented as one or more purely stochastic properties of the system being modeled.

*Proc. Third IEEE Int. Information Assurance Workshop, March 2005, Washington, DC, USA, J. Cole and S. Wolthusen eds., with typographical corrections.

This approach and the fundamental concept of a purely stochastic model of security or survivability can be problematic for systems that manage high-value resources or support critical missions. Most security breaches in low- or moderate-consequence systems are due to the presence of known and corrected but unpatched vulnerabilities or to unintentional misconfiguration. For these kinds of systems, a statistic such as mean effort to breach, defined on a range of probable intruder behavior, is more descriptive than a Common Criteria evaluation rating. However, high-consequence systems are subjected to sufficient engineering and operational management to practically rule out unpatched vulnerabilities or misconfigurations. Security breaches in these systems result from human ingenuity. A subtle design flaw is exploited or a previously undiscovered implementation flaw is located by the intruder's engineering efforts. The faults leading to these breaches are not stochastic.

In systems that manage high-value resources or support critical missions, we are not interested in quantifying over a wide range of intruder ability. Instead, we want to quantify protection of those resources and missions in the presence of the most competent intruders the system is likely to face.

Realistic quantitative models for high-consequence systems need to reflect detailed non-stochastic intruder behavior. Quantitative survivability (and security) should be modeled and measured with respect to the behavior of specific reasonably competent intruders.

This paper presents quantitative survivability modeling for high-consequence systems in three steps. First we draw a distinction between stochastic fault models and fault models for human-sponsored faults. Then we explain how to characterize human-sponsored faults in terms of attack potential and show why hard intruders are applicable to high-consequence systems. Finally, we construct some example attack-potential-based survivability models. These models present an interesting result concerning diverse replication for intrusion tolerance. Readers who are familiar with basic survivability definitions and stochastic process algebra may prefer to skip the rest of this introduction.

1.1 Related Work

Early work in quantitative modeling of security was done by Littlewood et al. [14] identifying intruder work factor as a fundamental quantitative measure of security. Jonsson and Olovsson [10] constructed a quantitative intruder model using real data. As we discuss in this paper, their model was not for a high-consequence system. Ortalo, Deswarte, and Kaâniche modeled operational security in a real system [15]. Their work also was not for a high-consequence system but does provide a concrete example of the kind of quantitative modeling we advocate for high-consequence systems. Their model has two distinct intruders: a stateful and a stateless intruder. In planning future attacks in a network, a stateless intruder only considers attacks that are possible from the host it has just compromised. A stateful intruder considers attacks from hosts it has previously compromised. Their quantitative results are presented for both kinds of intruders. Neither of the two intruders represents a hard intruder.

Singh, Cukier, and Sanders [21]; and Stevens et al.[23] quantitatively model architectures that could be candidates for high-consequence systems. They also present the use of quantitative models for validation and design studies during system construction rather than operational assessment of an existing system. The models they present are not attack-potential-based but could be adapted to attack-potential-based modeling.

The stochastic process algebra we use was created by Hillston [9]. As far as we know, our work is the first use of SPA to model survivability.

1.2 Definitions and Concepts

We begin with a definition of *high-consequence system*. A high-consequence system includes resources or missions that could cause serious harm to human beings, if there is a failure. Serious harm includes not only physical injury but other kinds of harm including loss of political freedom or financial well-being. Examples of high-consequence systems include not only national security systems but also medical and real-time control systems. Finally, any large-scale information system that supports large numbers of human beings (e.g. systems that support elections) is also high-consequence because of the difficulty of making good after a failure.

Given that definition, we lay out some general dependability concepts from Avižienis, Laprie, and Randell [2, 13] which we quote:

- *Dependability* of a computing system is the ability to deliver service that can justifiably be trusted.
- *Failure* is an event that occurs when the delivered service deviates from correct service.
- An *error* is that part of the system state that may cause a subsequent failure.
- A *fault* is the adjudged or hypothesized cause of an error.

It is important to point out that we do not use the definition of security given in the preceding references. The distinction we draw is that *security is dependability in the face of human-sponsored attacks.* We use the term *security breach* as a synonym for failure.

Powell, Stroud, et al.[17] provide an insightful interpretation of general dependability concepts for security. We follow their definition, bearing in mind our qualification of human sponsorship:

- An *attack* is a malicious interaction fault aiming to intentionally violate one or more security properties; an intrusion attempt via a vulnerability.
- A *vulnerability* is an accidental fault or non-malicious intentional [1] fault, in

[1] Non-malicious intentional faults are misconfigurations introduced for reasons such as performance or usability.

the requirements, specification, design, implementation, or configuration of the system or its use, that could be exploited to create an intrusion.

- An *intrusion* is a malicious, externally-induced fault resulting from a successful attack.

We notice that an intrusion always causes a security breach while an attack or a vulnerability only has the potential to cause a security breach. We do not attribute human sponsorship to vulnerabilities. A maliciously introduced fault (i.e. flaw) in requirements, specification, design, implementation, or configuration is really an attack.

Following conventional security practice, we qualify attack, vulnerability, or intrusion with a general security property that may be violated: e.g. confidentiality, integrity, or availability. For example, we may have a confidentiality attack or an availability intrusion.

1.3 Stochastic Process Algebra

This paper presents some example survivability models based on stochastic process algebra (SPA). PEPA (Performance Evaluation Process Algebra) [8, 5, 6] is a stochastic process algebra that has been successfully applied to a range of problems. Stochastic process algebras add a performance or quantitative element to the behavior modeling of process algebras.

