ABSTRACT

Title of Document: SYSTEM DYNAMICS MODELING AND SIMULATION OF ENTERPRISE COMPUTER SECURITY

Shalom Nachum Rosenfeld, Master of Science, 2006

Directed By: Professor Michel Cukier, Reliability Engineering
Dr. Ioana Rus, Fraunhofer Center USA

To support decision-making, training, and understanding complex trends in enterprise computer security, we have built an executable model representing the major components of an organization's computer security, including its machines, users, administrators, countermeasures, and attacks. We use "if-then" rules to express behaviors, incorporating the notions of “archetypes”, i.e. frequently-observed patterns of system behavior, and “system dynamics”, a discipline which views system behavior in terms of stocks and feedback loops. This thesis describes the model, and then discusses several archetypal behaviors and their results, namely: *Symptomatic Fixe*s (or “*Shifting the Burden*”), *Escalation*, and *Escalation* combined with *Limits to Growth*. Simulation is used to display these behaviors quantitatively, and to show the effects of possible solutions. We conclude by discussing how such results can be useful for practical computer security, and how this model can both feed off other security research and fuel it.

SYSTEM DYNAMICS MODELING AND SIMULATION OF ENTERPRISE COMPUTER SECURITY

By

Shalom Nachum Rosenfeld

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:
Professor Michel Cukier, Chair
Dr. Ioana Rus
Professor Min Wu

Acknowledgements

The author gives his highest thanks to his advisors, Dr. Michel Cukier and Dr. Ioana Rus, for their tireless dedication, for believing in him, and for working so hard to give him the opportunity to make this thesis happen. The entire E.C.E. and Reliability departments, as well as the Fraunhofer Center, are also to be thanked for all of their assistance.

The author also thanks Dr. Min Wu for her assistance on the examining committee and incisive commentary and questions.

Interestingly, the author's first introduction to archetypes occurred when Dr. Virgil Gligor tangentially described the Tragedy of the Commons during a lecture on distributed systems.

In addition to the above, a series of mentors (from a wide variety of fields) should be acknowledged; what follows is an incomplete list: Rabbi Shlomo Crandall; Rabbis Emanuel, Israel, and Rafael Moshe Gettinger; Rabbi Rafael Pollack; Rabbi Simcha Fishbane; Rabbi Michoel Elias; Rabbi Yehudah Shmulevitz; Rabbi Yitzchak Breitowitz; Dr. Elliot Bartky; Mr. Mitch Aeder; Professor Kenneth Kramer. Their individual contributions are too great to be enumerated here.

The author thanks his colleagues from the Reliability Lab and from other E.C.E classes, whose assistance proved invaluable.

None of this could have happened without the role models of the author's parents, grandparents, and entire extended family.

The author thanks his wife for her care, patience, and support.

Finally, the author thanks G-d for all of the above – and everything.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

An enterprise computer system is highly complex, consisting of multiple hosts with different platforms and different applications, all networked and most likely connected to the Internet. These components have flaws that make the system vulnerable and allow attackers to exploit these vulnerabilities.

Humans and machines form an even larger and more complex system with many different components and interactions. Control actions and reactions on one side of this system might have not only a local effect, but could also affect the rest of the system, often resulting in feedback loops. These effects manifest themselves over time with different delays. The properties of the system (security being one of them) will emerge from its structure and all these interactions between its components.

Some of the events in such systems are non-deterministic. This, and the fact that we do not have complete and fully accurate knowledge about these systems, leads to a level of information uncertainty that must be acknowledged and handled appropriately. Due to all of the above intricacies of such a system, it is extremely difficult to understand and analyze its emerging properties and the properties of the services it provides.

It is a hard task to characterize and assess the security of such a system, let alone to predict malicious acts and to design a strategy for eliminating or at least reducing their effects. Nonetheless, such a strategy is imperative, especially for systems such

as national infrastructures, military or other government systems, emergency systems, or banks.

Protection against attacks can be achieved by preventing, detecting, and tolerating them. Tolerating attacks might require the system to function in a degraded mode once under attack. If attacks defeat all lines of defense and eventually succeed, then the system must be able to recover quickly to an operational and secure state. Of course, all actions needed for proper prevention, detection, and tolerance have costs associated with them, including the price of buying and maintaining tools, the effort and time to install and run them, and personnel training. A strategy for security achievement and risk reduction can comprise a combination of the aforementioned actions. Given resource constraints, as well as trade-offs that might be needed between security on one hand and other operational properties (for example usability or performance) on the other hand, designing such a strategy is a very challenging task and requires extensive knowledge and experience.

1.2 Approach

To support this decision-making process of designing an appropriate security strategy, we developed a quantitative executable model of an organization's operational computer security. Like all models, this is an abstraction of the real system, focused on representing the security-significant aspects of the system and associated processes. The model targets and represents the perspective of the person who must make decisions regarding actions that must be taken for security assurance and security-related risk management. The user of the model can set different values

for the model parameters, corresponding to different usage, vulnerabilities, attacks, and defense profiles. The simulator can be run and different "what-if" scenarios can be executed. Simulation will help a security manager, security engineer, or system administrator answer questions such as: if my environment is characterized by these values, then what methods and tools to select and apply for managing security risks and satisfy the users needs of my system? How will the selected actions work together? What is their effectiveness and cost efficiency? To what changes is my environment most sensitive? If I make specific changes in my security strategy, what will be their impact? What changes if my system gets attacked more/less or if the time to exploit changes? Should I hire more system administrators? Should I spend more on training them?

The model aims first at understanding security risk reduction in computer systems, then at diagnosing such systems and identifying their weaknesses, as well as prospectively examining the effectiveness of different solutions. The description of the behaviors this model can exhibit is founded upon the notion of system archetypes.

1.3 Archetypes

Archetypes are a concept related to systems thinking, developed in the mid 1980s, in an attempt to describe complex behavior and to convey ideas in an easier and more efficient manner. Archetypes are frequently-observed patterns of systems behavior and are a "natural vehicle for clarifying and testing mental models" about systems or situations [For61]. The systems literature describes ten distinct archetypes, as listed by [Bra02] and outlined in Appendix I. [Wol03] argues that in fact, all of these can be

categorized into one of four "core generic" archetype classes: "Underachivement" includes *Limits to Growth, Attractiveness Principle, Tragedy of the Commons,* and *Growth and Underinvestment*; and "Relative Archievement" includes *Success to the Successful*. "Out-of-Control" includes *Fixes that Fail, Shifting the Burden,* and *Accidental Adversaries;* and lastly, "Relative Control" includes *Escalation* and *Accidental Adversaries*. [Wol03] acknowledges that the more common description of archetypes (i.e. that of [Bra02]) is more intuitive and easier to grasp and apply to simulation, so it is used here. Archetypes have been mainly applied in business or industrial processes. There has recently been some work performed at MIT in applying systems thinking and archetypes to systems safety [Mar03], but in security this is a new idea.

Beyond the common archetypes of [Bra02], we keep in mind that other archetypes may be observed in security. This would not be surprising, as [Mar03]'s application of archetypes to safety engineering uncovered several security-specific archetypes. This thesis, however, restricts itself to the application of common archetypes to security. While Appendix I describes how each of the ten archetypes might be applied to security, this thesis gives a detailed understanding of the following archetypes: *Symptomatic Fixes* (also known as *Shifting the Burden*), *Escalation, Limits to Growth,* and a combination of the latter two.

We use archetypes for understanding and modeling security aspects (needs, problems, actions) in the context of an enterprise that uses computers/information technology systems for running its business and needs to ensure the security of its information, services, and/or systems. We are representing and simulating security-

related organizational behavior and trends and using archetypes for documenting and understanding the domain, the problems, and their potential solutions. Mental models might be able to handle archetypes in isolation, but for the entire system (which contains combinations of such archetypes) mental models are not adequate due to the complexity, non-determinism, and uncertainty of the system. Computer simulation is in fact already recommended in [Sen94] for extending one's grasp of archetypes.

1.4 The Model

For our model, we employ the continuous modeling feature of the Extend simulation environment [Ima05]. This is a graphical simulation tool that focuses on the levels of holding tanks and their inputs and outputs, governed by constants, equations, delays, and random values. (A screenshot of a holding tank and its inputs and outputs can be found in Appendix II.) The level of each holding tank changes at each simulation step, and a typical simulation run can consist of hundreds or even thousands of such steps. The result is an easy-to-use way to set up and numerically solve systems represented by a series of differential equations. The feedback loops stressed by system dynamics and archetypes can easily be represented by a holding tank whose output is connected to its input. Thus, continuous modeling with Extend is a good fit for the system dynamics modeling approach described above.

Out model consists of approximately 350 Extend basic "blocks", such as constants or holding tanks. We outline it here, with complete details left for Appendices II and III.

In the model, staff-hours (of the system administrators) can be allocated to various tasks related to the security of a typical system. We model the following seven countermeasures:

- **"Firewall Efforts."** Overseeing and maintaining the system's firewall.
- **"Antivirus Efforts."** Maintaining the system's antivirus software, keeping it updated, resolving user issues related to the antivirus.
- **"Intrusion Detection System (IDS) Efforts."** Maintaining the IDS, installing new signatures, resolving alarms.
- **"Encryption Efforts."** Maintaining the system's encryption software.
- **"Enforcement Actions."** This includes tasks such as: scanning for and fixing configuration vulnerabilities, which are effectively "doors" to the system that were inadvertently left open; monitoring the users to prevent unsafe practices, such as downloading viruses or using "weak" passwords which are easily guessed; applying proper access control to prevent unauthorized use; and more generally, devising and enforcing a company security policy. See [Dan04] for more on these tasks. All of these require no additional hardware or software *per se*, only a great deal of attention from the support staff (or system administrators).

These appear as the five most prevalent "security technologies" used in Gordon's survey ([Gor05a]) of 700 corporate, governmental, and academic institutions, where we have subsumed Gordon's "Access Control Lists" under our term "Enforcement Actions." To these five we add a task familiar to any computer user:

- **"Software Patches."** Downloading and installing patches to correct vulnerabilities in the operating system(s) and applications; resolving problems caused by patches.

Lastly, we consider a somewhat different approach that has only recently been discussed by the security community:

- **"Tolerance Measures."** This includes designs to *tolerate* an attack (rather than prevent or detect it), even if it succeeds. Multiple layers, graceful degradation of performance, and (in some instances) backups are all tolerance measures.

In our current model, the effectiveness of each countermeasure is a factor only of the countermeasure's presence or absence (implemented as a series of Y/N switches in the model) and the number of staff-hours per machine allocated to the corresponding task. Although the IDS and firewall seem independent of the system size, additional machines will mean additional alarms, which will require more attention. Additionally, the system has an overall vulnerability measure, which is reduced by the number of staff-hours per machine allocated to enforcement actions and software patches.

The attacks on the system are divided into two categories: **"Simple" (or "kiddy-script") attacks** tend to rely on known vulnerabilities and require little action from the attacker other than downloading and running the attack. **"Sophisticated attacks"** may involve finding new vulnerabilities, can often defeat many countermeasures, and usually come from a single knowledgeable attacker, such as one who might actually write the "kiddy-scripts" of the former category. While viruses, which are the

costliest type of attack according to the respondents of [Gor05a], are written by some very sophisticated attackers, an existing virus propagates in well-understood ways and can be easily defeated by the proper countermeasures; we thus include viruses in the "simple attack" category.

For both categories (simple and sophisticated), a specified number of attacks are considered to be attempted against the system each day. (Alternatively, the simulation can also be set to add some random variation to the specified number of attempts.) Given the effectiveness of each of the various countermeasures, and the system's vulnerability (or lack thereof), a fraction of those attacks will succeed. The primary outputs of our current model, then, are the numbers of "successful simple attacks" and "successful sophisticated attacks." Note that a result of "*n* successful simple attacks" may not appear as *n* separate incidents. Several of these may exploit the same vulnerability, turn out to be variants of the same virus, and so on. For now, the number of successful attacks should be taken only as our metric of the quality of countermeasures versus attempted attacks.

1.5 Thesis Structure

The remainder of this thesis is structured as follows: Chapter 2 introduces the *Symptomatic Fixes* (or "*Shifting the Burden*") archetype; describes one instance of it in computer security as we have modeled it; discusses the results of several different simulations based on it; and considers how this archetype might apply elsewhere in security. Chapter 3 goes through a similar approach with the *Escalation* archetype. Chapter 4 introduces the *Limits to Growth* archetype, whereupon an instance is

described that describes a combination of *Limits to Growth* and *Escalation*. Chapter 5 outlines related work, and Chapter 6 gives conclusions and some future work. This is followed by Appendices I, II, and III, a bibliography, and finally a list of this author's publications and submissions.

Chapter 2: *Symptomatic Fixes* Archetype

2.1 Symptomatic Fixes Description

In this archetype, the symptoms of a problem are observed. Rather than analyze the root cause of the problem, the manager (or "decision-maker", or "actor") attempts to fix the symptom. This "shifting of the burden" from the problem's actual cause to its symptom often distracts the manager from the former; it can also mask the symptoms of the original problem, making it more difficult to diagnose.

Armed with an understanding of this archetype, a manager will consider the possibility that the most readily apparent solution may not ultimately be the best one. Instead, time must be taken to analyze, and only then properly treat, the root cause.

For a simple illustration in computer security, we paint a scenario in which a company's computer system (or just "system") is continually falling prey to successful attacks known as "kiddy-scripts." These attacks are launched by novice attackers, and generally only succeed if the system contains vulnerabilities such as software that is not up-to-date. The successes of these attacks should be seen as a symptom of a deeper problem. Reducing the system's vulnerability to thwart these attacks could be considered a fundamental solutions; such a fundamental solution would include the frequent installation of software patches. It is possible (in fact, likely) that implementing such a solution properly will take time and thus not yield dramatic gains very quickly; in the long run, however, positive effects of this solution will be observed. We choose software patches as one action that can be taken to

reduce overall system vulnerability vis-à-vis kiddy-scripts, though it is certainly not the only action.

Alternatively, it is all-too-possible for a company to instead view the successful attacks as the only issue here and therefore install an Intrusion Detection System (IDS) to detect the occurrence of these attacks – a symptomatic fix. The company's support staff (or "system administration staff") is then too distracted from installing patches. In time, many new vulnerabilities will be discovered in the software run by the system; once published, these will be exploited by new "kiddy-script" attacks. Invariably, a certain percentage of attacks do evade an IDS, and thus, as the known vulnerabilities in the system increase, the number of successful attacks will also increase, despite the company's continued efforts to install, maintain, and improve their IDS. These effects are displayed in Figure 1, an "influence diagram" showing the effects of given variable on one another over time.

In this diagram, we begin in the center with the problem symptom of successful simple attacks. In the loop beneath the symptom, we see the fundamental solution: increased successful simple (or "kiddy-script") attacks cause an increased need for the fundamental solution of applying software patches, and, in fact, applying this solution will reduce the problem symptom. Such a loop can be described as "more of A leads to less of B leads to less of A, and so on until equilibrium is reached", and is known as a "balancing loop." Alternatively, the symptomatic solution is found in the loop above the problem symptom. If we focus on this loop itself, it appears to offer the same advantages as the fundamental fix, sometimes more easily or more rapidly in the short term (though this is not indicated in the influence diagram).

Unfortunately, though, we also see that an increased use of the IDS can increase a side effect: the distraction of the support staff from other tasks, including patch application. This, of course, reduces the chance of a fundamental fix being applied. Starting at the top of the diagram and proceeding around its periphery clockwise, we see: increased IDS efforts leads to an increase in support staff distraction, therefore less patches are applied; the problem symptom will re-emerge, and more of the symptomatic fix will be attempted. This loop can be described as "more of A causes more of B causes more of A, and so on", and is known as a "reinforcing loop." [Wol03] includes this archetype under his more generic term "Out-of-Control", as a balancing loop is desired to control the problem symptom, but it is not obtained.

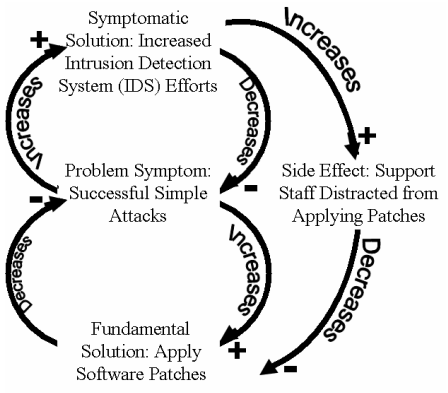

Fig. 1. Influence Diagram for a "*Symptomatic Fixes*" Situation.

2.2 Simulation Setup

To see quantitative results, an Extend model was used simulating a system containing on the order of 200 machines, sustaining 100 simple attacks per day. A certain percentage of these attacks will be defeated by an IDS (and depending on how

well the IDS has been maintained), and a certain percent will be defeated if the system's software is well-patched. Note that even if we say "*n*% of the attempted attacks succeeded", the system's users may not observe for 100 attempted attacks, *n* separate failures, as many of these attempts might target a small set of specific vulnerabilities and exploit them in the same way. Similarly, no single countermeasure should be expected to reduce the attack success rate to 0 by itself, as there are enough different types of attack that any single countermeasure can be defeated. We use the percentage of successful attacks only as a measure of the system's defenses and vulnerabilities. It is assumed that the software of this system is initially patched partially; therefore there is room for improvement if further patching is undertaken, while a loss will be felt if patching is ignored (as the discovery of new vulnerabilities will bring the software's status from "partially patched" to "mostly unpatched.") The model was executed for the equivalent of 6 months (real time) with different scenarios. (Each execution of this type runs in under 30 seconds on a conventional Pentium III computer running Windows 2000 Professional.)

For examining the effect of different effort allocation to the fundamental and symptomatic solution, we executed the simulation for four scenarios s1, s2, s3, and s4. In all four scenarios, the system is under pressure for the first d_1 days while the rate of successful attacks rises. This is due to the discovery of additional vulnerabilities. On day d_1+1, however, the company embarks on some course of action. Here we chose $d_1 = 9$, to demonstrate the effects over several days of taking no action at all.

In our first scenario, "s1", from day d_1+1 onwards, the company has its support staff dedicate a certain number of staff-hours per day to installing software patches to all the system's computers. This effort is held constant throughout the six-month period. The "if-then" rule that describes the organization's efforts in this scenario is given by:

IF: (Day> d_1)

THEN: Staff-Hours for Patches := x_1.

For the hypothetical situation that we are modeling, we considered 3 staff-hours a reasonable value for x_1 given the description of our system. This is considered the "fundamental fix" scenario, or the "solution" to the *Symptomatic Fixes* archetype.

In our second scenario, "s2", the company deploys an IDS on day d_1+1. For the next 170 days, efforts are gradually increased to maintain and improve the IDS: as new attacks are discovered, new plug-ins are added; as a consequence, more alerts that are signaled by the IDS must be analyzed, requiring more effort (although some of them might be just false alarms). In an attempt to keep the IDS functioning well, the company increases its IDS efforts with the following rule:

Begin with y_0 staff-hours for the IDS.

FOR: every day

IF: (Successful Attacks today > Successful Attacks two days ago)

THEN: increase staff-hours for IDS by y.

We have assigned the values y_0 = 1.5, y = 0.03. (This will lead to a gradual increase from moderate IDS effort at day ten to a strong IDS effort of approximately seven staff-hours by the end of the simulation.) Meanwhile, as IDS efforts increase,

less efforts are available for patches: *Staff-hours for Patches := 4 – Staff-Hours for IDS,* to a minimum of zero. We consider this our case of "increasing efforts to the symptomatic fix while decreasing efforts for the fundamental solution", or a strong instance of the "problem" archetype.

Our third scenario, s3, takes this a step further: as of day d_1+1, the same IDS efforts are made as in s2, but no patch efforts are made at all. Here we interpret the increasing side-effect loop in Figure 1 as the strengthening over time of the "mental barrier" (as [Wol03] calls it) that prevents consideration of the fundamental solution. Additionally, the side-effect loop is common for this archetype but not required, see [Sen90]. In any case, s3 is an even more extreme case of the problem archetype for *Symptomatic Fixes.*

Lastly, our fourth scenario s4 considers an alternative solution, one which the archetype literature concedes as sometimes viable. If the company understands its priorities, then it may be possible to use *both* the fundamental solution *and* a small dose of the quick fix. This would be codified by the following rules:

Staff-hours for Patches := 4 – Staff-Hours for IDS, as before.

The difference is the rule for IDS efforts:

Begin with y_0 staff-hours for the IDS.

FOR: every day

IF: {

(Successful Attacks Today > Successful Attacks two days ago)

AND (staff-hours for IDS <= z) },

THEN: increase staff-hours for IDS by y.

The value of y is the same 0.03, but y_0 is now reduced to 0.2. As in s1, we assume that a proper effort for patches can not be made with less than three staff-hours, so we set z to 1. s4 can thus be described as "symptomatic fix supplementing the fundamental fix." (Note that no "burden" is being shifted *per se* if the company understands what is fundamental and what is not.)

2.3 Results and Discussion

The primary outputs of these four scenarios, i.e. number of attacks successful per day, are plotted for comparison in Figure 2. We can also integrate under the curves of Figure 2, giving us the number of cumulative successful attacks for each of the four scenarios; these will be displayed in Table I below.

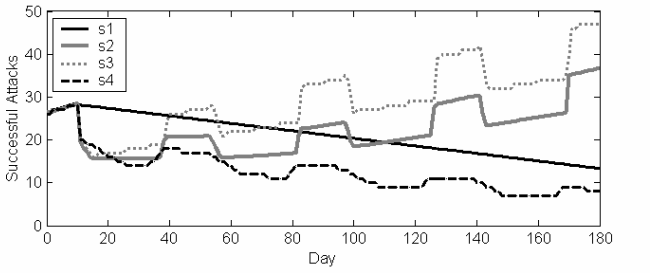

Fig. 2. Successful Attacks per Day for the Four *Symptomatic Fixes* Scenarios

Several important features can be observed in Figure 2. Firstly, when comparing the fundamental solution (s1) to the symptomatic fixes (s2 and s3), we see that the symptomatic fixes appear to do a much better job initially (e.g. looking at Day 30, s2 and s3 are approximately 10 successful attacks lower than s1), but by the end of the simulation period, the fundamental solution is far more successful: at Day 180, s1 is

22 successful attacks lower than s2, and 32 lower than s3. This demonstrates a common pattern in the performance of symptomatic fixes – while the symptomatic fix can cause temporary drops in the problem rate, the overall trend over time is for the problem rate to increase. (Diagrams similar to Figure 2 are seen in describing this archetype in [Sen94] and [Bra02].) While the security staff is distracted by the rises and falls in the performance of the IDS, the system's current software vulnerabilities, as well as those newly discovered, are neglected, leading to a rise in the percentage of attacks that are successful. The overall trend is a linear increase; this is not surprising, as we have modeled the vulnerabilities in unpatched software as increasing linearly in time. Comparing s1 against s2 and s3 also stresses the importance of behavioral monitoring over time. Were we to stop the simulation after one month or so, our conclusions would be very different as to what measures are most effective!

Focusing on s3, we see that it presents an even more extreme case of s2's failures, as the patch efforts have been eliminated entirely. Lastly, we turn our attention to s4. Recall that s4 begins with less IDS efforts than s2 and s3; it therefore appears initially to allow more attacks to succeed, e.g. at Day 16, s4 is 2 successful attacks higher than s3, and 3 higher than s2. However, by the end of the simulation period, s4 is clearly the winner in reducing successful attacks. Notice as well the height for the "waves" of symptomatic fixes: they are greatest in s3, smaller in s2, and smaller still in s4; this height represents the degree of the "crisis/fix" pattern, which is lowest when the proper application of fundamental fixes prevents crisis action (s4), and greatest when no fundamental fix is present (s3). Lastly, while s4 clearly prevents more attacks than

s1, notice how they approach each other asymptotically – in the long run, adding the symptomatic fix will cease to provide any good beyond the fundamental solution.

We now turn our attention to the effort required in each of these scenarios. s1 consisted simply of a constant 3 staff-hours per day for patches, and nothing else. Figure 3 shows the efforts of the support staff, in staff-hours per day, invested in s2, in which IDS efforts decreased patch efforts.

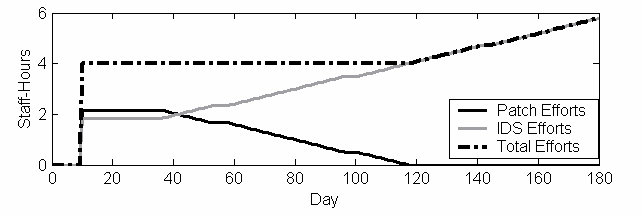

Fig. 3. Efforts per Day for s2, the IDS-Decreases-Patches Scenario

Notice how patch efforts decrease steadily until approximately Day 120, at which point they stay at zero for the remainder of the simulation. Until Day 120, any efforts for IDS came out of efforts for patches, so total efforts were constant; after Day 120, the total efforts are all IDS efforts. Figure 3 further highlights the attractiveness of the symptomatic fix, as the initial IDS effort requires less staff-hours per day (less than 2) than what would be required of a fundamental fix (a steady 3 for s1). In the long run, however, staff-hours are continuously added to the IDS effort in an attempt to raise its results; by the time six months have passed, the company realizes that it is investing 6 staff-hours per day into the IDS. We can also integrate the curves in Figure 3 to measure cumulative effort of the simulation period, to be shown in Table I.

In s3, the only efforts present are those for IDS. These are shown in Figure 4.

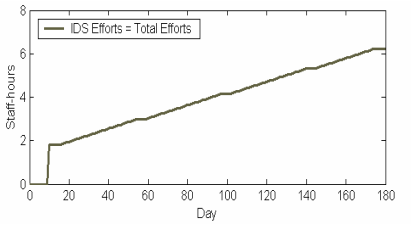

Fig. 4. Efforts per Day for s3, the IDS-only Scenario

Observe that by Day 180, approximately 6.25 staff-hours are being used for IDS efforts. In s2 (see Figure 3), that number was only approximately 5.8. The same rule produced both figures: "increase IDS efforts every day that successful attacks are higher than they were two days ago." Compared to s3, s2 allowed for some patches as well, so there were less days when this trigger occurred, therefore less IDS efforts were demanded over the course of the simulation.

Lastly, Figure 5 displays the efforts of s4, which combined IDS and patch efforts with an emphasis on the latter.

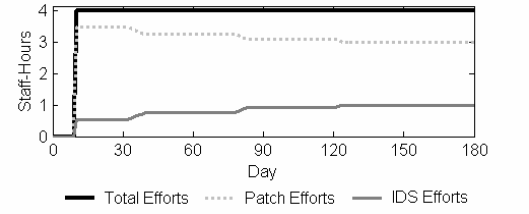

Fig. 5. Efforts per Day for s4, the IDS-Supplements-Patches Scenario

Notice how few increases are made to IDS efforts. Again, this happens because the trigger condition of successful attacks being too high is very rarely met, due to the appropriate patching strategy.

We now compare all four scenarios in terms of their cumulative effort and cumulative successful attacks, as displayed in Table I.

Considering cumulative values, we see indeed that cumulative successful attacks are lower for s1, the fundamental solution, than for symptomatic fix scenarios s2 and s3.

Table I: Cumulative Successful Attacks and Efforts for All Four Scenarios

	Cumulative Successful Attacks	**Cumulative Efforts (Staff-Hrs.)**
s1	3833	513
s2	4007	740
s3	5232	689
s4	2345	684

Noticing that s1 requires over 100 less staff-hours' worth of effort than s2 or s3, we see that in the long run, the fundamental solution is not only more effective than the symptomatic fix; it is less costly as well. The only question remaining is in comparing s1, "fundamental solution alone", with s4, "fundamental solution combined with symptomatic fix." A company will have to decide for itself whether the additional 151 staff-hours of efforts are worth the reduction in 1500 successful attacks. How such calculations are made is touched upon in related work, below. In any case, simulation allows the company to consider the effects of its actions, and choose its optimal course with these effects in mind.

By analogy with these results, when other variables of interest in the system have a similar evolutionary trend, the *Symptomatic Fixes* archetype might be manifesting itself. In that case, the situation must be diagnosed and the real cause and the corresponding solution must be examined; this solution has to be applied, thus fixing the real problem. Of course, the results of this shifting must be monitored over time,

to make sure that the diagnosis was correct and that the solution was correctly implemented.

Lastly, the above simulations show the applicability of the model as a decision tool, by allowing one to see the effects of different proposed solutions before implementing them. In the example presented here, the decision was regarding the allocation of effort to different security efforts (IDS and patches). The model might also be useful in exploring and making security policies, as well as for training security staff.

2.4 Other Instances of Symptomatic Fixes in Security

We have presented only one possible instance of *Symptomatic Fixes* here, and thus we have opened the door to many related opportunities. Our simulation model includes many security-related tasks not described here (such as user training, enforcement of the security policy, and maintaining tolerance measures such as backups, to name a few), and in place of the Patch Efforts described here, this simulation could be run with other tasks or some combination thereof, as well as considering more-sophisticated attacks. Just as different parties may see different tasks as "the" fundamental solution ([Sen94]), attacks of different sophistication may have different "fundamental solutions."

Additionally, [Sen90] finds that the best way to describe the history of a particular company's strategies is by combining the *Symptomatic Fixes* archetype with another archetype, namely *Limits to Growth*. Thus, the applicability of this combination and other archetype combinations should be considered in computer security as well.

A variant on *Symptomatic Fixes* described in [Sen90] and [Sen94] is known as *Shifting the Burden to the Intervener,* in which the fundamental fix involves the internal actors repairing problems, and the symptomatic fix involves outsiders. This brings to mind some sentiments in the security community about security being incorporated into system design at each step of the process, rather than ignoring security and relying on an expert to add security features shortly before release or deployment.

Lastly, there has been much discussion in the security community (see [Hun06]) regarding whether better security behavior should be taught to the users of a system, or placed entirely on the shoulders of the system administrator. Similarly, in a system where the roles of system administrator and security officer are divided, the interactions between them may follow archetypal patterns. We had begun to document anecdotal accounts of such interactions, and our model leaves room to add detail to its human-factors portion of the model, including the interactions between users, system administrators, and security officers. *Shifting the Burden to the Intervener* could thus shed light on these human interactions.

For additional information of *Symptomatic Fixes* as it pertains to security, please see [Ros06b], from which this chapter was excerpted.

Chapter 3: *Escalation* Archetype

3.1 Escalation Description

In the *Escalation* archetype, each of two parties makes efforts and achieves results towards reaching its own well-defined goals. However, each party desires greater results than its counterpart. Thus, each party continues to increase its efforts, with neither party achieving dominance for an extended period. This can theoretically continue *ad infinitum*.

As an instance of this archetype in security, we investigate the action-reaction effects of attacks on an organization's computer system and the organization's attempts to better defend its assets, all the while advertising its strengths in an attempt to attract more business. We begin with a company that spends little on security measures, but sustains few attempted attacks because it's not a very well-known or worthwhile target. While some simple "kiddy-script" attacks blindly go after any available computer system and can be seen as the ever-present "attack noise", other simple attacks (such as a "Zombie DDoS", see [Gib02]) are consciously directed at an organization by an attacker. These are more likely if the organization is better-known. Furthermore, an organization will be targeted by sophisticated attacks if its assets are valuable (e.g. credit card numbers stored on its servers), or if its defenses are considered formidable, in which case breaching them poses a worthwhile challenge.

We suppose that the organization decides to attract new customers by increasing its security spending and advertising its new security strength. As the prominence

and/or asset desirability of the organization rise, the motivation to attack its system is increased, raising both the quantity and sophistication of attempted attacks. To counteract these, the company increases security spending again. Alas, this furthers the motivation to attack, leading to another increase in attempts. This process can continue for several more rounds.

These effects are displayed in Figure 6, an influence diagram showing the effects of given variable on one another over time. (Similar influence diagrams are drawn for archetypes in [Bra02].)

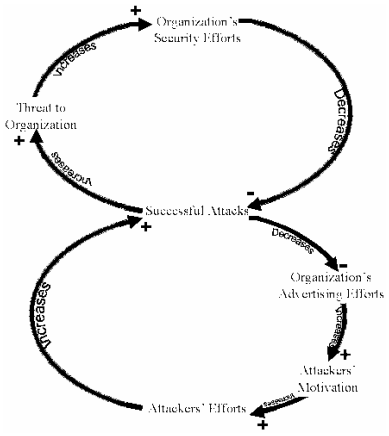

Fig. 6. Influence Diagram for *Escalation*

The upper loop in Figure 6 reads as follows: "Increasing the organization's security efforts will decrease the number of successful attacks against it. An increase in successful attacks leads to a greater threat to the organization. The greater the threat, the more security efforts will be added to counter it." Thus, if the attackers' efforts are constant, we would observe the following behavior: increased security

efforts will decrease the number of successful attacks, decreasing the threat to the organization, decreasing the need for additional security efforts. This forms a "balancing" or "negative" loop, as after several rounds of such behavior, no further efforts will be required.

A similar pattern is found in the lower loop: "Increased successful attacks cause the organization to advertise less. (We assume the resources that would have been funneled into advertising are now needed to recover from all of the attacks.) Advertising efforts increase the motivation to attack the organization, leading to more efforts on the part of the attackers, and therefore more successful attacks." Thus, the attacker behavior in and of itself should also form a balancing loop, as enough successful attacks will prevent any advertising, at which point the organization is no longer a very visible or worthwhile target, so attack efforts are not increased again.

However, in our scenario, both the organization and the attackers respond to one another, violating the assumptions we had made for balancing loops. Traversing the outermost loop of Figure 6 describes the overall behavior: an increase in the organization's security efforts increase its advertising efforts (or otherwise raise its prominence and asset desirability), increasing the motivation and therefore the efforts to attack the organization, leading to a rise in successful attacks. The organization feels threatened and therefore increases its security efforts, and the spiral continues from there. As both the organization's and the attackers' efforts continue to increase in time, this forms a positive loop. The number of successful attacks, however, reflects the ratio of attackers' efforts to the organization's security efforts, and thus should exhibit stable oscillations. [Wol03] describes this archetype as "Relative

Control", as each party's balancing loop is used in an attempt to gain control over the relative quantity "success of one party / success of the other party."

3.2 Simulation Setup

Clearly in our case, the number of successful attacks becomes the barometer of "success of attackers compared to success of defenders." Increased efforts by attackers over time can be modeled by an increasing number of attempted attacks, both simple and sophisticated. The organization's efforts can be fulfilled by: introducing countermeasures that were not previously present; changing the allocation of support staff-hours to various tasks; training the support-staff (which, to a point, increases their effectiveness); and increasing the staff-hours available for security tasks. The latter may require hiring in the long run, but in the short term may often be achieved simply by encouraging overtime, reassigning personnel within the company, etc.

In the simulation scenarios presented here, we have simplified by limiting the organization to one action, namely increasing staff-hours, and did not include other actions. We assume that all countermeasures are present, but they all begin with inadequate support staff. In time, increasing the staff-hours to each task will result in a greater number of attacks not successful. We have further simplified by scripting the actions of both the organization and the attackers as an automated series of "If-Then" rules, so the simulation runs without external intervention. The rules we use are based on our assumptions of how a company in such a situation would behave,

and they quantitatively capture the qualitative behavior described in Figure 6. These rules are as follows:

The organization decides to increase efforts:

FOR: e*very x_1 days*

IF: (Successful Simple Attacks > x_2)

THEN: increase staff-hours allocated to Antivirus, Firewall, IDS, Enforcement Actions, and Software Patches by $\vec{w} = \{w_1, w_2, w_3, w_4, w_5\}$, respectively.

These tasks begin with $\vec{w}_0$ staff-hours allocated at the start of the simulation.

These countermeasures and vulnerability-reduction tasks are very effective at preventing or detecting simple attacks. Faced with sophisticated attacks, however, their effects are diminished: the antivirus does not address these attacks, which aren't viruses; the IDS and firewall can sometimes be deceived; and enforcement actions and software patches can only reduce known vulnerabilities, whereas the sophisticated attacker may discover and exploit new vulnerabilities. Thus, the company responds to sophisticated attacks in a different way than to simple attacks:

FOR: every x_1 days

IF: (Successful Sophisticated Attacks > x_3)

THEN: increase staff-hours allocated to Encryption by v_1 and Tolerance by v_2. Tolerance and Encryption are allocated $\vec{v}_0$ staff-hours at the beginning of the simulation.

We assume that these countermeasures are no less effective against sophisticated attacks than against simple attacks. Today's commercial encryption is believed to be

unbreakable by any private individual with a handful of computers, no matter how clever, and tolerance works despite the success of the attack.

As some tasks may require more staff-hours than others to be done well, different numbers can be specified for each task. In any case, decisions to increase staff-hours are implemented as follows: *Any increase in staff-hours requires a d_1 day delay to reassign personnel. d_2 days after the increase occurs, the company advertises its added security efforts.*

This leads the attackers to launch additional attacks, according to the following assumed behavior: *Begin with y_1 simple attacks. Any day that advertising is present, increase the simple attacks by y_2%.*

Simple attacks can be increased rapidly, as this merely requires directing automated "kiddy-scripts" against the system. The number of sophisticated attacks, however, grows at a different (generally slower) rate: *Begin with y_3 sophisticated attacks. Any day that advertising is present, wait d_3 days as sophisticated attacks are prepared; then increase the sophisticated attacks by y_4%.*

In our execution, the number of simple attacks attempted is given by the above rules. To allow for some randomness, we chose to let the number of attempted sophisticated attacks vary by a (Gaussian) standard deviation of 5%. Additionally, if the number of successful sophisticated attacks is found to be between 0 and 1, then a random number is drawn to determine if the attack succeeds.

We simulate a system of approximately 200 machines, choosing a simulation period of six months (180 days). Keeping these numbers in mind, we have run the simulation with the following values: $x_1 = 7$, $x_2 = 4$, $x_3 = 1$, $\vec{w}_0 = \{1.8, 2.2, 6.0, 2.4,$

2.4}, $\vec{w}$ = {1.1, 1.3, 3.7, 1.5, 1.5}, $\vec{v}_0$ = {0.72, 0.87}, v_1 = 1.8, v_2 = 2.2, y_1 = 20, y_2 = 29.8, y_3 = 0.6, y_4 = 9, d_1 = 14, d_2 = 5, and d_3 = 2. In our opinion, these values, used with the above rules over a 180-day period, describe a linear progression from minimal attention to complete dedication vis-à-vis staff-hours for security tasks.

3.3 Results and Discussion

Successful attacks per day are used as our measure of "organization's efforts vs. attackers' efforts"; the results are shown in Figure 7.

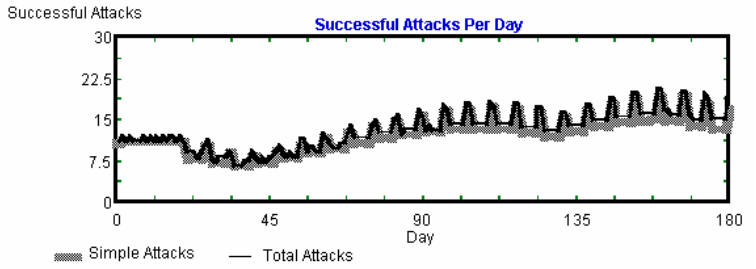

Fig. 7. Successful Attacks per Day, First *Escalation* Scenario

Certainly from Day 90 onwards, the system reaches a sort of equilibrium, as successful attacks hover around 13. This is a result of the matched opposing efforts of the organization and the attackers. Yet while the overall metric (i.e. successful attacks) does not change much, both efforts are ongoing. Figure 8 shows the efforts of the organization, in staff-hours per day dedicated to security tasks.

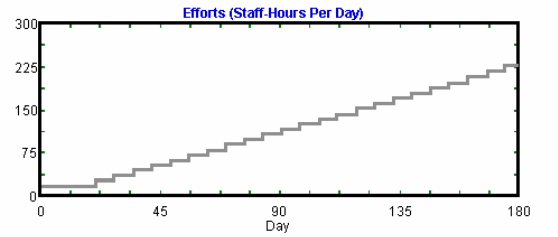

Fig. 8. Staff-Hours per Day, First *Escalation* Scenario

According to the rules and the specific values of the variables described above, the first decision to increase staff-hours occurs at Day 7, and is implemented fourteen days later; thus, the first increase is seen at Day 21. After that, increases can occur as often as every seven days: decisions to increase are made every seven days, and previous weeks' decisions will be implemented while waiting the fourteen days for this week's implementation. Overall, the organization's efforts grow, fairly linearly, up to approximately 220 staff-hours per day. Assuming eight-hour days, this translates into twenty-seven people, which is high but not unreasonable for a system of 200 machines. Of course, this growth is matched by the increase in both simple and sophisticated attacks. Figure 9 shows the attempted simple attacks.

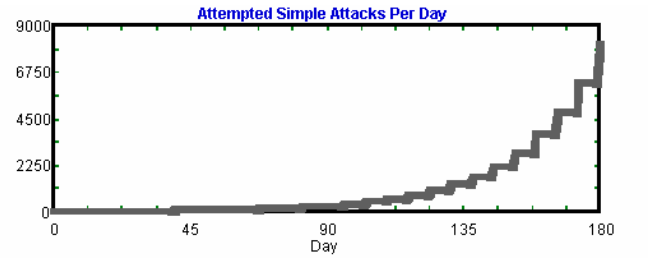

Fig. 9. Attempted Simple Attacks per Day, First *Escalation* Scenario

The number of attempted simple attacks can rise rather drastically, as this only requires that novice users unleash their automated processes against the system. This reaches almost 9,000 attempted attacks on the entire system per day, or 45 attempts, including viruses, per machine. While we stress the *behavioral trends* here much more than the specific numerical results, these numbers can be "reality-checked" against some empirical findings involving honeypots. [Dac04] observed attacks from 6,285 IP addresses over four months, averaging over two new attack sources per hour. Similarly, [Pou04a] observed 28,722 new attack sources over sixteen months. [Pou05] found 924 attack sources per day in Germany, and [Pou04b] mines a year of collected data and concludes with a very conservative estimate of 753 attack tools available to simple attackers. In light of these results, and considering that in our case, the organization has "begged for attacks" by advertising, our numbers seem fairly realistic (or in agreement with the existing empirical data.)