Activity prefix is the most fundamental PEPA construct. Prefixing allows us to prefix activities to a process to describe its behavior. So the process described as $(\alpha, r_\alpha).(\beta, r_\beta).P$ engages in activity α, then does activity β, and finally acts like process P. Each activity (α, r_α) has a duration that is exponentially distributed with mean $1/r_\alpha$. An activity rate can be any positive real number or the special value $\top$ that indicates a *don't care* rate, i.e. $\top$ is so fast that other activities always determine the rate. Choice between processes represents choices of behavior. Choice between processes P and Q is denoted $(P + Q)$; the combined process will either act like process P or process Q. If process P is chosen this reflects the fact that the activities of process P completed before process Q.

The rate of an activity can be used to simulate probabilistic choice. For example, suppose we want to simulate a probabilistic choice between processes P_1 and P_2. We define an activity (α, r) and prefix it to both alternatives, with a modified rate $p(Q) \cdot r$ that reflects the probability $p(Q)$ that we wish to assign to each alternative Q. Setting $p(P_1) = 1/3$ and $p(P_2) = 2/3$ as the probabilities we get

$$(\alpha, r/3).P_1 + (\alpha, 2r/3).P_2 \tag{1}$$

that acts like process P_1 one-third of the time and acts like process P_2 two-thirds of the time. We use this technique to model intruder and defender choices.

Since modeling interaction between processes is the key motivation for process algebra, PEPA has a cooperation operator $\bowtie$ for interprocess communi-

cation. A cooperating process $(P \underset{L}{\bowtie} Q)$ synchronizes on the activities in the *cooperation set* L. The activities in the cooperation set will occur together in processes P and Q, with the duration of the slower activity.

A PEPA process algebra model has a corresponding Markov process. This Markov process can be solved to obtain the steady-state distribution of the states of the Markov model. When we describe a PEPA model as being in a certain state, we mean that the corresponding Markov process is in that state.

2 Fault Models

A *stochastic fault* is a fault whose occurrence or non-occurrence is predicted by one or more random variables. It is not possible to show the occurrence or non-occurrence of a stochastic fault by a logical argument based on the design of the component. That is, we cannot apply *fault prevention*. What we can do with a stochastic fault is apply the laws of mathematical probability to predict its likelihood. It is possible to show, using a logical argument, the *consequence* of a stochastic fault given its occurrence, if the type of fault, the design, and the state of the faulty component are known. Stochastic faults can occur when no human intruder is present; in fact this is the usual paradigm.

Stochastic faults can also be used to accurately model the behavior of naive attackers that do not understand the design of the defensive mechanisms [10]. Stochastic faults are useful for modeling the behavior of a range of probable intruders operating against a low- to moderate-consequence system. Stochastic faults are also accurate for modeling physical damage to information systems though not necessarily all forms of physical attack. Multiple stochastic faults are usually statistically independent. If there are dependencies between faults then the dependencies can be described by threshold assumptions.

Jonsson and Olovsson constructed a stochastic model of the security intrusion process [10]. This model characterizes intruders in terms of their stationary *mean time to breach* (MTTB). As we will discuss in detail in Section 3, their model is very accurate for the low-consequence system they investigate, because they carefully define the knowledge, access, and capabilities of the intruders they model. This is critical. However, the system-relevant attack behavior of the intruder is not reflected in the model. All system-relevant behavior is abstracted into phases of intruder skill. That is, their model does not relate its statistic to the security architecture and protocols under attack. Also, their model gives no definition of security per se. That is, there is no clarification of what security policy or claim is being violated by an intrusion. The resulting intruder is purely stochastic.

Thus MTTB quantification does not tell us if our system is safe from denial of service, information leaks, or something else. A purely stochastic fault model does not represent details of the intrusion itself and so tells us little about the merits and demerits of specific design choices. As Singh, Cukier, and Sanders state [21]

> We believe that probabilistic models for intrusion-tolerant systems should, either explicitly or logically, include sub-models of the attacker, the intrusion-tolerance mechanism being used, the application, and the resource/privilege state of the system.

This brings us to the idea of a *sponsored fault*. A sponsored fault is caused by a human intruder using design to exploit vulnerabilities in a system. If the initial conditions and the design of a component are known then the occurrence or non-occurrence of a particular sponsored fault can be shown by logical argument. Sponsored faults frequently comprise a set of vulnerabilities organized according a design [20, 16, 22, 25].

3 Characterizing Sponsored Faults

The literature regarding stochastic faults has well-understood ways of characterizing these faults [3]. Stochastic faults are described in terms of the kinds of failures they cause and the frequency at which they occur, e.g. no more than t simultaneous faults during the life of the system. In contrast to this, sponsored faults are best described in terms of the intruders that sponsor them.

Schudel and Wood [20] analyzed the qualitative use of a work factor as a means of characterizing sponsored faults. Intruder work factor (e.g. mean time to accomplish an attack) is part of a good metric for survivability or security. However, intruder work factor is determined by the *attack potential* of an intruder. A work factor metric should be coupled with a description of intruder attack potential that determines it. Intruders are best characterized by directly defining their attack potential. Work factor or MTTB can then be determined by modeling or experiment against the system of interest.