Figure 10 shows the daily average of attempted sophisticated attacks.

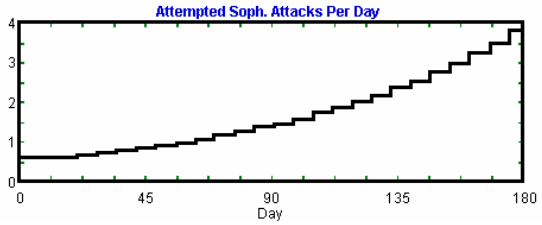

Fig. 10. Attempted Sophisticated Attacks per Day, First *Escalation* Scenario

The growth of attempted sophisticated attacks is much slower, as it requires higher human effort and expertise.

We also observe that the linear increase in the Organization's Efforts (i.e. staff-hours) can balance out the exponential increase in Attackers' Efforts (i.e. attempted

attacks). This is the case because in our model, a linear growth in countermeasure effectiveness leads to a lower percentage of successful attacks – an exponential decline.

Our simulation resulted in an overall relatively constant average number of successful attacks, an equilibrium of sorts between the results of the two striving parties (organization and attacker). Given these results, an organization may attempt to "beat" this escalation by increasing its efforts beyond the values given here; or it may consider cutting costs by reducing its efforts, if the results will be the same. We therefore ask how this equilibrium is affected if we modify the values representing the amount and frequency of increases in security efforts.

Firstly, we ask how much can be gained by the organization if it increases its efforts a bit more. In this scenario, when staff-hours increase, they increase not by $\vec{w}$, but by $1.1\,\vec{w}$ instead. Figure 11 shows the results.

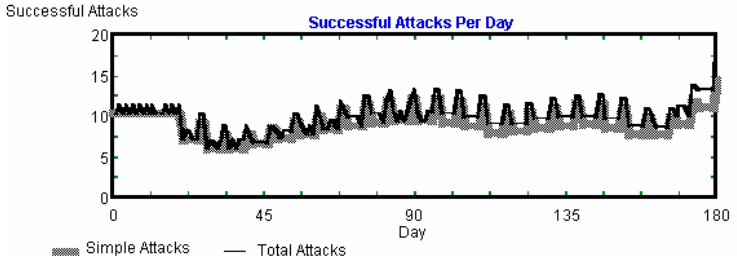

Fig. 11. Successful Attacks per Day: Results of 10% Increase in Efforts

Compared to Figure 7, Figure 11 has a similar overall shape, but the average number of successful attacks hovers around 10, versus the 13 of Figure 7. Thus, by increasing efforts by 10%, the organization can reduce its equilibrium by 3 successful attacks.

Secondly, we ask how much is lost if the organization does not increase its efforts as much. Now the increase in staff-hours is not $\bar{w}$, but $0.9\,\bar{w}$ instead. Figure 12 shows the results.

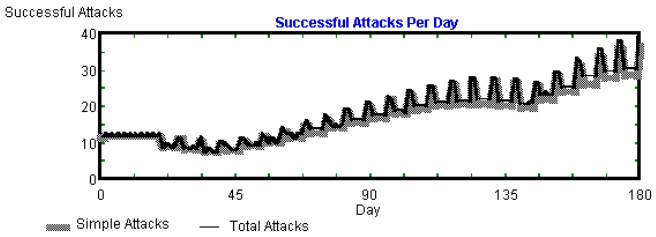

Fig. 12. Successful Attacks per Day: Results of 10% Decrease in Efforts

Suddenly, the equilibrium has risen to approximately 25 successful attacks. (It is not even clear whether an equilibrium exists by the end of the period, given the graph's steep climb from Day 140 onwards.) Thus, for a company considering changing its efforts, simulation here has shown that a small increase in efforts will not do much good, but a small decrease in efforts will cause much harm. This echoes [Sen90]'s discussion of "leverage", the large effects of small changes. A benefit of simulation is thus demonstrated.

Lastly, we test the sensitivity of this equilibrium by modifying a different value. Instead of the amount of the efforts' increase (i.e. $\bar{w}$), we change the frequency of increased efforts. x_1, the delay between increases (if increases are required), had been 7. We now change it to 6, and run the simulation. Intuitively, since the organization's reaction is more frequent, we expect the number of successful attacks to decrease. The results are shown in Figure 13.

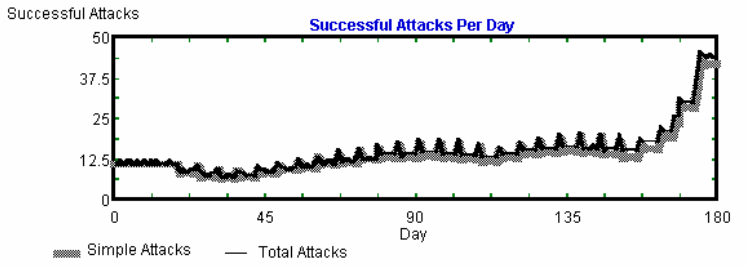

Fig. 13. Successful Attacks per Day: Results of More Frequent Efforts

Clearly, by Day 160, the equilibrium has been upset. The increase in attempted attacks is outpacing the increased efforts of the organization, and successful attacks begin to climb. This is due to the fact that the organization's more frequent efforts consisted of security spending followed by advertising, which attracted more frequent efforts of the attackers that it could not match. This example illustrates not only the utility of simulation for predicting the effects of small changes, but also the benefits of simulation in revealing unexpected behavior. This example also demonstrates a systems concept: sometimes the best way to survive an *Escalation* scenario is to not react as often, even if a reaction appears necessary. [Sen90] gives a case study of two manufacturers of a new design of stroller, both of which are making a respectable profit margin on their sales. Then the manufacturers entered *Escalation*, lowering their prices in an attempt to raise market share. Little time passed before both manufacturers no longer had a profit margin. The risk of reaction (reduced profit margin) had not been weighed against the risk of no reaction (reduced market share), and perhaps a slower reaction may have offered the greatest overall gain. Similarly, we have demonstrated the possibility that an organization can be out-escalated by its

opponent. Simulation thus grants the would-be entrant into *Escalation* the opportunity to pause and consider such outcomes.

Returning to our security scenario, we had described the increase in attacks as due to the organization's advertising. While some companies (e.g. search engines) cannot exist without high visibility, our results behoove an organization to consider the effects of its advertising and whether they outweigh the risk of additional attacks. Additionally, the automated "if-then" rule for advertising used here was to advertise anytime an increase in efforts is made. While the influence diagram of Figure 6 indicates that enough successful attacks will prevent further advertising, our if-then rules had assumed that point was not yet reached in the system, e.g. it only occurs when successful attacks reach 60 or higher.

Alternatively, a rule can be constructed which states, "Advertise only if successful attacks are below a certain threshold." Such a rule is included as part of the *Escalation* behavior in Chapter 4.

All of the above scenarios involved automated rules to govern the choices of both organization and attackers. As an alternative, the model also allows for a rule of "pause the simulation whenever a certain condition occurs." In our case, then, pauses may be configured, for example, whenever the successful attacks (simple, sophisticated, or sum of the two) exceed a threshold value. The simulation then pauses, and the end-user of the simulation may consider making changes such as introducing a new countermeasure or increasing staff-hours before resuming the simulation. The behavior over time of the aggregated attackers can also be paused and adjusted by hand; this feature may be useful to security researchers, but for a

company considering the impact of various choices that it could make, the attacker behavior is out of its hands and would thus presumably be represented by automated rules.

Lastly, here we assume that the organization is free to increase its efforts without any additional constraints. Practically, such increases may carry risks other than those of increased attacks; such risks are described by another archetype, *Limits to Growth*, and are described in the next chapter.

3.4 Other Instances of Escalation in Security

On the *Escalation* archetype, [Sen90] lists the international arms race as the most obvious example of *Escalation*, and [Hof05] specifies an "information technology security arms race." This arms race consists of advances in attack technology, which necessitate improvements in security technology. For example, [Hof05] argues that "with the advent of binary differs . . . patching is no longer a viable defense strategy", and instead advocates recent advances in Intrusion Prevention Systems. But this "race" develops over the course of a decade or longer: see [Dwa05] for a timeline from the 1980s to today. Given the vast unpredictability of long-term innovation, this is hardly something a single organization can simulate to aid its decision-making; we have thus chosen not to model it here.

[Sen90] also suggests a generic solution to this archetype's woes: often there can be an agreement to reverse the cycle, as each party agrees to simultaneously "ease off." While this may succeed in international politics (as it arguably did in détente),

the notion of “we’ll use less security technology if you agree to attack our computers less” is obviously not applicable in this case, particularly when the anonymous attacks, attackers, and motivations are myriad. This option is therefore not considered in our scenario.

For additional information on *Escalation* in computer security, please see [Ros06a], from which this chapter was excerpted.

Chapter 4: *Limits to Growth* and *Escalation* Archetypes, Combined

4.1 Limits to Growth Description

In the *Limits to Growth* archetype, a growing action is applied, which leads to increased gains or results. These gains encourage further growth, forming a reinforcing loop. However, the gains soon reach some natural limit, at which point the limiting process places downward pressure on further gains. Despite continued growth action, the gains will plateau and, in some cases, decline.

As an instance of this archetype in security, we consider the effects of security demands on an organization's computer staff of a fixed size. Suppose that an organization has a certain number of employees dedicated to various computer-related tasks such as technical support, hardware maintenance and upgrades, and security-related tasks such as monitoring a firewall or an IDS, or maintaining antivirus software. Initially, the organization pays modest attention to security, but then decides to make some investment in it. Whether the investment includes purchasing equipment (IDS, encryption or antivirus software, and the like), security training, overtime, or higher salaries for employees who focus on security, it always involves reassigning personnel to security. Encouraged by the noticeable gains in security, further investments lead to more reassignments of personnel to security. This continues to be a good strategy until insufficient personnel are available for non-security tasks. At this point, numerous non-security-related technical problems arise

in the computer system, forcing the security personnel to pause their efforts as these problems are addressed. Reassigning more employees to security (or demanding more of the current security employees) will bring no further gains; in fact, the additional technical problems as well as the support staff's decreased efficiency from facing demands it can not meet may result in a decline in gains. An influence diagram for this situation is shown in Figure 14.

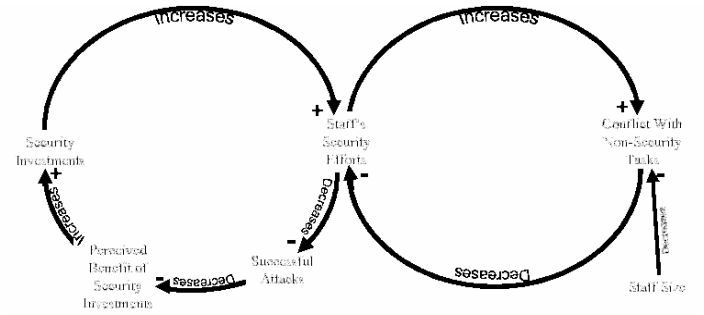

Fig. 14. Influence Diagram for *Limits to Growth*

Traversing the left side clockwise reads: "security investments increase the staff's security efforts, decreasing the number of successful attacks. More successful attacks would decrease the management's perceived benefit of security investments. More perceived benefits of investments leads to further investments." Reversing the double negative yields: "investments leading to efforts leading to gains in security (i.e. *less* successful attacks), increasing the perceived benefit of investments and therefore leading to further investments" – this is a reinforcing loop. The right-hand loop, however, describes how increasing the staff's security efforts can conflict with non-security-related tasks, due to a personnel shortage. As indicated by the upward arrow, this effect is decreased if the staff size is sufficiently large. Lastly, an increase of such

conflicts will cause problems that diminish the staff's security efforts. A balancing loop is thus formed, as security efforts will (unconsciously) decrease as long as the conflict of resources with non-security tasks is present. Given a constant number of attempted attacks, implementing this archetype should result in a continuous reduction of successful attacks (i.e. increase of gains for security investments) until insufficient personnel are available for other tasks; at that point, the number of successful attacks will cease to fall further, and may in fact begin to rise. [Wol03] includes this archetype in the category "Underachievement", as a reinforcing loop is desired for growth, but it is not successful.

The simulation model can incorporate this Limit to Growth with the following property: Some value p is the highest percentage of staff efforts that can optimally be reallocated to security with no ill effect. If total demand for security efforts exceed (p/100)*SysAdminCapacity, then the "effective" hours for security are given by the SysAdminCapacity, minus some constant k times the excess demand. In the simulation described below, we have used p = 23 and k = 1.2, believing these values to be a reasonable description of a typical system.

4.2 Combined Archetypes

While the use of an archetype can present a complex system in readily-grasped terms, a given scenario or story may not neatly fit into a given archetype. The general archetypes of [Sen90], [Sen94], and [Bra02] are unique only in that they have been frequently observed in diverse settings, and that they provide useful "building blocks" for other influence diagrams. For each given case study, [Sen94] recommends

beginning with the influence diagram of one easily-observed archetype (or simply a balancing or reinforcing loop), then "widening and deepening" the diagram by adding additional "loops" to describe the observed behavior. Thus, a combination of archetypes is often the simplest way to grasp a system's behavior when two or more different behavior patterns are exhibited simultaneously. (Such a combination, that of *Limits to Growth* with *Shifting the Burden*, can be found in [Sen90].)

Observe that both *Escalation* and *Limits to Growth* hinge on the organization's security investments and successful attacks; we thus connect their influence diagrams through these values. The resulting combined diagram is shown in Figure 15.

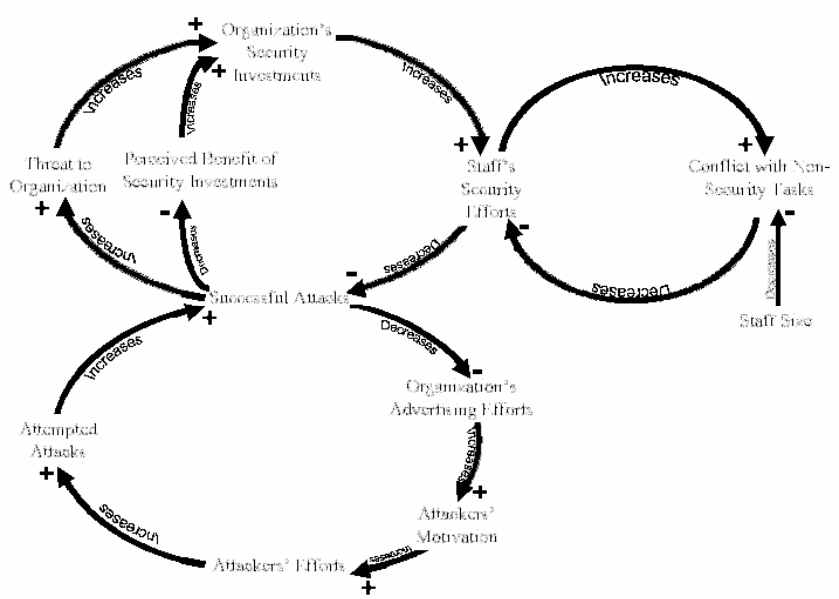

Fig. 15. Influence Diagram for Combined *Limits to Growth* and *Escalation*

Observe that the two influence diagrams largely address different issues, except for the upper-left-hand corner of Figure 15, which links successful attacks to security investments. While *Escalation* had assumed a "positive" effect (i.e. more successful

attacks increase the threat, increasing investments), *Limits to Growth* assumes a "negative" effect (successful attacks decrease perceived benefit, reducing investments).

In combining the two archetypes it becomes clear that both patterns may be true for different organizations with different cultures, or for different levels of management. Additionally, recall that an the influence diagram shows only "increases" and "decreases", but quantitatively some links may be stronger than others. Thus, both patterns may be present within a single organization; a visible shift from increases to decreases in investment, or vice versa, will occur at times when the weight of one pattern exceeds that of the other. For example, when the organization's management first invests in security, its perceived benefit is low, so further investments hinge on a reduction in attacks; later, security investments are believed an appropriate cure if successful attacks rise; finally, successful attacks may reach some upper limit at which point the management begins to lose its faith in investments and reduce them.

The overall trend of this combined archetype, when viewed in terms of successful attacks, will look as follows: a stable oscillation (due to *Escalation*) until security efforts exceed their optimal value (for the given staff size), followed by a rise in successful attacks (from *Limits to Growth*). At this point, several possibilities exist: the organization may continue (for a short duration) to advertise, leading to further attempted attacks; it may follow the "threat" pattern and push for more security investments; and/or it may follow the "perceived benefit" pattern and reduce security investments. Depending on these three options, the stable oscillation and rise in

successful attacks will be followed by either a leveling off or a rise in successful attacks; the former would occur if the organization halts both advertising and investments, keeping attempted attacks constant. The highest risk, leading to a significant increase in successful attacks, occurs if the organization continues advertising, raising the attempted attacks, as its continued investments cause more woes for its computer staff, further diminishing their effective efforts.

4.3 Simulation Setup

The behavior of the organization's management (which invests in security and demands staff-hours for it) and the aggregated attackers (who attempt the attacks) are then given by a series of rules (similar to those of Section 3.2), following the escalatory behavior described above. Here we demonstrate one possible outcome by assuming that the perceived benefit or "faith" in investments is held constant, and thus the decision regarding further investments is determined only by the threat to the organization. This decision is modeled by the following rule: *The simulation begins with an initial demand of w_0 staff-hours for security. Every x_1 days, { IF (Successful Simple Attacks > θ_1), THEN increase staff-hours demanded for "simple" security tasks by w. Additionally, IF (Successful Sophisticated Attacks > θ_2), THEN increase staff-hours demanded for "sophisticated" security tasks by v.} A delay of d_1 days is incurred for personnel reallocation.*

(The description of tasks as "simple" or "sophisticated", as well as the task-by-task composition of *w* and *v*, are unchanged from Section 3.2.)

The organization's advertising efforts are modeled by the following rule: *Every x_2 days, IF (Successful Simple Attacks < θ_3), THEN decide to advertise. A delay of d_2 days is incurred before the advertising occurs.*

Lastly, the aggregated attackers' response is modeled as follows: *The initial value of Simple Attacks Attempted / Day is a_0. Each day, IF (Advertising occurs), THEN a delay of d_3 days occurs as the word spreads and new attack tools are accumulated, where upon Simple Attacks Attempted / Day is increased by a%. The initial value of Sophisticated Attacks Attempted / Day is b_0. Each day, IF (Advertising occurs), THEN a delay of d_4 of days occurs as the word spreads and new attacks are engineered, whereupon Sophisticated Attacks Attempted / Day is increased by b%.*

We simulate a system of approximately 200 machines. We have chosen a period of six months (180 days) for our simulation. Successful attacks per day are used as our measure of "attackers' gains vs. organization's gains."

With these values in mind, we first simulated a "baseline scenario" characterized by the following values: $x_1 = 7$, $x_2 = 7$; $w_0 = 29.8$, $w = 9.1$, $v = 1.6$; $\theta_1 = 6$, $\theta_2 = 2$, $\theta_3 = 18$; $d_1 = 14$, $d_2 = 1$, $d_3 = 2$, $d_4 = 7$; $a_0 = 15$, $a = 26$; $b_0 = 0.6$, $b = 7$. These values describe, in our opinion, an organization's 180-day progression from minimal security efforts to full security efforts; a realistically aggressive advertising campaign; common delays for each action described; and a progression in terms of attack attempts from the minimal attack "noise" received by an inconspicuous organization to the high number that a prominent organization receives.

4.4 Results and Discussion

The results of this simulation are shown in Figure 16.

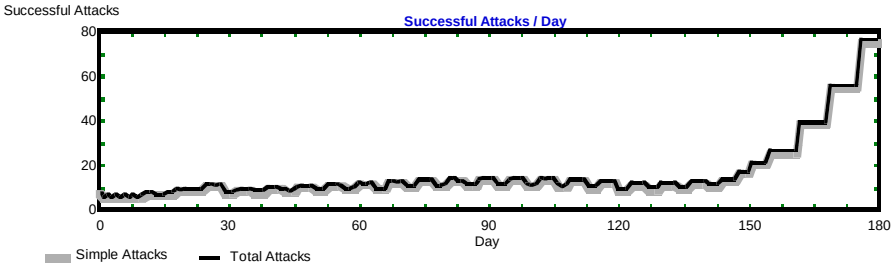

Fig. 16. Successful Attacks per Day, *Escalation* with *Limits to Growth*

The number of successful attacks seems to oscillate fairly stably until approximately Day 145, at which point it rises dramatically. Until Day 145, the number of successful attacks hovers at about 11, which is this system's equilibrium of escalation: security efforts, followed by advertising, followed by new attack attempts, followed by further security efforts. Around Day 145, however, the *Limits to Growth* archetype emerges: the demand for staff-hours exceeds the optimal load the staff can bear, the staff's performance deteriorates, and successful attacks rise. Note that successful attacks exceed θ_3=18, the organization's threshold for cessation of advertising, at approximately Day 155.

Correspondingly, the number of attempted attacks (simple plus sophisticated) is shown in Figure 17

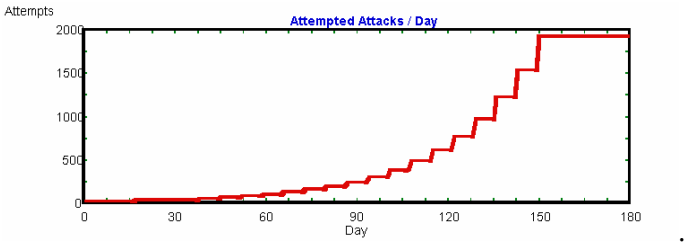

Fig. 17. Attempted Attacks per Day, *Escalation* with *Limits to Growth*

The number of attempted attacks escalates as often as every seven days (the organization's wait time between advertisements) until approximately Day 155. At that point, the company halts its advertising, and a constant 1925 attacks per day are attempted for the remainder of the period. Yet, returning to Figure 16, *successful* attacks are found to rise several times between Days 155 and 180. As the organization continues to reallocate staff to security and increase its demands on them, the personnel shortage for other tasks leads to more technical problems, sidetracking the increasingly overwhelmed security staff; attempted attacks thus become successful as the state of the countermeasures deteriorates and system vulnerability rises. *Limits to Growth* leads here to a decline in gains, not to a plateau.

By examining the behavior of the system, one can realize the problem of the increase in "successful attacks" around day 145. In response to this problem, the organization should take some action. Below we show how the use of simulation can support decisions regarding what action best fits the goals and context of a given organization.

Firstly, as our system was described, the increase in attempted attacks came not directly as a result of increased security efforts, but as a result of the organization's

advertising. While this may not be the case for all organizations, certainly any organization considering advertising must weigh potential benefits (such as increased clientele) against the possibility of (and its preparedness for) *Escalation*.

Secondly, even when *Escalation* is called for, it may be wise to escalate less strongly. The organization's rule for increasing security efforts was given as: "Every x_1 days, if successful (simple, sophisticated) attacks are greater than (θ_1, θ_2), increase efforts by (w, v)." Increasing the period x_1 (i.e. reducing the frequency of possible escalation), raising the thresholds θ_1 and θ_2 (reducing the frequency of when escalation is called for), and/or reducing w and v (the quantities of escalation when it is employed) are all possible solutions. When a threat is perceived, the effect of reaction must be weighed against the risk of no action, and sometimes the greatest overall gain is achieved by a slower or weaker reaction. Similarly, when we turn to *Limits to Growth*, it is noted that if the limits will not be (or cannot be, as in [Mar03]) removed, then reducing the growth action will delay the onset of the limiting factors, as well as slowing the deterioration of growth once the limits manifest themselves. A reduction solution thus heeds both archetypes. To see which of these three reductions is most effective here, all three were simulated: reducing the frequency of increased efforts, raising the threshold for increased efforts, and reducing the quantity of efforts. Experimenting with each solution individually as well as combined with others, we found that our system responded most favorably to simply reducing the quantity of escalation *w* by 30%: each time the organization decides to increase its security efforts, it does so by 6.4 staff-hours, as opposed to the 9.1 of the baseline case. The results are shown in Figure 18.

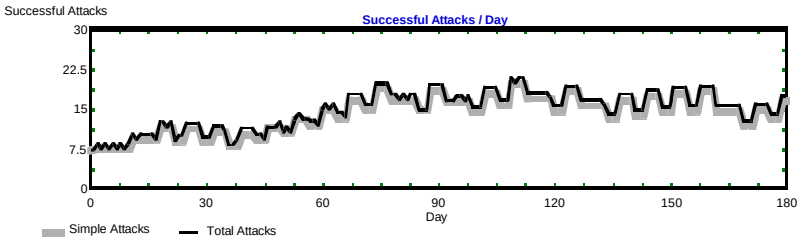

Fig. 18. Results of Reduced Escalation (Successful Attacks per Day)

Note that the equilibrium number of successful attacks has risen to about 15 (as opposed to the 11 in Figure 16), but there is no dramatic climb in successful attacks by Day 180. The weaker reaction has not pushed demands on the staff beyond their point of optimality in these six months. This approach can thus be thought of as "partly losing *Escalation*, but winning *Limits to Growth*." Note as well that the number of successful attacks reaches θ_3=18, the advertising threshold, several times, leading to less advertising and thus less attempted attacks. Integrating Figures 16 and 18, we find that the number of cumulative successful attacks is less for reduced escalation (~2650) than for full escalation (~2800). Given a particular organization's structure, goals, and priorities, the above tradeoffs (equilibrium number of attacks, rise in attacks, advertising opportunities, cumulative attacks) should be considered to find whether reduced escalation is more in its interest than full escalation.

Thirdly, a solution commonly found for *Limits to Growth* is to cease the growth action, and instead concentrate on removing the limiting condition. In our case, this would translate into hiring additional support staff. [Sen90] stresses the concept of "leverage", i.e. an organization's efforts will yield maximal gains if it carefully chooses where and when to apply those efforts. While hiring too early is prohibitively expensive, if hiring is delayed too much, the limit will set in and deterioration of

gains will begin. Additionally, the stronger the limit has become, the harder it is to remove it; in our case, once the support staff is overwhelmed with demands, it will not have time to introduce new hires to the intricacies of the computer system. Thus, the point of highest leverage for hiring is *when it will take effect just before the demands on personnel exceed their optimal load.* This requires great prediction skills on the part of the manager, including a sense of "feedback" regarding the support staff's load. Otherwise, the best strategy is to hire *as soon as possible once a decline in gains is visible.* This also requires the manager to realize that indeed, gains have diminished since the optimal personnel load was reached. As opposed to the previous strategies, which are executed before-the-fact, this strategy describes how an organization might now respond to problems. Following full escalation, Figure 16 showed a rise in attacks around Day 145. Figure 19 shows the results if the organization responds rapidly and additional personnel are available as of Day 155.

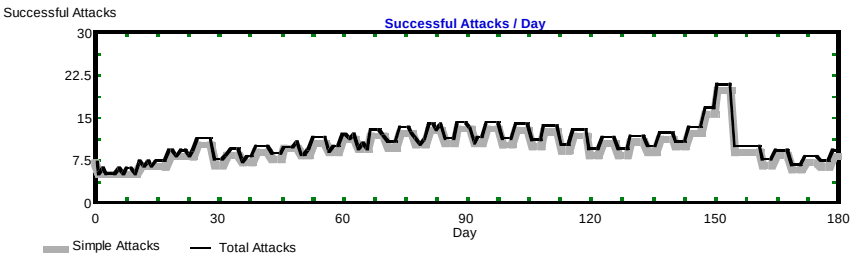

Fig. 19. Full Escalation, with Hiring at Day 155

The oscillations and steep rise occur as in Figure 16, followed by a steep drop in attacks due to the hiring. Integrating, we find a total of approximately 1,880 successful attacks, far less than in full or reduced escalation. (Of course, this benefit

comes with the cost of hiring.) The sooner the hiring, the less of the peak around Day 150; the longer the wait to hire, the greater the peak.

Figures 16, 18, and 19 have shown the results of the three above scenarios in terms of the number of successful attacks. An organization must also consider factors such as labor costs, and thus Figure 20 displays the cumulative staff-hours employed for each scenario: "baseline" (full escalation, without hiring), "reduced escalation", and "escalation with hiring." Note that the efforts of "baseline" and "hiring" will coincide until Day 155, at which point the curve for "hiring" will grow more steeply.

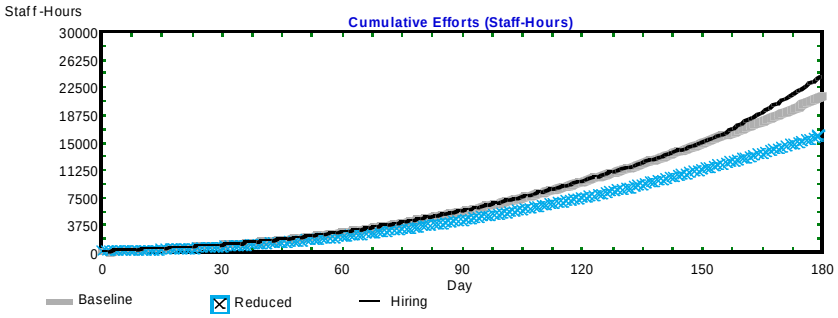

Fig. 20. Cumulative Staff-Hours for Each Scenario

Thus, given the number of successful attacks and number of staff-hours employed, both per-day and cumulatively, an organization can consider its best options as it encounters this combination of archetypes.

For additional information on the combination of *Escalation* and *Limits to Growth* as it occurs in security, please see [Ros06c], from which this chapter was excerpted.

Chapter 5: Related Work

5.1 System Dynamics and Archetypes

System dynamics thinking is introduced in [For61]. An introduction to archetypes can be found in [Sen90], with added details and recommendations in [Sen94], while [Bra02] extends this work to list ten different archetypes. [Wol03] argues that all existing archetypes can be included in one of four "core generic" archetype categories such as "Underachievement" or "Out-of-Control"; however, [Wol03] acknowledges that the more-specific, more-familiar ten archetypes (such as "*Escalation*" or "*Limits to Growth*") are more rapidly applied to real systems, and we have thus used them here. [Mar03] applies systems thinking and archetypes to safety engineering. Some archetypes found in safety are clearly those seen elsewhere (such as "Eroding Goals"), but others seem unique to safety. (This is the case partly because safety measures can be a victim of their own success – when no accidents occur, there can be pressure to reduce safety measures.) For now we have focused on the more-common archetypes of [Bra02] regarding security, but future work may find that new archetypes apply to safety as well.

5.2 Sources of Data

Empirical data regarding computer security are still fairly rare as of now. Anecdotes detailing attacks and their responses, such as [Gib02], are very illustrative of the attacker/defender interaction, but few such anecdotes have been published.

Some information regarding what is general practice in the security world today can be found in [Gor05a], a survey of several hundred organizations. For example, our model includes IDSs but not biometrics because the former is found to be significantly more prevalent in real life today.

Most data on attacks are gathered from analyzing “honeypots” or “honeynets”, systems designed to be attacked. Such studies include our own laboratory’s [Pan05], as well as [Dac04], [Pou04a], [Pou04b], and [Pou05].

Hypothesized attacker behavior is described in [Jon97], based on empirical findings from controlled attack experiments. This focuses on the behavior of the individual attacker, while more data are needed on the aggregated effects of multiple attackers.

To help meet the dearth of empirical data regarding security, nine teams are collaborating on the projects DETER and EMIST [Baj04]. DETER involves building a massive (currently approximately 200 machines, intended to reach 1000 machines) “researcher- and vendor-neutral” network testbed for emulating various types of attacks, countermeasures, and network topologies. Meanwhile, the EMIST project seeks to formalize methodologies for measuring these effects. Combined, these projects should provide a wealth of useful, unbiased, and well-accepted emulated attack data. Both studies will enrich our model with quantifiable values, e.g. honeynet findings might show that 20 buffer-overflow attacks of a certain type are attempted each day, and the DETER/EMIST findings would tell us that the attack will succeed 80% of the time if the network has Topology A but only 60% of the time with Topology B.

Regarding user factors, [Lar03a] uses surveys to understand Internet usage, and [Lar03b] conducts studies with test websites to investigate users’ privacy behaviors online. The authors of these papers have indicated that their future work will analyze

user behavior regarding network security, which should be applicable to our user model.

Comment [SNR1]: This better?

5.3 Economics and Security

[Cam03] considers the effects of public disclosure regarding security breaches on a company's stock prices. [Gor02], [Gor05b], and [Bod05] all use economic analysis in determining how much security investment is worthwhile for a company, given its priorities; however, details are not provided as to what should be done specifically with the investments. This provides the connection point to our model.

Economic requirements are also used to lead to assumptions or specifications for related computer security, e.g. determining the subjective cost and total welfare regarding network routing [Fei05] or requirements on trusted platforms placed by digital rights management [Ber04], [Ber05].

5.4 Other Modeling Approaches in Security

One approach in security has been to probabilistically quantify an attacker's behavior and its impact on a system's ability to provide certain security-related properties. Attempts have been made to build models that take into account both the attacker and the system being validated. A general model of an intrusion-tolerant system is proposed in [Gon01] to describe security exploits by considering attack impacts; the system state is represented in terms of failure-causing events. [Jha01] proposes a combination of state-level modeling, formal logic, and Bayesian analysis

quantify system survivability. Finally, Ortalo et al. [Ort99] propose modeling known vulnerabilities in a system using a "privilege graph". By combining a privilege graph with simple assumptions concerning an attacker's behavior, the authors then obtain an "attack state graph." Parameter values for such a graph have been obtained experimentally; once obtained, an attack state graph can be analyzed using standard Markov techniques to obtain several probabilistic measures of security. [Ste04] uses a probabilistic model for validating an intrusion-tolerant system that combined intrusion tolerance and security, allowing the designers to make choices that maximize the intrusion tolerance before they implement the system. Compared to these models, the model presented here is more generic in its inclusion of other human elements such as users and system administrators. Additonally, other than [Ort99] which uses data collected empirically to assess some of the parameters values in the model, the other ones are not developed to easily be linked to empirical data.

Cyberciege ([Nav06], [Irv05]), developed by the Naval Postgraduate School, is a computer game with a very engaging user interface and virtual world, intended for training students to understand security engineering. Cyberciege focuses on detailed access control, user-by-user, for a small number of users. Each piece of hardware is hand-selected from a list of fictional brands (e.g. "BitFlipper router"), and physical security measures are implemented on a user-by-user basis. The determination of whether an attack succeeds is by comparing asset desirability and how well standard procedures have been followed. Cyberciege's level of detail models the role of an individual security officer who might oversee a dozen computers at most, while our

model abstracts one level higher, to the manager who oversees several hundred machines.

In a similar vein, Fred Cohen & Associates offer a security simulator [Coh06] on their website (http://all.net/games/index.html). Fully described in [Coh99], this simulator gives examples of how a single attack of varying sophistication might succeed against different computers with different countermeasures. The defender strength, i.e. to what degree the defender does the right thing, is specified as a percentage by the user before running the simulator. If an attack succeeds, the dollar loss due to the attack is estimated based on the attacker profile, e.g. how much will a successful attack by a private investigator cost? Our approach attempts to add in more empirical data, as described in Section 5.2. Additionally, our work extends the "defender strength" idea by allowing for strengths of each countermeasure: a system may have a 90% effective firewall but only a 70% effective IDS. Furthermore, rather than specify a value for defender strength, the user of our model inputs managerial decisions such as how much effort is allocated to which security tasks and how skilled the staff is – the model then uses these inputs to determine the resulting defense strength for each countermeasure.

Chapter 6: Conclusions and Future Work

6.1 Conclusions

The archetype and results of simulation execution presented here show the value of systems dynamics modeling for enterprise security. The evolution over time of two slightly different "what-if" scenarios may result in very different pictures, reinforcing the value of simulation. Systems thinking, combined with simulation, can assist an organization in placing its efforts in the places that will give the most "leverage" to their goals, and in diagnosing and solving problems. This approach thus leads to a more enlightened weighing of costs vs. benefits for the proposed decisions that an organization might make.

System dynamics simulation is also an intuitive and powerful tool for understanding computer security, as well as for training professionals. In time, our model will mesh with much other research currently being done by others, leading to gains in a wide variety of directions.

6.2 Future Work

A great deal of future work remains as well, including:

- "Deepening" the simulation model with more detail, e.g. where linear rates had been assumed, perhaps logarithmic or exponential would be more accurate. The documentation of the simulation model already reveals several ways it can be deepened.

- "Broadening" the model to include such factors as:
    - User details describing their interaction with the security policy.
    - Asset properties. Currently we only show successful attacks; future work can link this to system availability, confidentiality, and integrity.
    - Internal attacks. Currently it is assumed that the firewall is X% effective against all simple attacks, for example, which assumes that all simple attacks come from outside the firewall.
- Obtaining additional empirical data for use as parameters in the simulation model. Sources for such data, including work from our own research group, are described in Section 5.2.
- Modeling other instances of the above archetypes, modeling other archetypes, other combinations of archetypes, and looking for new archetypes. Appendix I gives a few ideas for modeling other archetypes.
- Documenting real-world case studies in security, using archetypes to explain the situations, and using simulation to suggest improvements. (For example, [Sen94] first describes the story of an airline's failure, applies archetypes to describe it, and then builds a simulator through which it is shown, for example, that had the airline not cut its ticket prices quite so steeply, it would not have gone bankrupt.) We have already begun interviewing one system administrator and documenting his case study, but obtaining the necessary details, applying archetypes, and simulating the case study are all left for future work.

Appendix I: Archetypes

Here we briefly describe each of the ten archetypes of [Bra02], giving one possible example from security.

Shifting the Burden, or **Symptomatic Fixes**. We witness a problem symptom, and rather than think about the root cause, we try to fix the symptom. Doing so distracts us from the actual cause of the problem, or masks the symptoms so it's harder to diagnose the problem. Suppose a system is continually falling victim to successful "script-kiddy" attacks (symptom). The company may install an I.D.S. to catch the attacks (symptomatic fix), when in reality the attacks wouldn't make it into the system if the company had a good firewall, and wouldn't succeed if they kept their vulnerabilities down. (Fundamental fixes.)

Fixes that Fail. Here, the attempted fix actually worsens the underlying problem in time. The newly-installed I.D.S may have a high false-alarm rate and require a great deal of the sysadmin's attention. The sysadmin is now too busy to attend to other duties (such as addressing vulnerabilities), so the number of successful attacks actually increases.

Success to the Successful. There is a tendency to believe that if putting some money into Approach A yields good results, then putting more money into Approach A (and ignoring Approach B) will further improve results. For example: for an investment of $100, a Host-Vulnerability-Scanner will yield more improvements than an IDS. But continued investment into the Host-Vuln-Scanner (diverting funds from the IDS) will not help much if at all.

Limits to Growth. Increased efforts and investments produce increased results, until the system reaches its natural limit. At that point, results will either plateau or decline. For example, given an inexperienced sysadmin staff of a fixed size, training them will result in significant gains to the network's security. But eventually, their size (rather than skill) becomes the limiting factor, so further training will accomplish nothing.

Attractiveness Principle. Increased efforts are no longer producing results, with two different limits fighting growth. The manager must decide which limit to address first/more. Suppose we have a simultaneous investment in both more/better sysadmin staff, and some technology (maybe a firewall). At some point, the Return on Investment will drop; at that point, we must decide which factor is more of a limiting one.

Growth and Underinvestment. A successful approach may initially seem to fail if it wasn't given proper investment/support/capacity. For example, a company may double its system size; if the SysAdmin size (which is the capacity in our case) is kept constant, overall performance will drop. If, instead, the SysAdmin size is properly increased, the company will see a gain.

Eroding Goals. If a goal is not immediately met, it can be tempting to reduce the initial goal. A manager may try for an Availability (or confidentiality, etc.) Level of 3, find that the expenses next month are too high, so s/he drops the goal to Level 2. The next month, the company is hit with a massive attack, causing more loss than had it held the course at Level 3. (Another example would be, "We want an IDS that

catches 100% of all attacks. What, that gives too many false alarms? Okay, maybe 90%. Still too many alarms? Okay, maybe set it to 80%.")

Escalation. Party A puts in more efforts, yielding more results; this threatens Party B, who does likewise, and so on. (The U.S./ U.S.S.R. arms race during the Cold War is a good example.) If a company increases its security efforts and publicizes how secure it is, or otherwise makes itself more of an attractive target, it will receive more sophisticated attacks, which will require more security investments, and so on.

Accidental Adversaries. Two parties initially agree towards cooperation, but then Party A perceives an offense (often unintentional) from Party B; it then retaliates, and the situation escalates from there. An example here would be the SysAdmin and User, who agree they want the company to succeed, but then the user accidentally breaches the security policy, leading the SysAdmin to impose a harsher security policy and other enforcement measures. The user (or another user) may become annoyed and retaliate.

Tragedy of the Commons. If two efforts independently consume a common resource without respecting one another, both will see reduced gains as the resource runs out. In our case, if a company decides to invest more in IDS as well as Host-Checking-Tool, but maintains the size of its SysAdmin (which is the "common" resource consumed), both will not yield full results.

Appendix II: Model Screenshots

The basic building blocks for continuous modeling in the Extend simulation environment include holding tanks, constants, and equations, to name a few.

As one example, we show a simplified version of how the antivirus software effectiveness is modeled. Suppose that this system needs its virus definitions updated on a daily basis; if so, an antivirus that has been totally neglected for too long of a period will become close to useless, as it fails to catch the majority of viruses circulating the Internet today. Thus, antivirus effectiveness is reduced each day by some average "daily loss rate" which describes the occurrence of new viruses, and increased each day by the number of staff-hours updating its definitions (or otherwise maintaining it) that day. The effectiveness is then measured on a 0-to-1 scale and output. This is modeled in Figure 21.