3.1 Attack Potential

Attack potential is an intruder's potential for succeeding in a specific kind of penetration attempt or attack. Attack potential is based on the intruder's *initial access*, *initial knowledge* and *capabilities*. Jonsson and Olovsson very carefully measured, documented, and reported all of these attributes in their stochastic modeling of security. Their description was also closely tied to the specific system under attack. Many subsequent quantitative studies have not done this as carefully. Because of this, their results are difficult to apply to sponsored faults because they are based on a poorly defined notion of attack potential. In other cases, results have been defined in terms of outcomes or symptoms to be avoided, as in denial of service, or loss of confidentiality, assuming a stochastic intruder. The limitation of outcome- or symptom-based results is that, given a suitably naive or weak intruder, any solution may avoid the symptom.

Finally, even some of the best work only aims to stop *probable intruders* rather than *hard intruders*. Probable intruders are the most frequently seen but not the hardest to defeat. Hard intruders are in the far right side of the attack potential distribution; they are fully competent in the kind of attack

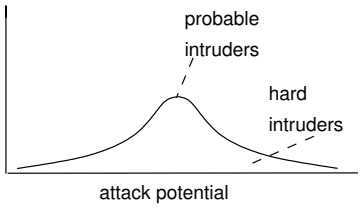

Figure 1: Distribution of Intruder Attack Potential

under consideration. Figure 1 shows the distinction between hard and probable intruder attack potential. Figure 1 also shows the only probabilistic aspect of sponsored faults: the distribution of intruder attack potential. All other aspects are matters of either capability, initial access, or initial knowledge. Schneider and Zhou state[19]

> Fault-tolerance and attack-tolerance are ultimately tied to a set of assumptions about the environment in which a system must function. Weaker assumptions should be preferred, since then there is less risk that they will be violated by natural events or an adversary's attacks.

This identifies the fundamental value of the hard intruder for high-consequence systems. A weak or probable intruder represents a strong assumption about the environment; the harder the intruder, the less we assume about the environment.

The cryptographic protocol research community has accepted the wisdom of a well-defined hard intruder in any research that seeks to deal with human adversaries. They consistently use well-defined intruders (e.g. the Dolev-Yao model[4]) in their work. A well-defined intruder has a clearly specified set of capabilities, initial access, and initial knowledge. This in turn clarifies the effectiveness and applicability of the results.

The interest in defining and using hard intruders arises out of both mathematical considerations and practical needs. First, comparison and measurement require some kind of upper bounds. It is especially important to know if a proposed solution will not handle hard intruders. Second, since intruders are human they not only learn but also share knowledge. Thus the hard but unlikely intruder of today becomes the norm or most probable intruder of tomorrow. In other words, the distribution of intruder attack potential shown in Figure 1 moves to the right over time, as shown in Figure 2.

3.2 A Hard Intruder for the Server Denial of Service Problem

At this point an example will help to clarify what we mean by hard intruder. Let us consider a hard intruder for a familiar problem: server survivability, where the goal is to provide some degree of resistance to denial of service [16, 23, 21].

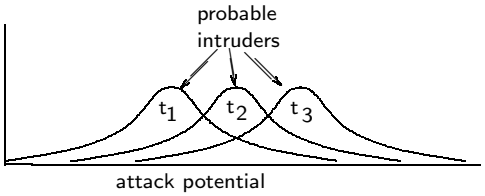

Figure 2: Attack Potential Increasing Over Time

For this problem, a hard intruder might be described as one with the following *initial access*

- Root (i.e full administrative) access to the host of at least one authorized client of the server;

the following *initial knowledge*

- Complete knowledge of the implementation (source code, compiler options, etc.) and documentation for the client;
- Complete knowledge of the implementation (source code, compiler options, etc.) and documentation for the server, for each different kind of server used in a diverse replication approach;
- complete knowledge of the implementation of any survivability mechanisms including those used for recovery or reconfiguration;
- complete knowledge of the implementation of any intrusion detection systems in use;
- sound statistical knowledge of current configuration data, including intrusion detection rule bases;

and the following *capabilities*

- full application of the best available software engineering technology, including security vulnerability analysis and formal methods;
- detection avoidance via use of a representative copy of the system to be attacked, for learning, planning, and rehearsal (i.e. none of these activities are conducted on the target system) [20];
- use of a strategy of *persistence*: when a vulnerability is found that can compromise a server, this vulnerability will be used repeatedly, on the same replica, until a new version of the software without the flaw is installed or the server is permanently disconnected from its network;
- use of a strategy of *conservation of vulnerabilities*: if replicated diversity is used in the defense then the intruder will postpone any attack until a vulnerability is available for each kind of replica; that is, attacks will use replicated diversity as well.

This particular intruder definition includes reasonable attack potential that can critically impact the expected survivability of a system. That is, each capability, known fact, or access cannot be shown to be either computationally infeasible, humanly unlikely, or prohibitively expensive for the intruder. This particular intruder is arguably more justifiable than the Dolev-Yao intruder but still poses significant challenges to current survivability mechanisms. The capabilities we have discussed here do not include all of the denial-of-service-relevant capabilities that could be justified [22, 11, 25, 18, 12, 24].

Apparently simple changes in intruder initial access, initial knowledge, or capabilities can have significant impact on the claimed effectiveness of a solution. For example, as the models in our next section will show, if we remove the strategy (i.e. capability) of vulnerability conservation from our intruder then server survivability increases significantly. But there is no reason to claim that a competent intruder would not conserve flaws and use a diverse attack. So it is important to know precisely what kind of intruder is being supposed, that is, the intruder's specific attack potential.