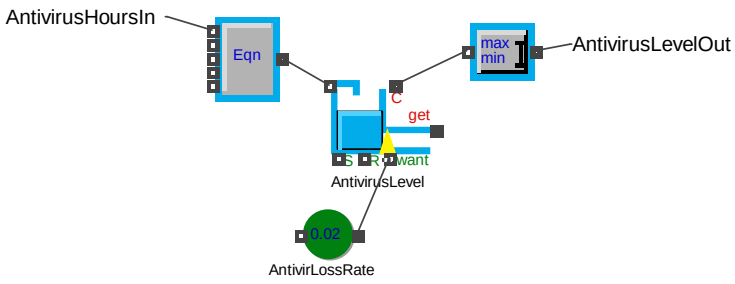

Fig. 21. Sample Screenshot of Holding Tank, Equation, and Constant Blocks

Notice the number of staff-hours in, subject to some function, the holding tank for the daily antivirus effectiveness, and the daily loss rate. Each day, the contents of the

tank are given, limited to the range between 0 and 1, and output as today's "antivirus level."

To allow for greater abstraction, all of the above blocks can be inserted into a custom-built "hierarchical blocks", such as the one shown in Figure 22.

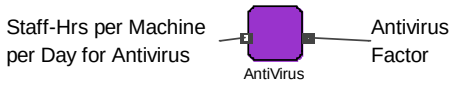

Fig. 22. Sample Hierarchical Block, Antivirus

Here we see only the input and the output; the remaining holding tank, equation block, etc. are all hidden inside the hierarchical block.

In our model, a certain number of attacks of a given sophistication level are attempted each day. Depending on the effectiveness of the various countermeasures and the system's vulnerabilities, a certain number succeed. Another hierarchical block, which performs this evaluation, is shown in Figure 23.

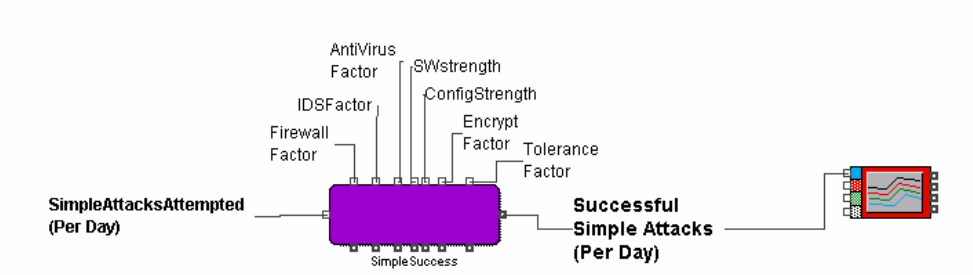

Fig. 23. Sample Hierarchical Block, Simple Attack Success

The block on the far-right of Figure 23 is an output plotter, used to generate many of the figures presented in this thesis.

The model has a great deal of constant parameters; for example, the antivirus daily loss rate of Figure 21. These are listed in a spreadsheet such as the one displayed in Figure 24.

ConfigVulnLevel, loss if ignored	.05, linear
", staff-hrs needed to maintain	1.5 / machine
NetVulns, loss if ignored	.04, linear
", staff-hrs needed to maintain	0.6 / machine
AppVulns, loss if ignored	.004, linear
", staff-hrs needed to maintain	0.13 / machine
AppVuln, loss from new S/W	0.8
ToleranceLevel, loss if ignored	.1, linear
", staff-hrs needed to maintain	0.67 / machine
EncryptionLevel, loss if ignored	.001, linear
", staff-hrs needed to maintain	0.067 / machine
Antiviruslevel, loss if ignored	.02, linear
", staff-hrs needed to maintain	0.4 / machine
FirewallLevel, loss if ignored	.033, linear
", staff-hrs needed to maintain	0.66 / machine
IDSLevel, loss if ignored	.05, linear
", staff-hrs needed to maintain	2 / machine

Fig. 24. Sample from Spreadsheet with Parameter Values

Lastly, while certain parameter values (such as antivirus loss rate) reflect the reality of the system, others (such as machine size, staff size and the presence of countermeasures) reflect decisions that a manager might make. To allow for easy "what-if" simulation, these parameters were extracted to a user-friendly Graphical User Interface, such as the one seen in Figure 25.

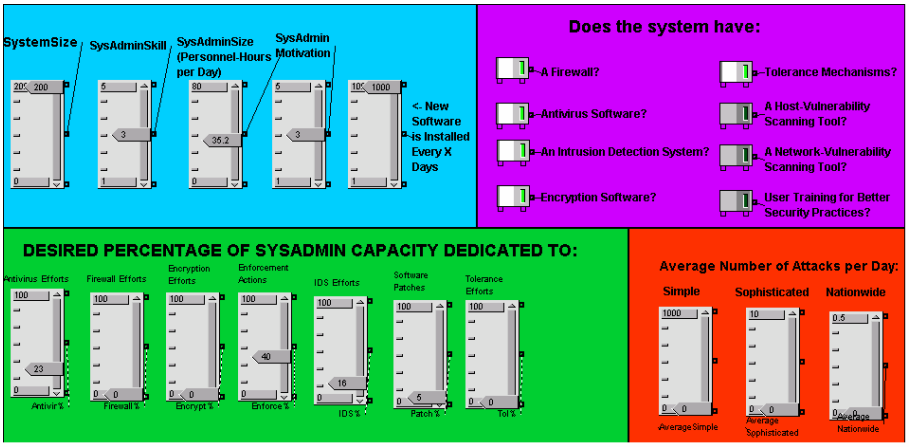

Fig. 25. Graphical User Interface Screenshot

Appendix III: Model Documentation

OVERVIEW: The end-user of the model sets several sliders and switches to describe the *system, countermeasures, allocation of sysadmin to various tasks,* and *attacks.* The end-user can then see the *costs* of this configuration. A certain number of attacks are then attempted on the system each day; given the details of the system and its countermeasures, the end-user can see how many of those attacks succeed, or how many were blocked by a given countermeasure. The end-user can also track the effectiveness of a given countermeasure over time.

SYSTEM INPUTS: These sliders describe the system and staff, and are listed in Table II.

Table II: Slider Inputs for the Model Graphical User Interface

Name in Model	**Type**	**Meaning**
SystemSize	Slider, 0-80	(# of machines)
SysAdminSize (Personnel-Hours per Day)	Slider, 0-80	Personnel-hours (or "man-hours") of System Administration and Security Officer staff employed per day. A SysAdminSize of 40 describes 5 people working 8 hrs/day each day, or 10 people working 4 hrs/day, etc.
SysAdminSkill	Slider, 1-5	Average overall skill of the System Administration and Security Officers Staff. The 1-5 scale is ours.
SysAdmin Motivation	Slider, 1-5	How motivated the SysAdmin staff is to protect the system; we impose a 1-5 scale.
New Software is Installed Every X Days	Slider, 1-1000	Interval (in days) between installation of new software (which contains new vulnerabilities). Patches are not included here.

Further descriptions of the system, e.g. Windows vs. Linux, would be a critical step in adding detail to the model; it will hopefully be considered in a future implementation.

COUNTERMEASURE INPUTS: We include several common countermeasures. In the 2004 CSI/FBI Computer Crime and Security Survey of 494 U.S. corporations, universities, government bodies, etc., the most common security technologies used (Fig. 16), by percentage of respondents, were: Antivirus software (99%); Firewalls (98%); Server-based access control lists (71%); Intrusion detection (68%); Encryption for data in transit (64%). We view the access control lists as part of the "SysAdmin's Enforcement Actions" and not a separate technology *per se*, as it is built into most operating systems today. For simplicity's sake, we chose to include both data-in-transit encryption and file-encryption as "encryption software."

A significant countermeasure not described directly in the CSI/FBI survey is the emerging field of attack *tolerance* (as opposed to prevention or detection). This could include designs for graceful degradation under attack; redundancy and diversity (in some cases); and other technologies allowing the system to succeed despite the attack. We thus include a countermeasure entitled "tolerance mechanisms."

Additionally, as 70% of the survey respondents (Fig. 17 in the FBI survey) identified some type of network security training for their users as important, we have included "user training for better security practices."

Lastly, we have included vulnerability-scanning tools which can assist the system administrator in finding vulnerabilities to fix. These include host-configuration vulnerability scanning tools, such as FERRET; and network-vulnerability scanning tools, such as NESSUS.

For all of the above countermeasures, we presently assume that they are either present in full strength, or not at all. (They're controlled by binary switches.) Future implementations of the model may modify this. The countermeasures are given in Table III.

COST/EXPENSE EQUATIONS & OUTPUTS: Given the above descriptions, we can now compute the system's expenses. (For now, we simply tally the number of successful attacks, rather than describing the monetary loss they cause the company; this too will hopefully be improved in a future model.)

Table III: Countermeasures Included in the Model

Name in Model	Type	Meaning
A Firewall?	Y/N Switch	"1" if the system has a firewall; "0" if it doesn't.
Antivirus?	Y/N Switch	"1" if every system has antivirus software installed.
An Intrusion Detection System?	Y/N Switch	"1" if an Intrusion Detection System is present.
Encryption Software?	Y/N Switch	"1" if encryption software is installed.
Tolerance Mechanisms?	Y/N Switch	"1" if tolerance mechanisms are present.
A Host-Vulnerability Scanning Tool?	Y/N Switch	"1" if the sysadmin uses a tool such as FERRET to check host-configuration vulnerabilities.
A Network-Vulnerability Scanning Tool?	Y/N Switch	"1" if the sysadmin uses a tool such as NESSUS to check for network vulnerabilities.
User Training for Better Security Practices?	Y/N Switch	"1" if the users are trained regarding network security.

Expenses reflect all the money spent on the system over the duration of the simulation (usually ~100 days). *StaffCost* is the cost per day of employing the sysadmin staff. *PurchaseCost* is the cost to purchase the various countermeasures, which we assume is a one-time payment. We then have:

Expenses = *(StaffCost* * Time) + *PurchaseCost.*

($) = *($/day) * (days)* + *($)*

In Extend terms, *Expenses* is an accumulating tank; *StaffCost* is the input, and *PurchaseCost* is the initial level.

StaffCost = STAFFCOSTPERHOUR * *SysAdminSize.*

($) = *($/hr) * (personnel-hours)*

The cost of employing the sysadmin staff per day. We assume an average cost of $35 per personnel-hour.

For *PurchaseCost,* we assume that *Tolerance Measures, Encryption Software,* and *an Antivirus* must be purchased for every machine in the system to be effective. (The effects of installing an antivirus on only half, 1/3, etc. of the machines would be another interesting question for future work.)

Per-system purchase costs = *SystemSize* *

{ (Tolerance Measures?)*TOLCOST + (Encryption Software?)*ENCRYPTCOST

+ (Antivirus?)* ANTIVIRUSCOST }.

($) = (# machines) * Σ{(1/0)*($/machine)}

We simply assume for now that tolerance measures cost $300/system. For encryption software, PGP is very commonly used (try Google searches for "encryption software" and the like); the most basic version of PGP Desktop Professional 9 costs $200; we have used the value $220 to allow for a few more features. For the antivirus, Norton Antivirus, one of the most popular products on the market, costs $40 /machine in the 5-user pack. *(Sources: manufacturer's websites.)*

We do not include the host-configuration or network-vulnerability scanning tools in costs or expenses, as the most popular products used (i.e. FERRET and NESSUS) are available for free. The remaining two *PurchaseCost* items are the firewall and IDS, whose cost is independent of the size of the system behind them.

PurchaseCost = per-system purchase costs + (*A Firewall?*)* FIREWALLCOST + (*An IDS?*)* (IDSCost).

($) = *($)* +*(1/0)*$* + *(1/0)* $.*

We assume that a high-quality firewall costs $10,000, given Dr. Cukier's experience with proprietary firewalls. For the IDS cost, we take the price of the Cisco 4250, which is $30,000.

SYSADMIN ALLOCATION: We describe the SysAdmin staff's "capacity" to maintain and protect the system as a function of its size, skill, and motivation:

TotalSysAdminCapacity = *SysAdminSize* * *SysAdminMotivation* * ln(*SysAdminSkill* + 1).

(Note that TotalSysAdminCapacity is measured in pseudo-personnel hours, as it can be increased by motivation and skill.) (The logarithm is used to reflect the

phenomenon that beyond a certain point, additional training accomplishes very little. We use (skill+1) so that a skill level of 1, the lowest, doesn't result in an ln(1) = 0 term.)

The end-user then decides what percentage of the *TotalSysAdminCapacity* should be dedicated to what task, using the sliders in the green box. The sysadmin needs to spend time and attention to deal with any given countermeasure (or its side effects!). We refer to these as "countermeasure efforts." Obviously, more efforts are needed during deployment than afterwards, but for now, we simply describe "efforts-per-day." (One approach would be to consider an average effort over the product's lifetime, including its deployment, but this again is for future work.) The order of the various efforts is consistent with that of the model, but it has no particular significance.

"Antivirus Efforts" consist primarily of keeping all of the antivirus definitions up-to-date. The percentage of *TotalSyadminCapacity* dedicated to Antivirus Efforts is called *Antivir%*.

"Firewall Efforts" consist of tasks needed to maintain the firewall, primarily through applying new patches as firewall vulnerabilities are discovered. *(Firewall%).*

"IDS Efforts" consist of maintaining the intrusion detection system, mostly by downloading new signatures. *(IDS%).*

"Encryption Efforts" consist of updating and maintaining the encryption software (quite possibly including helping users who run into difficulty using it). *(Encrypt%).*

"Enforcement Actions" include setting proper access control; monitoring the system for noticeable oddities; and developing and enforcing a security policy for the

users. For example, if a user tried using a "weak" (i.e. easily guessed) password such as "joe", a vigilant sysadmin would prevent him from doing so. (*Enforce%).*

"Software Patches" reflects the time spent per day on finding and installing patches for newly-discovered vulnerabilities in any of the system's netware, operating systems, or applications. *(Patch%).*

"Tolerance Efforts" depend on the particular tolerance measure; some measures are relatively low-maintenance (e.g. if graceful degradation has been built-in, then no further action is needed), but some are high-maintenance (e.g. if the system has a backup web server that runs a different operating system, the backup server has to be maintained as well). *(Tol%).*

"Addressing Alarms" refers to the alarms raised by the firewall and IDS; sometimes these were in fact attacks, but often they were legitimate actions. A good sysadmin should sort through these. *(Alarm%).* In the new versions, we've gotten rid of "addressing alarms" as its own task; it's now included in either "IDS Efforts" or "Firewall Efforts."

Comment [i2]: Page: 1
I suggest dropping the word "false" when we talk about alarms

The various desired percentages, as well as the *TotalSysAdminCapacity* and *SystemSize*, are input into the *HoursForTasks* block. The outputs of this block describe how many SysAdmin pseudo-personnel-hours (or more precisely, skill-motivation-personnel-hours each day) are actually allocated to each task.

If the various desired percentages (inputs) add up to 100 or less, then all of the desired demands can be met, and the process is simple:

Hours allocated to Firewall = (*Firewall%* / 100) * *TotalSysAdminCapacity.*

Hours allocated to IDS = (*IDS%* / 100) * *TotalSysAdminCapacity.*

The firewall and IDS are independent of the size of system behind them. (Or are they? Once we include analyzing alarms in Firewall Hours & IDS Hours, well, the bigger the system, the more alarms likely. In the new paper, I assumed staff-hours per machine for these as well.) For the other efforts, however, we must factor in the system size; after all, to spend a total of two hours per day on updating antivirus definitions for a single computer is certainly sufficient; for a thousand computers, it probably won't be. We thus talk of "hours allocated per system [per day]." Note that for now, we assume that doubling the system size will simply halve the personnel-hours available for a given task; in reality, larger system sizes tend to come with mechanisms for better management, so we might in the future consider a factor such as log(systemsize). For the moment, though, we've kept the divisor linear. The following hours are per-machine:

Hours allocated to Antivirus = (*Antivir%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Encryption = (*Encrypt%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Enforcement Actions = (*Enforce%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Software Patches = (*Patch%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Tolerance = (*Tol%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to (False) Alarms = (*Alarm%* / 100) * *TotalSysAdminCapacity / SystemSize.*

(A larger system will generate more alarms, and thus needs more attention.)

IF, however, the end-user specifies a series of percentages with the sliders that sum to >100%, not all of the desired hours will actually be allocated that way. A prompt can inform the end-user that the values have exceed 100%, and that s/he may wish to modify values before running the simulation. (This prompt was built into several earlier models; it was omitted from the Aug. 17 version for simpler presentation, but can be reincorporated if desired.) If the end-user chooses to continue, priority will be assigned from left-to-right, i.e. first *Antivir%* of the *TotalSysAdminCapacity*, if available, will be allocated to antivirus; then up to *Firewall%*, if available, to firewall, and so on.

(Extend description: looking inside the HoursForTasks block, we see a series of equations, converting the percentages into *HoursDemanded*. Below that, we see a series of holding tanks. All of the tanks are reset to their starting values at the end of each day by a periodic pulse. The first tank has starting value *TotalSysAdminCapacity*; the other tanks have starting value zero. At the beginning of each day, *TotalSysAdminCapacity* flows into the first tank; *AntivirusHours Demanded* is "wanted" from that tank; the quantity "gotten" is the hours actually allocated to antivirus. The remaining contents of the first tank flow into the second tank, where again a demanded quantity is "wanted"; up to that quantity is "gotten", and the remainder flows into the third tank; and so on. Hours "gotten" are either used directly (firewall & IDS) or divided by the systemsize.)

For the version in the archetypes paper: a new allocation block was built, in which all demands are met if they sum to less than 100%; if they exceed 100%, then they are doled out in proportion to their demand, i.e. if the demands are: {30, antivirus; 40, tolerance; 50, IDS}; but the SysAdmin capacity is only 60; then it will allocate 15/20/25, respectively.

Additionally, the archetypes version adds in the factor of sysadmin inefficiency if pushed beyond optimal capacity. Hours allocated to a given task are decreased by a linear multiple of the total demand's exceeding the optimal capacity. (Note that we haven't yet included a factor to describe the inefficiency of X net demanded hours for ten different tasks, which has greater inefficiency than X net demanded hours for one task.)

I actually didn't use the allocation system in the new (*Escalation*) paper, I just "fed" each task directly as many hours as were desired.

The newest paper (DSN) once again made use of the allocation system – pushing the Sysadmin too far resulted in the "limit to growth." In this paper, the limit set in much more quickly, as we rephrased things: "sysadmin hours" were for all tasks; as soon as security demands take too many of those hours, other things go wrong because of ignored tasks. The rule used was thus the following (the numbers used here: 23%, 1.25, etc., were a combination of guesses on my part, and what made the graphs come out okay, i.e. GIVEN our guesses for how to describe a 200-machine system going from nominal attention to full attention to security over 180 days, we

wanted the limit to kick in towards the end of this period, and that the limiting effect be fairly strong.)

Maximum optimal security load = OPTPERCENT * TotalSysadminCapacity.

IF (total hours demanded for security > maximum optimal load), EfficiencyStretch = STRETCHCONSTANT * (total hours demanded for security – maximum optimal load).

AvailableCapacity = TotalSysadminCapacity – EfficiencyStretch.

VULNERABILITIES: Many attempted attacks will only succeed if the system has (known) vulnerabilities. These are grouped into four categories:

"Mistakes" includes all user mistakes, such as not logging off, downloading a virus, and using weak passwords. *(Mistakes* block outputs *MistakeFactor.)*

"Host-Configuration Vulnerabilities" include settings that the sysadmin didn't set properly, such as leaving ports open, allowing everyone access to sensitive files, etc. (*ConfigVulns* block outputs *HostConfigVulnsFactor.)*

"Network Vulnerabilities" include those flaws that have been discovered in the network software, which could be exploited by an attack; these can be corrected with patches. *(NetVulns* block outputs *NetVulnsFactor.)*

" Application Vulnerabilities" include those flaws discovered in application software, which could be exploited by an attack (e.g. a flaw in Apache could be exploited for a denial-of-service attack; a flaw in Outlook Express might be exploited

to cause remote execution of code). These are also corrected with patches. *(AppVulns* block outputs *AppVulnsFactor.)*

We measure each of these subclasses as a "factor" between 0 and 1, where 0 is best (no known vulnerabilities of this type exist on this system) and 1 is worst (i.e. the system is permeated with vulnerabilities of this type). An overall "vulnerability factor" (*VulnFactor),* also between 0 and 1, is computed from these:

*VulnFactor = Min{1, [(*MISTAKECOEFF * *MistakeFactor)*

+ *(*HOSTCONFIGCOEFF * *HostConfigVulnsFactor)*

+ *(*NETVULNSCOEFF * *NetVulnsFactor)* +

+ *(*APPVULNCOEFF * *AppVulnFactor)] }*

The "Min" function keeps the overall *VulnFactor* to a maximum of 1. Note that it is possible to reduce one or two vulnerability subfactors, and yet still have an overall factor of 1 if the other subfactors have been ignored. We believe that this reflects the reality of system vulnerabilities. We have weighted host-configuration vulnerabilities most heavily, followed by mistakes and application vulnerabilities, and then finally network vulnerabilities. This was Rosenfeld's impression of the most-frequently exploited vulnerabilities. (The host-configuration vulnerability is particularly pernicious, as an attacker often need not "breach" any part of the system to perform an attack; therefore, such an attack is often not detected by an I.D.S.) [How did you obtain such a ranking? Is it more based on the number of vulnerabilities of each type present or on the impact that each of these vulnerability types has?] I was thinking impact, e.g. a single config error could be more dangerous than a single app

vulnerability. Thus my comment about "more pernicious as it doesn't require a breach." Again, this is all my judgment here.