3.3 Attack-Potential-Based Modeling With Hard Intruders

Our response to the significance of specific attack potential is to present quantitative results for different kinds of intruders. This is a kind of contingency analysis (as opposed to sensitivity analysis); we vary our assumptions about the intruder to investigate the significance of each assumption. The critical point, from the perspective of the designer or evaluator of a survivable system, is to make this contingency analysis a part of the quantitative results. A designer or an evaluator can then understand how closely the quantitative results apply to a proposed system and its environment.

We must be careful to conduct and present our contingency analysis in terms of hard intruder attack potential. To do this, we must vary the structure of the models to reflect 1) details of both the system artifacts that participate in an attack and 2) details of the intruder's attack potential. Otherwise, it is possible to merely vary the rate at which the intruder succeeds. This would bring us back to a purely stochastic fault model that gives little insight into the merits and demerits of specific design choices. Useful quantitative results should be presented as a description and justification of each specific intruder, a description of each model variant, and quantitative results for each intruder.

4 Modeling Survivability Using Different Intruders

Now we present a series of models that provide an example of attack-potential-based quantitative modeling of survivability for high-consequence systems. Our examples also demonstrate the significance of getting the intruder model right.

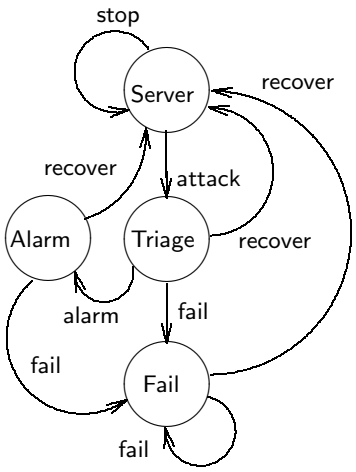

Figure 3: Activities of the Sponsored Fault Model

Our examples model survivability in the presence of denial of service attacks against a server. Each model measures the fraction of a mission that is supported by the server. We assume the mission has a short duration, say 100 hours, and that no changes can be made to the executable software during the mission. Our models show how long the server is up and working during the mission, that is, its availability in the presence of sponsored faults, which approximates survivability.

There are four attack-potential-based models: 1) a naive intruder attacking a single server, 2) an intruder with the capability of persistence, attacking a single server, 3) the same intruder attacking a diversely replicated server, and 4) a flaw-conserving intruder that makes diversely replicated attacks against a diversely replicated server.

Our examples reflect increasing levels of attack potential. The basic abstraction of the server and its interaction with the attacker is shown in Figure 3. This model is a simplification of the state model of Goseva-Popstojanova et al. [7]. Normal operation is indicated as *Server*. An *attack* event puts the server in a *Triage* condition where it automatically tries to identify and handle the attack. A *stop* event represents failure of the intruder's attack; the model stays in the Server condition. While in a triage condition the server may either recover, fail, or go into an *Alarm* condition. In an alarm condition the user is either able to manually recover the server or it fails. All failure events result in a hard failure for the server; it is in a *Fail* condition. The possibility of recovery still exists but it is much lower than from the alarm or triage conditions.

$$
\begin{aligned}
Intruder &\stackrel{def}{=} (searching, h).Attack;\\
Attack &\stackrel{def}{=} (start\_attack, k).(attack, a).Attack + (start\_attack, el).Intruder;\\
Server &\stackrel{def}{=} (attack, \top).Triage;\\
Triage &\stackrel{def}{=} (start\_triage, rt).(recover, r_1).Server + (start\_triage, at).(alarm, b).Alarm +\\
&\qquad (start\_triage, ft).Fail;\\
Alarm &\stackrel{def}{=} (start\_alarm, ra).(recover, r_2).Server + (start\_alarm, fa).Fail;\\
Fail &\stackrel{def}{=} (start\_fail, rf).(recover, r_3).Server + (start\_fail, ff).(fail, f).Fail;\\
&Intruder \underset{\{attack\}}{\bowtie} Server
\end{aligned}
$$

Figure 4: PEPA Model of a Single Server and Naive Attacker

4.1 Attack on a Single Server

The first process algebra model represents a naive attacker trying to shut down a single server. It is shown as Figure 4 below. The attacker is modeled by the PEPA process *Intruder*; process *Server* models the single server that is under attack. The process shown as Equation 2 below, uses the PEPA cooperation operator $\underset{\{attack\}}{\bowtie}$ to represent the interaction of the intruder and the server via synchronized participation in the event *attack*.

$$Intruder \underset{\{attack\}}{\bowtie} Server \tag{2}$$

This model has 27 states, where a state combines the current process of both the intruder and the server. The applicable activity rates for this model are shown in Table 1.

The results for the naive attacker against a single server are that the server is down for less that 6% of the mission. About a third of this 6% is spent in the process

$$(attack, a).Attack \underset{\{attack\}}{\bowtie} (fail, f).Fail$$

the rest is in one of the processes that represents recovery from failure. This is reasonable given the short mission, the intruder's lack of persistence, and the given likelihood of finding security flaws. It serves as a reasonableness check on the basic structure and rates of our models.

If we now assume a more sophisticated intruder who adopts a strategy of *persistence* then the survivability of the same server does not look as good. A

Rate	Value	Meaning
k	0.6000	Intruder persistence.
el	0.4000	Intruder's rate of giving up or fumbling the attack.
rt	0.2500	Likelihood of automatically recovering from triage.
ra	0.9000	Likelihood of manually recovering from alarm.
rf	0.9000	Likelihood of manually recovering after hard failure.
at	0.7000	Likelihood triage can't stop attack and goes to alarm.
fa	0.1000	Likelihood of being unable to recover after an alarm.
ft	0.0500	Likelihood of hard failure from triage.
ff	0.1000	Likelihood of remaining hard failed.
a	100.0	Duration of an attack.
h	0.00017	Rate at which the intruder discovers flaws.
m	0.99983	$1 - h$
r_1	0.1428	The duration of an automatic recovery.
r_2	0.2856	The duration of a manual recovery from alarm.
r_3	0.001428	The duration of a manual recovery from hard failure.
b	1.000	The duration of an alarm.
f	0.0001	The duration of a failure

Table 1: Transition Rates for Naive Attacker

persistent attacker will exploit the fact that no software fix is available and repeat any successful attack, for the duration of the mission. This repetition is not a repeat of the same kind of attack against another server, but repeatedly attacking the same server, as it is in the process of trying to cope with a previous attack. We model this by changing the rates k and el to $k = 0.999$ and $el = 0.001$. Notice that this is not a purely stochastic change in the previous model. The work factor, expressed as the rate h, has not changed.

So our new model has the same PEPA stochastic process algebra as before, with 27 states in the resulting Markov model. With the change in intruder capability, the single server is now down for about 97% of the mission. About 58% of its time is spent in the process

$$(attack, a).Attack \underset{\{attack\}}{\bowtie} (fail, f).Fail$$

that is, in hard failure under attack.

4.2 Attack on a Diversely Replicated Server

In response to the higher attack potential of the persistent intruder, we can use redundancy based on heterogeneous servers. Our diversely replicated server uses three distinct servers operating as a fault-tolerant or survivable state-machine ensemble. We count the server as still supporting the mission as long as one replica is still in the *Server* process. (Most agreement protocols would call for more replicas, but the added states can exceed the capabilities of our current modeling tool.)

$$Intruder_0 \stackrel{def}{=} (searching, h).Attack_0;$$
$$Attack_0 \stackrel{def}{=} (start\_attack, k).(attack_0, a).Intruder_1 + (start\_attack, el).Intruder_0;$$
$$Intruder_1 \stackrel{def}{=} (searching, h).Attack_1;$$
$$Attack_1 \stackrel{def}{=} (start\_attack, k).(attack_1, a).Intruder_2 + (start\_attack, el).Intruder_1;$$
$$Intruder_2 \stackrel{def}{=} (searching, h).Attack_2;$$
$$Attack_2 \stackrel{def}{=} (start\_attack, k).(attack_2, a).Attack_2 + (start\_attack, el).Intruder_0;$$

Figure 5: PEPA Model of Sequential Intruder for Replicated Server

We need a new stochastic process algebra model that retains the rates of the single-server persistent intruder model, but structurally reflects the attacker's interaction with the server. This model is shown in Figure 5

The processes $Intruder_0$ through $Intruder_2$ represent stages of the attack the intruder must make. Again the activity rates are as shown in Table 1 except rates $k = 0.999$ and $el = 0.001$, to model persistence. The intruder alternates between searching for a vulnerability and attacking. As a vulnerability is discovered, immediately it is used to attack the applicable server. This model assumes no common mode vulnerabilities; the intruder must find three distinct vulnerabilities to stop the replicated server.

We model the replicated server as a set of three PEPA processes $Server_0$ through $Server_2$, as shown in Figure 6 The model does not reflect the normal operation of the servers, but the sequential nature of the attack: the first server to be compromised is $Server_0$ and the last server to be compromised is $Server_2$. Intruder activities in process $Server_2$ cannot take place before activities in the previous server processes.

The fact that each server process is designed to cooperate on any of the attacks is an artifact of the model that is needed to prevent deadlocks. Attacks not intended for the particular server have no effect. For example, for $Server_0$ the component corresponding to an $attack_1$ is $(attack_1, \top).Server_0$ which causes the attack to fail; that is, an $attack_1$ event has no effect the $Server_0$ process.

Process $Server_1$ begins activity via the process $(start\_fail, ff).(fail, f).Server_1$ that is a component of process $Alarm_0$. Then the attacker begins to search for a vulnerability in the second server, modeled as the interaction between process $Intruder_1$ and process $Server_1$. The higher numbered server processes $Server_1$ and $Server_2$ can revert back to process $Server_0$ through recovery from their respective $Alarm$ and $Triage$ processes. That is, unlike the single server process model, a recovery does not recursively return to the corresponding server process, but goes back to next lower level of failure, thus allowing a chain of recovery back to complete restoration of all three replicas.