Archetypes Version: To show the difference between those vulnerabilities fixed by enforcement (i.e. host config and mistakes), and those fixed by patches (i.e. netvulns and appvulns), two more derived values were created: (A "strength" of 1 is best.)

ConfigStrength = 1 – (CONFIGHOSTCOEFF* *ConfigVuln*) – (CONFIGMISTAKECOEFF* *MistakeFactor)*;

SWStrength = 1 – (SWAPPCOEFF* *AppVulnFactor)* - *(*SWNETCOEFF * *NetVulnFactor);*

Comment [i3]: Page: 1
I don't understand this

SWStrength is a straight average of how well the apps and the netware/OS has been patched. (Again, with the "1 minus thing" to switch a 1->0 scale (vuln of 0 is best) to a 0-> 1 scale (strength of 1 is best).) For ConfigStrength, I weighted the average 60/40 between ConfigVulns and Mistakes; again, just my judgment as to how dangerous ConfigVulns are.

Newest version (*Escalation* paper): I left the "SWStrength/ ConfigStrength" same as the previous paper, as it worked perfectly well for my purposes here.

We now describe the workings of the individual vulnerability subfactors.

Mistakes. User mistakes are given as a factor of three conditions: The users' *Awareness* of security issues, as a value 0-to-1, where 0 is no awareness, 1 is very high awareness; the users' *Concern* for security issues, 0-to-1, (we assume at this

point that the lowest level of concern is 0, i.e. no concern; the issue of deliberate sabotage, where the user is "negatively concerned" with actively damaging the system, has not yet been incorporated into this model); and the sysadmin's *Hours Allocated to Enforcement Actions*, in pseudo-personnel-hours per machine per day. With proper sysadmin enforcement actions in place, the users' ability to make dangerous mistakes can be sharply minimized or eliminated altogether. We then compute an overall *Mistake Factor*, as a value between 0 (no dangerous user mistakes) and 1 (dangerous user mistakes happen all the time).

MistakeFactor = 1 – [{(AWARENESS + CONCERN*)* / USERMISTAKEDIVISOR}

+{ *Hours Allocated to Enforcement Actions* / ENFORCEMENTMISTAKEDIVISOR}]

Archetype version: the 0.7 was changed to 1.5. [Where are these numbers coming from?]

The *MistakeFactor* is then limited by a max of 1 and a min of 0. The numbers were designed as follows: even given perfect awareness and concern, i.e. *Awareness + Concern = 2*, if there are no enforcement actions, the mistake factor will still be .091 (not 0) to account for human error; (for example, this author recalls once downloading a virus simply because he accidentally clicked the wrong button.) Conversely, given sufficient enforcement actions (we assume .7 pseudo-hours per system per day would suffice), the mistake factor will go to zero, regardless of user awareness or concern. Note that the mistake factor is "memoryless" and employs no holding tanks; we assume that if awareness, concern, or enforcement were to suddenly decrease today, the effects would be felt immediately.

Comment [i4]: Page: 1
is this realistic?

While we would like to make *Awareness* and *Concern* variables that the end-user could adjust, for now we simply set both *Awareness* and *Concern* to 0.2. The *User Training for Better Security Practices?* switch adds TRAININGEFFECT to *Awareness*, raising it to 1. [Where are these numbers coming from?] (Future implementations may describe the effects of training over time, i.e. a gradual rise in awareness.) These numbers are all just guesses on my part about the "average" user's awareness and concern, and how much training can help. (I wanted to demonstrate the effectiveness of full training, so I let it raise Awareness all the way to 1.)

ConfigVulns has two inputs: *Hours Allocated to Enforcement Actions* (in pseudo-hours per system per day), and *A Host-Vulnerability Scanning Tool?* (0 if not present, 1 if present). A *HostConfigVulnsFactor* of 0 is best. The scanning tool assists the sysadmin by finding the vulnerabilities present; however, the sysadmin must still spend time fixing these vulnerabilities! We thus model the scanning tools as an increase in the "effective hours" (or "virtual enforcement hours") available for fixing vulnerabilities. If the scanning tool is not present, "virtual enforcement hours" = hours allocated to enforcement actions. If the scanning tool *is* present, "virtual enforcement hours" = CONFIGTOOLMULTIPLIER * hours allocated to enforcement actions.

Yes, it was my assumption for the present that the tool can double the sysadmin's effective hours here; just picked a number to try.

The vulnerability level is then described using (1 – the level of a holding tank), i.e. the holding tank is "full" when all vulnerabilities have been patched, and "empty"

Comment [i5]: Page: 1
For future versions we might want to elaborate on this, make it more similar to the attcks model, i.e., deal with number of vulnerabilities rather than with vulnerability level

when no vulnerabilities have been patched. (All of the holding tanks in this model have the default setting of "want" connector not being able to reduce the tank value below zero.) The holding tank has a "loss rate"; this describes the fact that over time, new vulnerabilities are discovered; additionally, new user accounts are created, etc., all requiring attention from the sysadmin to prevent additional vulnerabilities. We set the loss rate here to .05, i.e. if the sysadmin configured the system perfectly, but then ignored it for twenty days, it would now look very vulnerable. (For now we assume a linear loss rate. Further detail may modify this in the future.) (Yes, all assumptions of mine which could use validation.) The next question we ask is, how many pseudo-personnel hours are required to maintain the tank at its "full" level of 1? For configuration vulnerabilities, we assume it to be 1.5 pseudo-hours per system per day. (Or with the scanning tool, .75 pseudo-hours per day.) We then have a "divisor" of (1.5 / .05) = 30, i.e.: input to the tank = *Virtual Enforcement Hours* / 30.

The tank is designed that if the current level of the tank is already 1 (full), additional input (i.e. additional hours) will not raise the tank level further. Lastly, the initial level of the tank is also decided by the number of virtual enforcement hours. If 1.5 or more pseudo-hours are available, the initial level will be 1. Otherwise, the initial level will be (pseudo-hours allocated / 1.5). Expressed in terms of the "divisor" and "loss rate" constants, this is:

StartLevel = Min{1, [*Hours / (Divisor * LossRate)*] }.[I don't understand this discussion on the tank. What is the main message?]

What happens, as given, is that if the model starts out with enough staff-hours to keep the configvulns "happy", it will start at "full" and stay that way. Otherwise, it will inevitably decline to zero.

This raises the question: if you go through a year of only spending half the time you should on patches (or tolerance, etc.), by the end of the year, how vulnerable is your software? Totally? 50% Not sure.

In the old models, anything that required sysadmin hours was designed that if it didn't get enough of them, it would ultimately decline to zero no matter what in the long term. In the *Escalation* paper, this was changed for IDS, Firewall, NetVulns and ConfigVulns: for all of them, the effectiveness TODAY of a given countermeasure is given by a maximum of two values: the tank level (which reflects what it had been given in the past), AND the number of hours given TODAY divided by the number of hours required to be fully happy. Thus, if I patch well for a long time, then ignore patches for a few days, the patch level will be 0.9 or so. On the other hand, I could have totally ignored patches for years, but if I spend some time on them today, patches will be somewhat effective today. Please let me know what you think of this.

For the archetype version, to get better-looking results we often changed the start level to something specified, e.g. 0 or 1 or some constant in between that worked nicely. This describes a scenario of "new sysadmin walks in on a system that had been totally ignored for a long time", or "incompetent sysadmin ruins a system that had been fine."

The above scheme of holding tanks, loss rates, divisors, and starting levels, will be found repeated throughout other parts of this model.

NetVulns are designed very much like configuration vulnerabilities, except here the inputs are *Hours Allocated to Software Patches* and the presence of *A Network Vulnerability Scanning Tool?* Once again, the scanning tool increasesthe virtual hours available for patching network vulnerabilities as follows:

Effective Netvuln-fixing Staff-Hrs. = NETTOOLMULTIPLIER **Hours Allocated to Software Patches*. The *NetVulnsFactor* is similarly given as (1 – tank level). (The output from the tank is limited to the range 0 to 1. However, the feature "if tank level = 1, don't allow further input" was not added to the *NetVulns* block (or any other block in this model yet) due to time constraints; thus, as is, the theoretical tank level can exceed 1, but the most it will read out is 1.) LossRate = 0.04 (i.e. totally vulnerable if ignored for 25 days), divisor = 15 (i.e. fully patched if given .6 pseudo-hours per system per day), initial level = virtual hours / 0.6, limited between 0 and 1. Archetype Version: Divisor was changed to 32. [Where is all that coming from?]

Comment [i6]: Page: 1
what does this mean in terms of the real world?

AppVulns has inputs *Hours allocated to Software Patches* and *New Software is Installed Every X Days.* As before, a tank 0-1 describes the "strength" of the software; it is replenished by "hours allocated to patches", with a divisor of 33.33. The loss rate is .02. ADDITIONALLY, anytime new software is added, this causes an additional loss of 0.8. The addition of new software is modeled as an event that occurs every Y days, where Y is Gaussian, mean *New Software Installed Every X Days*, std. deviation 30%. Archetype Version: LossRate is 0.004. [Explain why you selected these values.]

COUNTERMEASURES: Countermeasures behave much like vulnerabilities, only a factor of 1 is best (countermeasure is fully effective), 0 is worst. Each countermeasure

outputs its factor, 0-to 1; and has inputs for the hours allocated to it, as well as a binary value indicating whether it is present. (If the switch, e.g. *Antivirus Software?*, is off, the factor output will always be 0.) All of the countermeasures are represented by holding tanks. A "limit" block applied to the tank's contents level ensures that the output will be between 0 and 1. *However*, the feature "tank itself can not exceed 1", i.e. "if tank capacity = 1, today's input = 0", was not yet built into the countermeasures as it was into the vulnerabilities. (This can be easily changed.) Thus, for now, if the sysadmin were to put "super" efforts into a countermeasure for a while, the tank level would exceed 1; the factor output will still be 1; however, the sysadmin could ignore the countermeasure for a short time and it will still have a factor of "1", as the loss rate drains the tank from a value greater than 1 to 1. (Eventually, though, the tank will drain below 1.)

Comment [i7]: Page: 1
I don't understand

Comment [i8]: Page: 1
I believe THIS IS realistic

Starting level of the tank, if not manually adjusted, will follow the same equation as the vulnerability tanks: (hours allocated) / (total hours needed for the countermeasure to be "happy", = divisor * lossrate). (This is then limited between 0 and 1.)

Tolerance Mechanisms. These can be high-maintenance, as this includes diversity. Loss rate 0.1, divisor 6.67. [Why?] It was just assumed that tolerance measures are high-maintenance, especially if we include diversity; so I picked values these values: if tolerance measures are ignored for ten days, they become useless (loss rate 0.1); and that 2/3 pseudo-staff-hours per machine are required to keep these tolerance measures fully maintained.

Encryption Software. Once in place, this is fairly low-maintenance. Loss rate 0.001, divisor 66.6667. [Why?] Again, I just picked numbers that would imply low-maintenance, i.e. an encryption system, once ignored, takes > 2 years (i.e. 1000 days) to become useless (there still may be bug fixes, updates, and the like); and it doesn't take much work to keep the encryption up (or deal with users having problems with it), so I just figured an average of .067 pseudo-staff-hours per machine per day.

Antivirus Software. We assume that new definitions must be installed by the sysadmin. Loss rate 0.02 (i.e. useless after fifty days, given that ~2.5 new viruses come out each day, looking at a list from McAfee or the like.) Divisor 20. How much time per day per machine is needed to keep the antivirus up-to-date? I assumed 0.4 pseudo-staff-hours / machine / day.

Firewall and IDS [Indeed, they should be separate.]. We described "firewall efforts" and "IDS efforts" each as separate from "hours for analyzing (false) alarms", which includes the alarms generated by both. Thus, the *FirewallIDS* forms one unit, with inputs: *Hours Allocated to Firewall, Hours Allocated to (False) Alarms, Hours Allocated to IDS,* and the binary switches *A Firewall?* and *An Intrusion Detection System?*. Outputs are *FirewallFactor* and *IDSFactor*, both within [0,1]; and *FalseAlarms,* measuring how many staff-hours-per-machine's worth of alarms are generated on a given day.

Comment [i9]: Page: 1
why are these not separate?

Comment [i10]: Page: 1
why? Shouldn't they be separate?

Firewall effectiveness and IDS effectiveness each start off as independent holding tanks similar to those of the other countermeasures. Thus, *FirewallFactor* and *IDSFactor* are simply the contents value of their respective tanks. Firewall has

LossRate 0.0333 and Divisor 20; IDS has LossRate 0.033 and divisor 8. [Why these numbers?]

However, both the IDS and firewall generate more alarms as they become more effective.

*AlarmRate = (0.3 * FirewallFactor)+ (0.6 * IDSFactor).[No, they should not be mixed.]*

If these alarms are not addressed, they become *Ignored Alarms.*

IgnoredAlarms = AlarmRate – AlarmHours, with a minimum of 0. All of these are measured in staff-hours per machine per day.

IgnoredAlarms leads to a steep decline in the effectiveness of the IDS and firewall, with several days' delay.

The "want" (i.e. drain) on firewall effectiveness is the "natural" loss rate due to the need for routine maintenance, patches and the like, which was given as 0.0333; plus 0.25 * *IgnoredAlarms,* with a five-day delay on the latter.

Similarly, drain on IDS effectiveness is 0.0333 (natural loss rate), plus 0.33 * *IgnoredAlarms,* with a three-day delay on the latter.

Based on this, running the model with a high number of hours dedicated to the IDS or firewall, but few hours to analyzing alarms, will result in IDS and firewall effectivenesses that show decaying nonnegative oscillations, i.e. high, then low, then medium, then low, and so on, until they reach a level of zero.

Archetype Version: Here, we wanted to show gains per effort for a single variable, so we included alarm analysis into the IDS efforts (and hours). (The firewall model was unchanged, as our archetypes did not include a firewall.) We now have a holding

tank with a starting level of zero, and an input divisor of 40. Loss rate is now entirely a factor of the IDS effectiveness: LossRate = 0.35* IDSlevel [i.e. the contents of the holding tank], with a 15-day delay. This causes the oscillations seen in the attack success rate of the *Shifting the Burden* IDS scenario.

We then argue that even an ignored IDS will still catch *some* attacks; this assumption also keeps the oscillations in the archetypes paper from being too extreme. This is accomplished by simply letting *IDSFactor = (contents of holding tank) + 0.6*, with a maximum of 1. This means that a totally ignored IDS will still have 60% the effectiveness of a well-maintained one. (The author claims no sources in the literature to support this, other than "it made the graph look nice.")

Newest version: we've kept everything separate: addressing IDS alarms goes into IDS Efforts; addressing Firewall alarms goes into Firewall Efforts. Firewall has loss rate .033 (i.e. useless if ignored for 30 days, just my assumption) and divisor 20 (i.e. for full effectiveness, firewall should have 0.66 staff-hours / day; in this paper, I assumed .66 staff-hours per day per machine. (That's too high, isn't it? Again, that's skill-motivation-staff-hours, which is easily double the number of actual staff-hours.) For the IDS, loss rate 0.05 (i.e. useless if ignored after 20 days, again my assumption), and a divisor of 40, i.e. best to provide the IDS 2 staff-hours per day (per system).

ATTACKS: We divide the attacks into three categories by their sophistication. (This three-way division is found in some DARPA presentations that have not yet been published.) *Simple Attacks*, (or "kiddy-scripts"), almost always rely on known

vulnerabilities and require little action from the attacker other than downloading and running the attack. A "sitting-duck" server may be subject to 50 or more simple attacks per day. Dr. Cukier's empirical findings support roughly this number. (Though his ~50 did not include viruses.)

"*Sophisticated Attacks"* may involve finding new vulnerabilities, can often defeat many countermeasures, and usually come from a single knowledgeable attacker (such as one who might actually write the "kiddy scripts" used in the first category). The average company will sustain only a handful, at most, of sophisticated attacks per day. (Yup, just an assumption; Dr. Cukier is trying to get sophisticated attackers to hit his systems, but not much luck yet. Wasn't there a quote from Dr. Cukier about 95% simple / 5% sophisticated or something like that?) Certainly we must include computer viruses, the most costly computer-security breach as

reported in the CSI/FBI survey, in our discussion. While a computer virus does require a sophisticated author if it will spread, it spreads in fairly simple, predictable ways, and is easily defeated by simple countermeasures (antivirus) and patching vulnerabilities; we therefore include viruses in the simple, "script-kiddy" category.

Lastly, we have *Nationwide*-level *Attacks***,** which may be part of a war effort, global terrorism, possibly a multinational corporation attacking a competitor, and so on. Most companies will only see one of these every few months or so, if at all. (That seemed like common sense.) Attacks of this sophistication do not rely on vulnerabilities as they can "brute force" through most software; they can also defeat most countermeasures.

For the time being, we do not differentiate attacks other than their categories of *Simple, Sophisticated,* and *Nationwide.* The model's end-user inputs *AverageSimple, AverageSophisticated,* and *AverageNationwide* via sliders. The outputs of the respective "attack generator blocks" are *Simple/Sophisticated/Nationwide Attacks Attempted*. To add realism to our model, some randomness occurs between the input *Average*s and the output *Attempted*s:

All of the above behave the same way. If *Average* >= 1, then a number Y is output, where Y represents the number of attacks of that type attempted per day. Y is given by a Gaussian distribution, with mean *Average* and a standard deviation of 0.2 * *Average.* (i.e. "a standard deviation of 20%.")

If *Average* < 1, then exactly one attack is attempted every Z days, where Z follows a Gaussian distribution with a mean of (1/Average) and a std. dev. of 30%.

Archetype Version: for simplicity, and to prevent oscillations in the graph due to randomness, we simply let *AverageSimple* = *SimpleAttacksAttempted* = 100. (We circumvent the "attack generator block.") All other attacks are set to 0.

In the new (*Escalation*) paper, I had no randomness in Simple Attacks, but a 5% standard deviation in Sophisticated Attacks. Just numbers I picked to demonstrate some randomness; I don't know how much the numbers vary day-to-day in real-life.

ATTEMPTED VS. SUCCESSFUL ATTACKS, or ATTACK DEFENSES. Even if perfectly effective, a given countermeasure is only so successful at thwarting attempted attacks. For example, if we say 100 attacks are attempted per day, we include a certain number (call it X) of viruses. The best antivirus in the world will

thwart all X of those viruses, i.e. X% of the total attacks, but it can not defeat more than X% of the attacks because (100-X) attacks are not viruses. As for what percentage of attacks are not successful due to a given countermeasure, the only numbers available are those of experts' opinion and the CSI/FBI survey. The CSI/FBI survey is of limited use, however, as it records what percentage of correspondents reported observing a given type of attack on their system. Thus, we know that 78% of the businesses surveyed detected a virus last year, and 37% detected a DoS; that does not mean that 78% of the attacks out there are viruses or that 37% of them are DoSs! (Otherwise, the numbers exceed 100% quite rapidly.) Nonetheless, the numbers can be used as a very rough approximation for the prevalence of a given attack. Otherwise, the numbers given here represent the author's numerical interpretations of M. Cukier's descriptions of "fully effective", "partially effective", or "not effective" for each countermeasure against each category of attack.

Similarly, certain countermeasures may be very effective against simple attacks, but not against sophisticated ones. We therefore have three different blocks labeled *SimpleSuccess*, *SophistSuccess*, and *NatnwideSuccess*, respectively. (Extend's limits on the number of characters in a hierarchical block's name necessitated some creative spellings here.) Each of these takes as inputs *Attempted XYZ Attacks*, where XYZ is simple/sophisticated/nationwide. They also have inputs for the factors of all relevant countermeasures and vulnerabilities. The primary output is the number of successful attacks of a given category. The other outputs appear on the bottom of the AttackSuccess block, directly beneath the inputs for the various countermeasures and vulnerabilities. These outputs show the number of attacks per day not successful due

to the corresponding countermeasure or lack of vulnerability. Additionally, it appears that a “thinner” block has been attached to the bottom of each Attack Success block. This functions as an accumulator, showing how many attacks have been attempted, successful, or not-successful-due-to-a-given-factor, over the entire simulation period.