The intruder and the replicated server are combined using the PEPA coop-

$$
\begin{aligned}
Server_0 &\stackrel{def}{=} (attack_0, \top).Triage_0 + (attack_1, \top).Server_0 + (attack_2, \top).Server_0;\\
Triage_0 &\stackrel{def}{=} (start\_triage, rt).(recover, r_1).Server_0 + (start\_triage, at).(alarm, b).Alarm_0 +\\
&\qquad (start\_triage, ft).Fail_0;\\
Alarm_0 &\stackrel{def}{=} (start\_alarm, ra).(recover, r_2).Server_0 + (start\_alarm, fa).Fail_0;\\
Fail_0 &\stackrel{def}{=} (start\_fail, rf).(recover, r_3).Server_0 + (start\_fail, ff).(fail, f).Server_1;\\
Server_1 &\stackrel{def}{=} (attack_0, \top).Server_1 + (attack_1, \top).Triage_1 + (attack_2, \top).Server_1;\\
Triage_1 &\stackrel{def}{=} (start\_triage, rt).(recover, r_1).Fail_0 + (start\_triage, at).(alarm, b).Alarm_1 +\\
&\qquad (start\_triage, ft).Fail_1;\\
Alarm_1 &\stackrel{def}{=} (start\_alarm, ra).(recover, r_2).Fail_0 + (start\_alarm, fa).Fail_1;\\
Fail_1 &\stackrel{def}{=} (start\_fail, rf).(recover, r_3).Server_1 + (start\_fail, ff).(fail, f).Server_2;\\
Server_2 &\stackrel{def}{=} (attack_0, \top).Server_2 + (attack_1, \top).Server_2 + (attack_2, \top).Triage_2;\\
Triage_2 &\stackrel{def}{=} (start\_triage, rt).(recover, r_1).Fail_1 + (start\_triage, at).(alarm, b).Alarm_2 +\\
&\qquad (start\_triage, ft).Fail_2;\\
Alarm_2 &\stackrel{def}{=} (start\_alarm, ra).(recover, r_2).Fail_1 + (start\_alarm, fa).Fail_2;\\
Fail_2 &\stackrel{def}{=} (start\_fail, rf).(recover, r_3).Server_2 + (start\_fail, ff).(fail, f).Fail_2;
\end{aligned}
$$

Figure 6: PEPA Model for Replicated Server Attacked by Sequential Intruder

eration operator $\bowtie$ that causes its operands to synchronize on activities in the shared interface $\overset{S}{S}$, as shown in Equation 3.

$$Intruder_0 \underset{\{attack_0, attack1, attack2\}}{\bowtie} Server_0 \tag{3}$$

The resulting PEPA model has 157 states that reflect all of the possible combinations of vulnerabilities, attack progress, and defensive response possible for our replicated server.

This model shows the benefit of the diverse replication that forces an intruder to compromise different software on each server replica. For the same activity rates as the single server versus a persistent attacker, this replicated server is available for over 93% of the mission. The replicated server spends over 60% of its time in one of the two processes

$$Intruder1 \underset{\{attack_0, attack_1, attack_2\}}{\bowtie} Server0$$

$$Intruder2 \underset{\{attack_0, attack_1, attack_2\}}{\bowtie} Server0$$

That is, while the persistent intruder is searching for a vulnerability to carry out the next stage of its attack, the replicated servers are recovering completely back to full operation.

Stevens, et al. characterize the preceding approach (use a vulnerability as soon as it is discovered) as a aggressive strategy [23]. This aggressive strategy may be stressful to the replicated server but it is not always best from the intruder's perspective.

What would the survivability of the same replicated server be, if the intruder did not attack until a vulnerability was known for each diverse server replica? That is, what if the intruder conserves vulnerabilities and uses a replicated attack? This alternative strategy does not violate the diverse server assumption of failure independence directly. However, it simulates it, and thus poses a challenge to diverse replication as a defense. However, improved intruder effectiveness is not guaranteed: since it takes longer to find multiple vulnerabilities, a replicated diverse attack will occur less often. Will this longer search time tend to compensate for the possibly greater effectiveness of a diverse attack? That is, will the greater likelihood of causing failure in all replicas be offset by the longer time it takes to discover multiple vulnerabilities?

Our fourth and final model shows that this is not the case. If we use the same activity rates our last PEPA model shows that the flaw conservation or replicated attack strategy can be more effective than an aggressive strategy. The final model causes a diverse replicated attack to take longer to construct. So an attack happens less often, but its greater effectiveness outweighs the influence of the delay.

To match the characteristics of the intruder to the model we modify the previous intruder process to search sequentially for a vulnerability in each diverse

$$Intruder \stackrel{def}{=} (searching, h).(searching, h).(searching, h).Attack;$$
$$Attack \stackrel{def}{=} (start\_attack, k).(attack, a).Attack + (start\_attack, el).Intruder;$$

Figure 7: PEPA Model of Replicated Intruder

$$Server_i \stackrel{def}{=} (attack, \top).Triage_i;$$
$$Triage_i \stackrel{def}{=} (start\_triage, rt).(recover, r_1).Server_i + (start\_triage, at).(alarm, b).Alarm_i + (start\_triage, ft).Fail_i;$$
$$Alarm_i \stackrel{def}{=} (start\_alarm, ra).(recover, r_2).Server_i + (start\_alarm, fa).Fail_i;$$
$$Fail_i \stackrel{def}{=} (start\_fail, rf).(recover, r_3).Server_i + (start\_fail, ff).(fail, f).Fail_i;$$

Figure 8: PEPA Model of Server for Replicated Intruder

replica, before launching any attack. This intruder model is shown as Figure 7 The model of the replicated server must now be structured to permit diverse replicated attacks. By using the PEPA cooperation operator on a empty cooperation set we can establish completely concurrent operation of each server replica. This allows us to model a concurrent attack on all three replicas.

Our replicated server now looks like three processes $Server_i,\ i = \{0, 1, 2\}$ as shown in Figure 8.

Notice that now each server can recover independently. Otherwise, the intruder could attack concurrently but the servers could only recover sequentially, giving an unrealistic advantage to the intruder.