Each *AttackSuccess* block is designed in the same way, as a linked series of holding tanks: at the beginning of each day, all tanks are reset to zero. Then, a certain number of attacks (attempts) are input to the first holding tank; some are removed by the first countermeasure (in proportion to how effectively it is functioning, e.g. is the *AntivirusFactor* 1, i.e. it has been well-maintained, or something lower?); the remaining tank contents (i.e. remaining attacks) are transferred to the second tank, where some are removed by the second countermeasure, and so on; those that remain after all the tanks are done are deemed *Successful Attacks*.

For simple attacks, we have the following procession: As an attempted attack enters the system, it first encounters the firewall, then an IDS; if it passes those, it will be scanned by an antivirus. If it still passes through, it may be designed to exploit a given vulnerability in the system; if that vulnerability is not present, it will be thwarted here. If it still succeeds, encryption may sometimes help as follows: even if the system is breached and data is illegally accessed, an attacker will find the encrypted data meaningless; confidentiality is thus maintained. Finally, if all else fails, tolerance measures will mitigate the damage in many cases. Thus, starting with attempted simple attacks, we have the following: [Not completely right. In particular, the antivirus focuses mainly on email attachments. Otherwise, the antivirus can detect

Comment [i11]: Page: 1
Michel, what do you think?

the corruption of the computer. We can work this out during our next meeting, OK?] We're still working on this, but the models haven't changed it yet.

Remove (*FirewallFactor* * 90%*)* of the attempted attacks. I.e. if the Firewall is fully effective, it will catch 90% of the attempted simple attacks; if it's only 50% effective (supposing it hasn't been well-maintained), then it will catch only 45% of attempted attacks. Of those remaining, remove (*IDSFactor* * 60%); of those remaining, remove (*AntivirusFactor* * 78%); of those remaining, remove ((1 – *VulnFactor*) * 90%); this represents those attacks that were designed to exploit a given vulnerability; if that vulnerability is not found, the attack will not succeed.

Archetypes Version: in order to differentiate between the results of enforcement actions (which influence config vulns and mistakes) and patches (which influence NetVulns and AppVulns), we have each defeat attacks separately, rather than taking 90% * (1 – VulnFactor). Instead, remove (*SWStrength* * 60%), then (*ConfigStrength* * *80%).* In displaying those attacks defeated by *ConfigStrength,* we adjust the equations to show total attacks defeated by *ConfigStrength*, not those attacks defeated by *ConfigStrength* that were not previously defeated by *SWStrength*.

Then remove (*EncryptFactor* * 40%). (Encryption is only useful in preventing theft of data; it does very little, for example, against a DDoS attack.) Lastly, remove (*ToleranceFactor* * 75%). Take this result and apply the "floor" function, i.e. largest integer that is less than or equal to it. (Thus, if after all the countermeasures, we have 2.2 attacks succeeding, count that as 2. If we have 0.9 attacks succeeding, count that as 0.) Archetype version: to make the lines smoother, we leave out the floor, and instead interpret the results simply as "percentage of attacks succeeding." We now

have the number of *SuccessfulSimpleAttacks*. (The various summed-over-time outputs are found simply by inserting accumulation tanks at the appropriate point in the chain.)

All of these percentages were either my assumptions, some comment from Dr. Cukier about "very effective/somewhat effective/not effective", and occasionally, the survey (see above about 78% saw viruses.)

For Sophisticated and Nationwide attacks, many less countermeasures are effective. Furthermore, even a single attack stands a good chance of succeeding. This is represented as follows: after reducing the appropriate percentages due to countermeasures and vulnerabilities, we are left with what should be X successful attacks. If X >=1, round X to the nearest integer; that is how many attacks of this type are successful today. If $0 < X < 1$, one attack will succeed an average of X% of the time. This is accomplished by selecting a random value r uniformly distributed on [0,1]; if $r < X$, the attack succeeds; otherwise, it does not.

For sophisticated attacks, the antivirus is ineffective because all viruses are treated as simple attacks. An IDS can be defeated by a clever attacker, so it is not included. Encryption (which we assume can not be defeated without a supercomputer of some type (Dr. Cukier agreed with this; I've heard in the news that every now and then a team of experts with 100 computers has cracked a given file encrypted with RSA, after working on it for a few months.) which is beyond the reach of a single sophisticated attacker) is still as effective as with simple attacks; the same goes for tolerance. A firewall is effective, but less so because it can sometimes be defeated.

Lastly, some sophisticated attacks are designed to exploit known vulnerabilities, but often a sophisticated attacker can find his/her own new vulnerabilities in the software. We thus are left with the following:

Remove (*FirewallFactor* * 30%) of attempts; of the remaining, remove ((1 – *VulnFactor*) * 50%); of the remaining, remove (*EncryptFactor* * 40%); lastly, of the remaining, remove (*ToleranceFactor* * 75%). The remaining value is rounded to the nearest integer if it is >= 1, or used as a probability if it is < 1, as described above. The result is the number of *SuccessfulSophisticatedAttacks*. Only source other than Dr. Cukier's comments or my guesses I can add here is this: Encryption is helpful for whatever percentage of attacks sought to steal sensitive data. What is that percentage? Survey talks dollar costs of various attacks (e.g. theft of data vs. DoS), but not the percentage breakdown of the number of attacks themselves.

In the case of Nationwide attacks, we assume that network and application vulnerabilities are irrelevant as the code is subject to "strong smart force", the nationwide-scale attackers may have access to the code being used; similarly, the nationwide attacker possesses a supercomputer, quantum computer, or some other method of defeating commercially available cryptography. The only countermeasures that are effective (and partially at that) are the firewall (if it is a hardware firewall of proprietary design, as M. Cukier described in an experience of his) and tolerance measures.

Remove (*Firewall Factor* * 20%), then of the remaining, remove (*ToleranceFactor* * 50%). Apply the rounding or probability as described above; the result is the number of *SuccessfulNationwideAttacks*. Dr. Cukier had said something

about Tolerance being fully effective against simple & sophisticated attacks; halfway effective against nationwide attacks.

Lastly, the three categories of successful attacks can be summed; each AttackSuccess block is connected to an addition block. The result is *All Successful Attacks (Per Day).* Similarly, if one wishes to see all successful attacks over the entire simulation period, the various *Successful ABC Attacks (Sum Total),* for ABC = {Simple, Sophisticated, Nationwide}, sum to *All Successful Attacks (Sum Total).*

Bibliography

[Baj04] R. Bajcsy et al, "Cyber defense technology networking and evaluation," *Communications of the ACM*, vol. 47, no. 3, March 2004, pp. 58—61.

[Ber04] D. Bergemann, J. Feigenbaum, S. Shenker, and J. Smith, "Towards an economic analysis of trusted systems," presented at Third Annual Workshop on Economics and Information Security (WEIS '04). Minneapolis, May 13—14, 2004.

[Ber05] D. Bergemann et al, "Flexibility as an instrument in DRM systems," presented at Fourth Annual Workshop on Economics and Information Security (WEIS '05). Cambridge, MA, June 2—3, 2005.

[Bod05] L. Bodin, L. Gordon, and M. P. Loeb, "Evaluating information security investments using the analytic hierarchy process," *Communications of the ACM*, vol. 28, no. 2, Feb. 2005, pp. 79—83.

[Bra02] W. Braun, "The System Archetypes," (2002), Available at http://www.uni-klu.ac.at/gossimit/pap/sd/wb_sysarch.pdf

[Cam03] K. Campbell, L. A. Gordon, M. P. Loeb, and L. Zhou, "The economic cost of publicly announced information security breaches: empirical evidence from the stock market," *Journal of Computer Security*, no. 11, 2003, pp. 431—439.

[Coh99] F. Cohen, "Simulating CyberAttacks, Defenses, and Consequences," (March 1999), Available at: http://all.net/journal/ntb/simulate/simulate.html

[Coh06] Fred Cohen & Associates, "Strategic Games", (2006), http://all.net/games/index.html

[Dac04] M. Dacier, F. Pouget, and H. Debar, "Honeypots: practical means to validate malicious fault assumptions," In *Proc. 10th IEEE Pacific Rim Int'l Symposium on Dependable Computing (PRDC '04)*, 2004, pp. 383—388.

[Dan04] D. Danchev, "Reducing 'Human Factor' Mistakes." WindowSecurity.com, (2003), www.windowsecurity.com/articles/Reducing_Human_Factor_Mistakes.html

[Dwa05] Z. Dwaikat, "Attacks and countermeasures," *CrossTalk: The Journal of Defense Software Engineering*, Oct. 2005, Available at: www.stsc.hill.af.mil/crosstalk/2005/10/0510Dwaikat.html

[Fei05] J. Feigenbaum et al, "Subjective-cost policy routing," in *Proceedings of the Workshop on Internet and Network Economics, Lecture Notes in Computer Science*, vol. 3828, Berlin: Springer, 2005, pp. 174--183.

[For61] J. W. Forrester, *Industrial Dynamics*, Cambridge: The Wright-Allen Press, 1961.

[Gib02] S. Gibson, "The Strange Tale of the Denial of Service Attacks Against GRC.com." Gibson Research Corporation, (March 2002), Available at: www.grc.com/files/grcdos.pdf

[Gon01] F. Gong, K. Goseva-Popstojanova, et al, "Characterizing intrusion tolerant systems using a state transition model," In *Proc. DARPA Information Survivability Conference and Exposition II* (*DISCEX '01)*, 2001.

[Gor02] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Trans. Information and System Security*, vol. 5, no. 4, November 2002, pp. 438—457.

[Gor05a] L. A. Gordon, M. P. Loeb, et al, "Tenth Annual CSI/FBI Computer Crime and Security Survey," Computer Security Institute, (2005), Available at: www.gocsi.com/forms/fbi/csi_fbi_survey.jhtml

[Gor05b] L. A. Gordon and M. P. Loeb, *Managing Cybersecurity Resources: A Financial Perspective*, New York: McGraw-Hill, 2005.

[Hof05] S. Hofmeyr, "The Information Technology security arms race," *CrossTalk: The Journal of Defense Software Engineering*, Oct. 2005, http://www.stsc.hill.af.mil/crosstalk/2005/10/0510Hofmeyr.html

[Hun06] C. L. Huntley, "A Developmental View of System Security," *Computer*, vol. 39, no. 1, pp. 113—114, January 2006.

[Ima05] Imagine That, Inc., *Extend.* (Version 6.07). [CD-ROM]. [Windows 98/ME/NT4/2K/XP]. San Jose, CA, 2005.

[Irv05] C. E. Irvine, M. F. Thompson, and K. Allen, "CyberCIEGE: gaming for information assurance," *IEEE Security & Privacy Magazine*, vol. 3., no. 3, May-June 2005, pp. 61—64.

[Jha01] S. Jha and J. M. Wing, "Survivability analysis of networked systems," in *Proc. of the 23rd International Conference on Software Engineering (ICSE '01)*, 2001, pp. 307—317.

[Jon97] E. Jonsson and T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior," *IEEE Transactions on Software Engineering*, vol. 23, no. 4, April 1997.

[Lar03a] R. LaRose and M. S. Eastin, "A social cognitive explanation of Internet uses and gratifications: toward a new theory of media attendance," presented at International Communication Association, Communication and Technology Division. San Diego, May 2003.

[Lar03b] R. LaRose and N. Rifon, "Your privacy is assured --- of being invaded: Web sites with and without privacy seals," presented at IADIS International Conference. Lisbon, Portugal, June 3—6, 2003.

[Mar03] K. Marais and N. Leveson, "Archetypes for organizational safety," presented at IRIA '03. Williamsburg, VA, 2003, Available at: http://sunnyday.mit.edu/papers/iria-marais.pdf

[Nav06] Naval Postgraduate School and Rivermind, Inc., *Cyberciege*, version 1.5b, Feb. 2006, [Win2000/XP] http://cisr.nps.navy.mil/cyberciege/index.htm

[Ort99] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, Sept.-Oct. 1999, pp. 633—650.

[Pan05] S. Panjwani et al, "An experimental evaluation to determine if port scans are precursors to an attack," in *Proc. International Conference on Dependable Systems and Networks (DSN05)*, Yokohama, Japan, June 28—July 1, 2005.

[Pou04a] F. Pouget and M. Dacier, "Honeypot-based forensics," presented at AusCERT Information Technology Security Conference 2004 (AusCERT '04). Ashmore, Australia, May 23—27.

[Pou04b] F. Pouget, M. Dacier, and V. H. Pham, "Understanding threats: a prerequisite to enhance survivability of computing systems." Presented at International Infrastructure Survivability Workshop 2004 (IISW04), in conjunction with 25th International Real-Time Systems Symposium (RTSS04). Lisbon, December 5—8, 2004, Available at: http://www.honeynet.org/papers/individual/IISW04.pdf

[Pou05] F. Pouget, M. Dacier, and V. H. Pham, "Leurre.com: On the advantages of deploying a large scale distributed honeypot platform," In *Proceedings E-Crime and Computer Conference 2005 (ECCE '05)*, March 2005.

[Ros06a] S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling and simulation of the escalation archetype in computer security," presented at *2006 Symposium on Simulation and Software Security (SSSS '06)*. Huntsville, AL, April 2006.

[Ros06b] S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling the symptomatic fixes archetype in enterprise computer security," submitted to *30th Annual International Computer Software and Applications Conference (COMPSAC06)*. Chicago, Sept. 2006.

[Ros06c] S. N. Rosenfeld, I. Rus, and M. Cukier, "Archetypal behavior in computer security," submitted to *Sixth European Dependable Computing Conference (EDCC-6)*. Coimbra, Portugal, Oct. 2006.

[Sen90] P. M. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization*, New York: Doubleday Currency, 1990.

[Sen94] P. Senge, A. Kleiner, et al, *The Fifth Discipline Fieldbook*, New York: Doubleday, 1994.

[Ste04] F. Stevens, T. Courtney, et al, "Model-based validation of an intrusion-tolerant information system," In *Proc. of the 23rd Symposium on Reliable Distributed Systems (SRDS '04)*, 2004, pp. 184—194.

[Wol03] E. F. Wolstenholme, "Toward the definition and use of a core set of archetypal structures in system dynamics," *System Dynamics Review*, vol. 19, no. 1, Spring 2003, pp. 7—26.

Publications and Submissions

- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling and simulation of the escalation archetype in computer security," 2006 Symposium on Simulation and Software Security (SSSS '06). Huntsville, AL, April 2006. (50% acceptance rate.)
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling the symptomatic fixes archetype in enterprise computer security," submitted to *30th Annual International Computer Software and Applications Conference (COMPSAC06)*. Chicago, Sept. 2006.
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Archetypal behavior in computer security," submitted to *Sixth European Dependable Computing Conference (EDCC-6)*. Coimbra, Portugal, Oct. 2006.
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modelling the tragedy of the commons archetype in enterprise computer security," soon to be submitted to *IEE Proceedings Information Security*.

ABSTRACT

Title of Document: SYSTEM DYNAMICS MODELING AND SIMULATION OF ENTERPRISE COMPUTER SECURITY

Shalom Nachum Rosenfeld, Master of Science, 2006

Directed By: Professor Michel Cukier, Reliability Engineering
Dr. Ioana Rus, Fraunhofer Center USA

To support decision-making, training, and understanding complex trends in enterprise computer security, we have built an executable model representing the major components of an organization's computer security, including its machines, users, administrators, countermeasures, and attacks. We use "if-then" rules to express behaviors, incorporating the notions of “archetypes”, i.e. frequently-observed patterns of system behavior, and “system dynamics”, a discipline which views system behavior in terms of stocks and feedback loops. This thesis describes the model, and then discusses several archetypal behaviors and their results, namely: *Symptomatic Fixe*s (or “*Shifting the Burden*”), *Escalation*, and *Escalation* combined with *Limits to Growth*. Simulation is used to display these behaviors quantitatively, and to show the effects of possible solutions. We conclude by discussing how such results can be useful for practical computer security, and how this model can both feed off other security research and fuel it.

SYSTEM DYNAMICS MODELING AND SIMULATION OF ENTERPRISE COMPUTER SECURITY

By

Shalom Nachum Rosenfeld

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2006

Advisory Committee:
Professor Michel Cukier, Chair
Dr. Ioana Rus
Professor Min Wu

Acknowledgements

The author gives his highest thanks to his advisors, Dr. Michel Cukier and Dr. Ioana Rus, for their tireless dedication, for believing in him, and for working so hard to give him the opportunity to make this thesis happen. The entire E.C.E. and Reliability departments, as well as the Fraunhofer Center, are also to be thanked for all of their assistance.

The author also thanks Dr. Min Wu for her assistance on the examining committee and her incisive commentary and questions.

Interestingly, the author's first introduction to archetypes occurred when Dr. Virgil Gligor tangentially described the Tragedy of the Commons during a lecture on distributed systems.

In addition to the above, a series of mentors (from a wide variety of fields) should be acknowledged. What follows is an incomplete list: Rabbi Shlomo Crandall; Rabbis Emanuel, Israel, and Rafael Moshe Gettinger; Rabbi Rafael Pollack; Rabbi Simcha Fishbane; Rabbi Michoel Elias; Rabbi Yehudah Shmulevitz; Rabbi Yitzchak Breitowitz; Dr. Elliot Bartky; Mr. Mitch Aeder; Professor Kenneth Kramer. Their individual contributions are too great to be enumerated here.

The author thanks his colleagues from the Reliability Lab and from other E.C.E classes, whose assistance proved invaluable.

None of this could have happened without the role models of the author's parents, grandparents, and entire extended family.

The author thanks his wife for her care, patience, and support.

Finally, the author thanks G-d for all of the above – and everything.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

An enterprise computer system is highly complex, consisting of multiple hosts with different platforms and different applications, all networked and most likely connected to the Internet. These components have flaws that make the system vulnerable and allow attackers to exploit these vulnerabilities.

Humans and machines form an even larger and more complex system with many different components and interactions. Control actions and reactions on one side of this system might have not only a local effect, but could also affect the rest of the system, often resulting in feedback loops. These effects manifest themselves over time with different delays. The properties of the system (security being one of them) will emerge from its structure and all these interactions between its components.

Some of the events in such systems are non-deterministic. This, and the fact that we do not have complete and fully accurate knowledge about these systems, leads to a level of information uncertainty that must be acknowledged and handled appropriately. Due to all of the above intricacies of such a system, it is extremely difficult to understand and analyze its emerging properties and the properties of the services it provides.

It is a hard task to characterize and assess the security of such a system, let alone to predict malicious acts and to design a strategy for eliminating or at least reducing their effects. Nonetheless, such a strategy is imperative, especially for systems such

as national infrastructures, military or other government systems, emergency systems, or banks.

Protection against attacks can be achieved by preventing, detecting, and tolerating them. Tolerating attacks might require the system to function in a degraded mode once under attack. If attacks defeat all lines of defense and eventually succeed, then the system must be able to recover quickly to an operational and secure state. Of course, all actions needed for proper prevention, detection, and tolerance have costs associated with them, including the price of buying and maintaining tools, the effort and time to install and run them, and personnel training. A strategy for security achievement and risk reduction can comprise a combination of the aforementioned actions. Given resource constraints, as well as trade-offs that might be needed between security on one hand and other operational properties (for example usability or performance) on the other hand, designing such a strategy is a very challenging task and requires extensive knowledge and experience.

1.2 Approach

To support this decision-making process of designing an appropriate security strategy, we developed a quantitative executable model of an organization's operational computer security. Like all models, this is an abstraction of the real system, focused on representing the security-significant aspects of the system and associated processes. The model targets and represents the perspective of the person who must make decisions regarding actions that must be taken for security assurance and security-related risk management. The user of the model can set different values

for the model parameters, corresponding to different usage, vulnerabilities, attacks, and defense profiles. The simulator can be run and different "what-if" scenarios can be executed. Simulation will help a security manager, security engineer, or system administrator answer questions such as: if my environment is characterized by these values, then what methods and tools to select and apply for managing security risks and satisfy the users needs of my system? How will the selected actions work together? What is their effectiveness and cost efficiency? To what changes is my environment most sensitive? If I make specific changes in my security strategy, what will be their impact? What changes if my system gets attacked more/less or if the time to exploit changes? Should I hire more system administrators? Should I spend more on training them?

The model aims first at understanding security risk reduction in computer systems, then at diagnosing such systems and identifying their weaknesses, as well as prospectively examining the effectiveness of different solutions. The description of the behaviors this model can exhibit is founded upon the notion of system archetypes.

1.3 Archetypes

Archetypes are a concept related to systems thinking, developed in the mid 1980s, in an attempt to describe complex behavior and to convey ideas in an easier and more efficient manner. Archetypes are frequently-observed patterns of systems behavior and are a "natural vehicle for clarifying and testing mental models" about systems or situations [For61]. The systems literature describes ten distinct archetypes, as listed by [Bra02] and outlined in Appendix I. [Wol03] argues that in fact, all of these can be

categorized into one of four "core generic" archetype classes: "Underachivement" includes *Limits to Growth, Attractiveness Principle, Tragedy of the Commons,* and *Growth and Underinvestment*; and "Relative Archievement" includes *Success to the Successful*. "Out-of-Control" includes *Fixes that Fail, Shifting the Burden,* and *Accidental Adversaries;* and lastly, "Relative Control" includes *Escalation* and *Accidental Adversaries*. [Wol03] acknowledges that the more common description of archetypes (i.e. that of [Bra02]) is more intuitive and easier to grasp and apply to simulation, so it is used here. Archetypes have been mainly applied in business or industrial processes. There has recently been some work performed at MIT in applying systems thinking and archetypes to systems safety [Mar03], but in security this is a new idea.