The replicated server and intruder are combined as shown by Equation 4

$$Intruder \underset{\{attack\}}{\bowtie} (Server0 \bowtie (Server1 \bowtie Server2)) \tag{4}$$

The replicated attack versus replicated server model has 3,645 states. It spends about 31% of its time as either the process shown as Equation 5 below, or the two similar ones that have recovery taking place in $Server_1$ or $Server_2$ instead of $Server_0$

$$(attack, a).Attack \underset{\{attack\}}{\bowtie} (recover, r_3).Server_0 \bowtie (fail, f).Fail_1 \bowtie (fail, f).Fail_2 \tag{5}$$

The model also spends about 16% of its time as the process

$$(attack, a).Attack \underset{\{attack\}}{\bowtie} (fail, f).Fail_0 \bowtie (fail, f).Fail_1 \bowtie (fail, f).Fail_2 \quad (6)$$

In total, the model spends roughly 80% of its time in a process that has no $Server_i$ process active. That is, each of the three servers is either failed, recovering, or in an alarm state. This result is more surprising than it first appears. The intruder clearly will do better with replicated attacks since the servers cannot use the intruder's search time to perform any triage. However, replicated attacks require more time to discover; in this model exactly three times longer. Our model would indicate that it can be worth the wait to build a more diverse attack.

4.3 Presentation of Results

We prefer to present our results in terms of the *mean time to discovery* of a vulnerability [23], represented by the rate constant h in our PEPA models, as shown in Figure 9 below. The constant h gives the rate of vulnerability discovery. The value of h in these models approximates the success of vulnerability prevention efforts applied to the servers during development and installation. That is, smaller values of h can be interpreted to indicate increased hardening or intrusion resistance of the server. This is very useful for trade-off studies between component hardening and architectural choices. That is, the models help us compare the merits and demerits of hardening certain components rather than replicating them or placing them behind protective boundary controllers.

The quantitative results presented in this fashion show how the more sophisticated intruder impacts the diverse replication. If we assume an intruder with an aggressive flaw exploitation strategy then the replicated server shows a dramatic increase in survivability. On the other hand, if we assume an intruder who conserves flaws before attacking, the improvement is still significant, but not necessarily adequate for high-consequence systems. Finally, we see that hardening can bring the single server up to approximately the level of a replicated server. At this time, there is no repeatable approach to hardening software to the required flaw density[1].

5 Conclusions

Human-sponsored faults are subject to the full range of human innovation. An abstraction that simplifies this to a small collection of statistics about fault effects or time to breach hides this innovation and tells little about the actual intrusion-tolerance or survivability of a system. Intruders are best characterized by directly defining their attack potential.

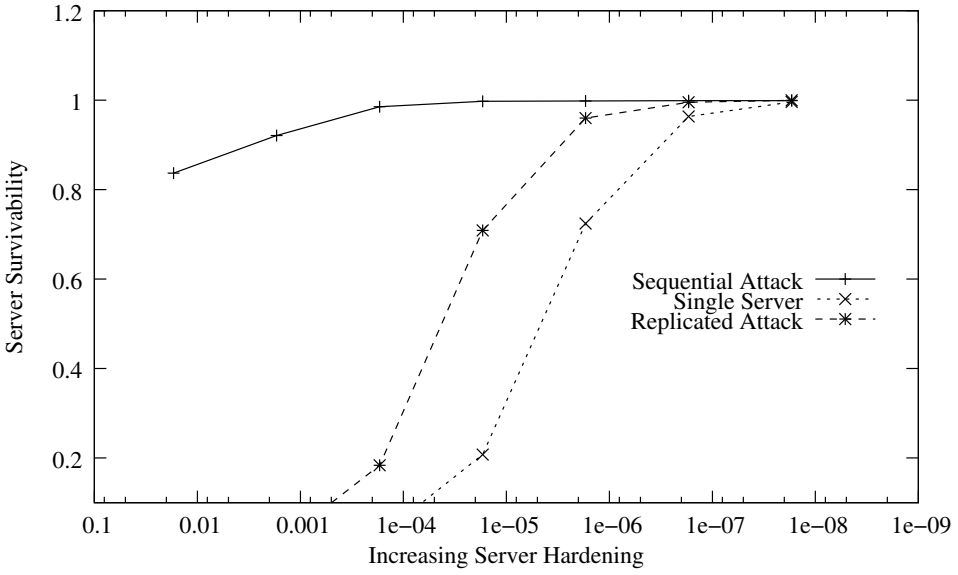

Figure 9: Server Survivability Against Different Intruders

Quantitative modeling of survivability for validation or measurement of high-consequence systems should be based on detailed intruder models. Survivability validation results cannot be applied without a well-defined and precise notion of the intruder's attack potential.

Detailed aspects of the intruder's attack potential can have significant impact on the expected survivability of an approach. Attack potential should be defined in terms of intruder capabilities, initial access, and initial knowledge. Quantitative models should make these non-quantitative or behavioral assumptions clear as part of their results. A contingency analysis based on variation of attack potential should be the basis for presentation of results.

Merely clarifying assumptions about intruder behavior is not sufficient for high-consequence systems. Weaker assumptions are to be preferred. That is, the quantification of survivability or security should be based on hard intruders. Intrusion-tolerance mechanisms such as diverse replication are expensive and not likely to be used for low-consequence applications. For this reason it is imperative to understand survivability with respect to hard intruders rather than a range of probable intruders.

Determining a suitable hard intruder is not trivial. As our last two models demonstrated, an apparently hard intruder may not be as hard as it looks. A simple change in capability or initial knowledge can significantly increase an intruder's attack potential.

We have also shown, incidentally, that stochastic process algebra can be useful for attack-potential-based quantitative modeling of survivability. As Donatelli et al. report [5] stochastic process algebra is a less mature technology

than generalized stochastic Petri nets (GSPN) but offers a more explicit compositional structure. This clarity of composition facilitates the construction of a collection of quantitative models that reflect structural differences in intruder attack-potential.

References

[1] R. Anderson. Why information security is hard - an economic perspective. In *Proc. 17th Annual Computer Security Applications Conference*, New Orleans, Louisianna, USA, December 2001. Also available in Chapt. 23 of R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, 2001.

[2] A. Avizenis, J. Laprie, and B. Randell. Fundamental concepts of dependability. In *Third Information Survivability Workshop*, Boston, MA, October 2000.

[3] R. Bazzi and G. Neiger. Simplifying fault-tolerance: providing abstraction of crash failures. *JACM*, 48(3), May 2001.

[4] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Trans. on Information Theory*, 29, 1983.

[5] S. Donatelli and J. Hillston. A comparison of performance evaluation process algebra and generalized stochastic petri nets. In *Proc. Sixth Int. Workshop on Petri Nets and Performance Models*, Durham, North Carolina, USA, October 1995.

[6] S. Gilmore, J. Hillston, and M. Ribaudo. PEPA nets: a structured performance modeling formalism. In T. Field, P.G. Harrison, J. Bradley, and U. Harder, editors, *Proc. 12th Int. Conference on Modelling Tools and Techniques for Computer and Communication System Performance Evaluation.*, LNCS 2324, London, UK, April 2002. Springer-Verlag.

[7] K. Goseva-Popstojanova, F. Wang, R. Wang, F. Gong, K. Vaidyanathan, K. Trivedi, and B. Muthusamy. Characterizing intrusion tolerant systems using a state transition model. In *DARPA Information Survivability Conference and Exposition II DISCEX II*, Anaheim, California, USA, June 2001.

[8] J. Hillston. *A Compositional Approach to Performance Modeling.* PhD thesis, University of Edinburgh, 1995. CST-107-94, also published by Cambridge University Press 1996.

[9] J. Hillston. *A Compositional Approach to Performance Modelling.* Cambridge University Press, 1996.

[10] E. Jonsson and T. Olovssson. A quantitative model of the security intrustion process based on attacker behavior. *IEEE Trans. on Software Engineering*, 23(4), April 1997.

[11] G. Karjoth and J. Posegga. Mobile agents and telcos nightmares. *Annales des Télécommunications*, 55(7/8):29–41, 2000.

[12] O. Kolesnikov and W. Lee. Advanced polymorphic worms: Evading IDS by blending in with normal traffic. //http:/www.cc.gatech.edu/~ok/w/ok_pw.pdf, Retrieved 9 November 2004.

[13] J.-C. Laprie, J. Arlat, J.-P. Blanquart, A. Costes, Y. Crouzet, Y. Deswarte, J.-C. Fabre, H. Guillermain, M. Kâniche, K. Kanoun, C. Mazet, D. Powell, C. Rabéjac, and P. Thévenod. *Dependability Guidebook*. Cépaduès-Editions, Toulouse, 1995.

[14] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann. Towards operational measures of computer security. *Journal of Computer Security*, 2((2-3)), 1993.

[15] R. Ortalo, Y. Deswarte, and M. Kaâniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25(5):633–650, Sept.-Oct. 1999.

[16] P. Pal, M. Atighetchi, F. Webber, R. Schantz, and C. Jones. Reflections on evaluating survivability: The APOD experiments. In *In Proc. 2nd IEEE International Symposium on Network Computing and Applications (NCA-03)*, Cambridge, MA, USA, April 2003.

[17] D. Powell and R. Stroud. Malicious- and accidental-fault tolerance for internet applications: Conceptual model and architecture. Technical report, MAFTIA deliverable D2 (available as LAAS-CNRS Rep. 01426 or University of Newcastle upon Tyne CS-TR-749), November 2001.

[18] J. Rochlis and M. Eichin. With microscope and tweezers: the worm from MIT's perspective. *CACM*, 32:689–698, June 1989.

[19] F. Schneider and L. Zhou. Distributed trust: Supporting fault-tolerance and attack-tolerance. Technical Report TR 2004-1924, Cornell Computer Science Department, January 2004. submitted for publication.

[20] G. Schudel and B Wood. Adversary work factor as a metric for information assurance. In *Proc. New Security Paradigms Workshop*, Ballycotton, County Cork, Ireland, September 2000.

[21] S. Singh, M. Cukier, and W. Sanders. Probablistic validation of an intrusion-tolerant replication system. In *Proc. 2003 Int. Conf. on Dependable Systems and Networks DNS'03*, San Francisco, California, USA, June 2003.

[22] stealth. Kernel rootkit experiences. *Phrack*, 0x0b(0x3d):Phile 0x0e, July 2003.

[23] F. Stevens, T. Courtney, S. Singh, A. Agbaria, J. Myer, W. Sanders, and P. Pal. Model-based validation of an intrusion-tolerant information system. In *23rd IEEE Int. Sym. on Reliable Distributed Systems (SRDS04)*, pages 184–194, Florianpolis, Brazil, October 2004.

[24] K. Tan, J. McHugh, and K. Killourhy. Hiding intrusions: From the abnormal to the normal and beyond. In F. Petitcolas, editor, *Information Hiding 2002 LNCS 2578*, pages 1–17. Springer-Verlag, 2003.

[25] K. Thompson. Reflections on trusting trust. *CACM*, 27(8), August 1984.