Beyond the common archetypes of [Bra02], we keep in mind that other archetypes may be observed in security. This would not be surprising, as [Mar03]'s application of archetypes to safety engineering uncovered several security-specific archetypes. This thesis, however, restricts itself to the application of common archetypes to security. While Appendix I describes how each of the ten archetypes might be applied to security, this thesis gives a detailed understanding of the following archetypes: *Symptomatic Fixes* (also known as *Shifting the Burden*), *Escalation, Limits to Growth,* and a combination of the latter two.

We use archetypes for understanding and modeling security aspects (needs, problems, actions) in the context of an enterprise that uses computers/information technology systems for running its business and needs to ensure the security of its information, services, and/or systems. We are representing and simulating security-

related organizational behavior and trends and using archetypes for documenting and understanding the domain, the problems, and their potential solutions. Mental models might be able to handle archetypes in isolation, but for the entire system (which contains combinations of such archetypes) mental models are not adequate due to the complexity, non-determinism, and uncertainty of the system. Computer simulation is in fact already recommended in [Sen94] for extending one's grasp of archetypes.

1.4 The Model

For our model, we employ the continuous modeling feature of the Extend simulation environment [Ima05]. This is a graphical simulation tool that focuses on the levels of holding tanks and their inputs and outputs, governed by constants, equations, delays, and random values. (A screenshot of a holding tank and its inputs and outputs can be found in Appendix II.) The level of each holding tank changes at each simulation step, and a typical simulation run can consist of hundreds or even thousands of such steps. The result is an easy-to-use way to set up and numerically solve systems represented by a series of differential equations. The feedback loops stressed by system dynamics and archetypes can easily be represented by a holding tank whose output is connected to its input. Thus, continuous modeling with Extend is a good fit for the system dynamics modeling approach described above.

Out model consists of approximately 350 Extend basic "blocks", such as constants or holding tanks. We outline it here, with complete details left for Appendices II and III.

In the model, staff-hours (of the system administrators) can be allocated to various tasks related to the security of a typical system. We model the following seven countermeasures:

- **"Firewall Efforts."** Overseeing and maintaining the system's firewall.
- **"Antivirus Efforts."** Maintaining the system's antivirus software, keeping it updated, resolving user issues related to the antivirus.
- **"Intrusion Detection System (IDS) Efforts."** Maintaining the IDS, installing new signatures, resolving alarms.
- **"Encryption Efforts."** Maintaining the system's encryption software.
- **"Enforcement Actions."** This includes tasks such as: scanning for and fixing configuration vulnerabilities, which are effectively "doors" to the system that were inadvertently left open; monitoring the users to prevent unsafe practices, such as downloading viruses or using "weak" passwords which are easily guessed; applying proper access control to prevent unauthorized use; and more generally, devising and enforcing a company security policy. See [Dan04] for more on these tasks. All of these require no additional hardware or software *per se*, only a great deal of attention from the support staff (or system administrators).

These appear as the five most prevalent "security technologies" used in Gordon's survey ([Gor05a]) of 700 corporate, governmental, and academic institutions, where we have subsumed Gordon's "Access Control Lists" under our term "Enforcement Actions." To these five we add a task familiar to any computer user:

- **"Software Patches."** Downloading and installing patches to correct vulnerabilities in the operating system(s) and applications; resolving problems caused by patches.

Lastly, we consider a somewhat different approach that has only recently been discussed by the security community:

- **"Tolerance Measures."** This includes designs to *tolerate* an attack (rather than prevent or detect it), even if it succeeds. Multiple layers, graceful degradation of performance, and (in some instances) backups are all tolerance measures.

In our current model, the effectiveness of each countermeasure is a factor only of the countermeasure's presence or absence (implemented as a series of Y/N switches in the model) and the number of staff-hours per machine allocated to the corresponding task. Although the IDS and firewall seem independent of the system size, additional machines will mean additional alarms, which will require more attention. Additionally, the system has an overall vulnerability measure, which is reduced by the number of staff-hours per machine allocated to enforcement actions and software patches.

The attacks on the system are divided into two categories: **"Simple" (or "kiddy-script") attacks** tend to rely on known vulnerabilities and require little action from the attacker other than downloading and running the attack. **"Sophisticated attacks"** may involve finding new vulnerabilities, can often defeat many countermeasures, and usually come from a single knowledgeable attacker, such as one who might actually write the "kiddy-scripts" of the former category. While viruses, which are the

costliest type of attack according to the respondents of [Gor05a], are written by some very sophisticated attackers, an existing virus propagates in well-understood ways and can be easily defeated by the proper countermeasures; we thus include viruses in the "simple attack" category.

For both categories (simple and sophisticated), a specified number of attacks are considered to be attempted against the system each day. (Alternatively, the simulation can also be set to add some random variation to the specified number of attempts.) Given the effectiveness of each of the various countermeasures, and the system's vulnerability (or lack thereof), a fraction of those attacks will succeed. The primary outputs of our current model, then, are the numbers of "successful simple attacks" and "successful sophisticated attacks." Note that a result of "*n* successful simple attacks" may not appear as *n* separate incidents. Several of these may exploit the same vulnerability, turn out to be variants of the same virus, and so on. For now, the number of successful attacks should be taken only as our metric of the quality of countermeasures versus attempted attacks.

1.5 Thesis Structure

The remainder of this thesis is structured as follows: Chapter 2 introduces the *Symptomatic Fixes* (or "*Shifting the Burden*") archetype; describes one instance of it in computer security as we have modeled it; discusses the results of several different simulations based on it; and considers how this archetype might apply elsewhere in security. Chapter 3 goes through a similar approach with the *Escalation* archetype. Chapter 4 introduces the *Limits to Growth* archetype, whereupon an instance is

described that describes a combination of *Limits to Growth* and *Escalation*. Chapter 5 outlines related work, and Chapter 6 gives conclusions and some future work. This is followed by Appendices I, II, and III, a bibliography, and finally a list of this author's publications and submissions.

Chapter 2: *Symptomatic Fixes* Archetype

2.1 Symptomatic Fixes Description

In this archetype, the symptoms of a problem are observed. Rather than analyze the root cause of the problem, the manager (or "decision-maker", or "actor") attempts to fix the symptom. This "shifting of the burden" from the problem's actual cause to its symptom often distracts the manager from the former; it can also mask the symptoms of the original problem, making it more difficult to diagnose.

Armed with an understanding of this archetype, a manager will consider the possibility that the most readily apparent solution may not ultimately be the best one. Instead, time must be taken to analyze, and only then properly treat, the root cause.

For a simple illustration in computer security, we paint a scenario in which a company's computer system (or just "system") is continually falling prey to successful attacks known as "kiddy-scripts." These attacks are launched by novice attackers, and generally only succeed if the system contains vulnerabilities such as software that is not up-to-date. The successes of these attacks should be seen as a symptom of a deeper problem. Reducing the system's vulnerability to thwart these attacks could be considered a fundamental solutions; such a fundamental solution would include the frequent installation of software patches. It is possible (in fact, likely) that implementing such a solution properly will take time and thus not yield dramatic gains very quickly; in the long run, however, positive effects of this solution will be observed. We choose software patches as one action that can be taken to

reduce overall system vulnerability vis-à-vis kiddy-scripts, though it is certainly not the only action.

Alternatively, it is all-too-possible for a company to instead view the successful attacks as the only issue here and therefore install an Intrusion Detection System (IDS) to detect the occurrence of these attacks – a symptomatic fix. The company's support staff (or "system administration staff") is then too distracted from installing patches. In time, many new vulnerabilities will be discovered in the software run by the system; once published, these will be exploited by new "kiddy-script" attacks. Invariably, a certain percentage of attacks do evade an IDS, and thus, as the known vulnerabilities in the system increase, the number of successful attacks will also increase, despite the company's continued efforts to install, maintain, and improve their IDS. These effects are displayed in Figure 1, an "influence diagram" showing the effects of given variable on one another over time.

In this diagram, we begin in the center with the problem symptom of successful simple attacks. In the loop beneath the symptom, we see the fundamental solution: increased successful simple (or "kiddy-script") attacks cause an increased need for the fundamental solution of applying software patches, and, in fact, applying this solution will reduce the problem symptom. Such a loop can be described as "more of A leads to less of B leads to less of A, and so on until equilibrium is reached", and is known as a "balancing loop." Alternatively, the symptomatic solution is found in the loop above the problem symptom. If we focus on this loop itself, it appears to offer the same advantages as the fundamental fix, sometimes more easily or more rapidly in the short term (though this is not indicated in the influence diagram).

Unfortunately, though, we also see that an increased use of the IDS can increase a side effect: the distraction of the support staff from other tasks, including patch application. This, of course, reduces the chance of a fundamental fix being applied. Starting at the top of the diagram and proceeding around its periphery clockwise, we see: increased IDS efforts leads to an increase in support staff distraction, therefore less patches are applied; the problem symptom will re-emerge, and more of the symptomatic fix will be attempted. This loop can be described as "more of A causes more of B causes more of A, and so on", and is known as a "reinforcing loop." [Wol03] includes this archetype under his more generic term "Out-of-Control", as a balancing loop is desired to control the problem symptom, but it is not obtained.

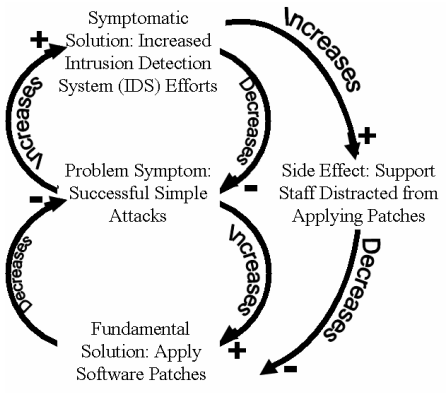

Fig. 1. Influence Diagram for a "*Symptomatic Fixes*" Situation.

2.2 Simulation Setup

To see quantitative results, an Extend model was used simulating a system containing on the order of 200 machines, sustaining 100 simple attacks per day. A certain percentage of these attacks will be defeated by an IDS (and depending on how

well the IDS has been maintained), and a certain percent will be defeated if the system's software is well-patched. Note that even if we say "*n*% of the attempted attacks succeeded", the system's users may not observe for 100 attempted attacks, *n* separate failures, as many of these attempts might target a small set of specific vulnerabilities and exploit them in the same way. Similarly, no single countermeasure should be expected to reduce the attack success rate to 0 by itself, as there are enough different types of attack that any single countermeasure can be defeated. We use the percentage of successful attacks only as a measure of the system's defenses and vulnerabilities. It is assumed that the software of this system is initially patched partially; therefore there is room for improvement if further patching is undertaken, while a loss will be felt if patching is ignored (as the discovery of new vulnerabilities will bring the software's status from "partially patched" to "mostly unpatched.") The model was executed for the equivalent of 6 months (real time) with different scenarios. (Each execution of this type runs in under 30 seconds on a conventional Pentium III computer running Windows 2000 Professional.)

For examining the effect of different effort allocation to the fundamental and symptomatic solution, we executed the simulation for four scenarios s1, s2, s3, and s4. In all four scenarios, the system is under pressure for the first d_1 days while the rate of successful attacks rises. This is due to the discovery of additional vulnerabilities. On day d_1+1, however, the company embarks on some course of action. Here we chose $d_1 = 9$, to demonstrate the effects over several days of taking no action at all.

In our first scenario, "s1", from day d_1+1 onwards, the company has its support staff dedicate a certain number of staff-hours per day to installing software patches to all the system's computers. This effort is held constant throughout the six-month period. The "if-then" rule that describes the organization's efforts in this scenario is given by:

IF: (Day> d_1)

THEN: Staff-Hours for Patches := x_1.

For the hypothetical situation that we are modeling, we considered 3 staff-hours a reasonable value for x_1 given the description of our system. This is considered the "fundamental fix" scenario, or the "solution" to the *Symptomatic Fixes* archetype.

In our second scenario, "s2", the company deploys an IDS on day d_1+1. For the next 170 days, efforts are gradually increased to maintain and improve the IDS: as new attacks are discovered, new plug-ins are added; as a consequence, more alerts that are signaled by the IDS must be analyzed, requiring more effort (although some of them might be just false alarms). In an attempt to keep the IDS functioning well, the company increases its IDS efforts with the following rule:

Begin with y_0 staff-hours for the IDS.

FOR: every day

IF: (Successful Attacks today > Successful Attacks two days ago)

THEN: increase staff-hours for IDS by y.

We have assigned the values y_0 = 1.5, y = 0.03. (This will lead to a gradual increase from moderate IDS effort at day ten to a strong IDS effort of approximately seven staff-hours by the end of the simulation.) Meanwhile, as IDS efforts increase,

less efforts are available for patches: *Staff-hours for Patches := 4 – Staff-Hours for IDS,* to a minimum of zero. We consider this our case of "increasing efforts to the symptomatic fix while decreasing efforts for the fundamental solution", or a strong instance of the "problem" archetype.

Our third scenario, s3, takes this a step further: as of day d_1+1, the same IDS efforts are made as in s2, but no patch efforts are made at all. Here we interpret the increasing side-effect loop in Figure 1 as the strengthening over time of the "mental barrier" (as [Wol03] calls it) that prevents consideration of the fundamental solution. Additionally, the side-effect loop is common for this archetype but not required, see [Sen90]. In any case, s3 is an even more extreme case of the problem archetype for *Symptomatic Fixes.*

Lastly, our fourth scenario s4 considers an alternative solution, one which the archetype literature concedes as sometimes viable. If the company understands its priorities, then it may be possible to use *both* the fundamental solution *and* a small dose of the quick fix. This would be codified by the following rules:

Staff-hours for Patches := 4 – Staff-Hours for IDS, as before.

The difference is the rule for IDS efforts:

Begin with y_0 staff-hours for the IDS.

FOR: every day

IF: {

(Successful Attacks Today > Successful Attacks two days ago)

AND (staff-hours for IDS <= z) },

THEN: increase staff-hours for IDS by y.

The value of y is the same 0.03, but y_0 is now reduced to 0.2. As in s1, we assume that a proper effort for patches can not be made with less than three staff-hours, so we set z to 1. s4 can thus be described as "symptomatic fix supplementing the fundamental fix." (Note that no "burden" is being shifted *per se* if the company understands what is fundamental and what is not.)

2.3 Results and Discussion

The primary outputs of these four scenarios, i.e. number of attacks successful per day, are plotted for comparison in Figure 2. We can also integrate under the curves of Figure 2, giving us the number of cumulative successful attacks for each of the four scenarios; these will be displayed in Table I below.

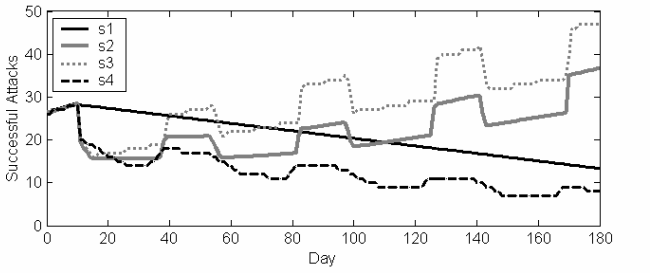

Fig. 2. Successful Attacks per Day for the Four *Symptomatic Fixes* Scenarios

Several important features can be observed in Figure 2. Firstly, when comparing the fundamental solution (s1) to the symptomatic fixes (s2 and s3), we see that the symptomatic fixes appear to do a much better job initially (e.g. looking at Day 30, s2 and s3 are approximately 10 successful attacks lower than s1), but by the end of the simulation period, the fundamental solution is far more successful: at Day 180, s1 is

22 successful attacks lower than s2, and 32 lower than s3. This demonstrates a common pattern in the performance of symptomatic fixes – while the symptomatic fix can cause temporary drops in the problem rate, the overall trend over time is for the problem rate to increase. (Diagrams similar to Figure 2 are seen in describing this archetype in [Sen94] and [Bra02].) While the security staff is distracted by the rises and falls in the performance of the IDS, the system's current software vulnerabilities, as well as those newly discovered, are neglected, leading to a rise in the percentage of attacks that are successful. The overall trend is a linear increase; this is not surprising, as we have modeled the vulnerabilities in unpatched software as increasing linearly in time. Comparing s1 against s2 and s3 also stresses the importance of behavioral monitoring over time. Were we to stop the simulation after one month or so, our conclusions would be very different as to what measures are most effective!

Focusing on s3, we see that it presents an even more extreme case of s2's failures, as the patch efforts have been eliminated entirely. Lastly, we turn our attention to s4. Recall that s4 begins with less IDS efforts than s2 and s3; it therefore appears initially to allow more attacks to succeed, e.g. at Day 16, s4 is 2 successful attacks higher than s3, and 3 higher than s2. However, by the end of the simulation period, s4 is clearly the winner in reducing successful attacks. Notice as well the height for the "waves" of symptomatic fixes: they are greatest in s3, smaller in s2, and smaller still in s4; this height represents the degree of the "crisis/fix" pattern, which is lowest when the proper application of fundamental fixes prevents crisis action (s4), and greatest when no fundamental fix is present (s3). Lastly, while s4 clearly prevents more attacks than

s1, notice how they approach each other asymptotically – in the long run, adding the symptomatic fix will cease to provide any good beyond the fundamental solution.

We now turn our attention to the effort required in each of these scenarios. s1 consisted simply of a constant 3 staff-hours per day for patches, and nothing else. Figure 3 shows the efforts of the support staff, in staff-hours per day, invested in s2, in which IDS efforts decreased patch efforts.

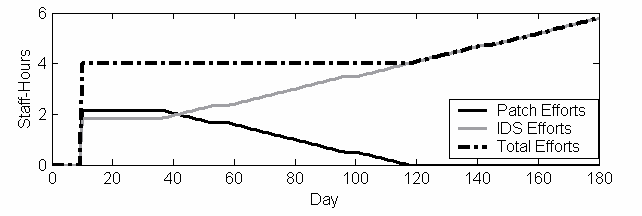

Fig. 3. Efforts per Day for s2, the IDS-Decreases-Patches Scenario

Notice how patch efforts decrease steadily until approximately Day 120, at which point they stay at zero for the remainder of the simulation. Until Day 120, any efforts for IDS came out of efforts for patches, so total efforts were constant; after Day 120, the total efforts are all IDS efforts. Figure 3 further highlights the attractiveness of the symptomatic fix, as the initial IDS effort requires less staff-hours per day (less than 2) than what would be required of a fundamental fix (a steady 3 for s1). In the long run, however, staff-hours are continuously added to the IDS effort in an attempt to raise its results; by the time six months have passed, the company realizes that it is investing 6 staff-hours per day into the IDS. We can also integrate the curves in Figure 3 to measure cumulative effort of the simulation period, to be shown in Table I.

In s3, the only efforts present are those for IDS. These are shown in Figure 4.

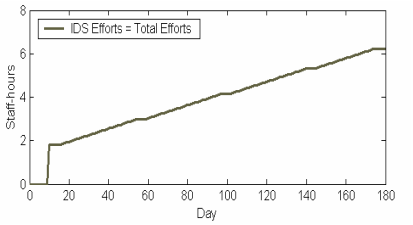

Fig. 4. Efforts per Day for s3, the IDS-only Scenario

Observe that by Day 180, approximately 6.25 staff-hours are being used for IDS efforts. In s2 (see Figure 3), that number was only approximately 5.8. The same rule produced both figures: "increase IDS efforts every day that successful attacks are higher than they were two days ago." Compared to s3, s2 allowed for some patches as well, so there were less days when this trigger occurred, therefore less IDS efforts were demanded over the course of the simulation.

Lastly, Figure 5 displays the efforts of s4, which combined IDS and patch efforts with an emphasis on the latter.

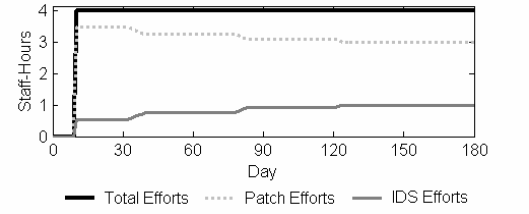

Fig. 5. Efforts per Day for s4, the IDS-Supplements-Patches Scenario

Notice how few increases are made to IDS efforts. Again, this happens because the trigger condition of successful attacks being too high is very rarely met, due to the appropriate patching strategy.

We now compare all four scenarios in terms of their cumulative effort and cumulative successful attacks, as displayed in Table I.

Considering cumulative values, we see indeed that cumulative successful attacks are lower for s1, the fundamental solution, than for symptomatic fix scenarios s2 and s3.

Table I: Cumulative Successful Attacks and Efforts for All Four Scenarios

	Cumulative Successful Attacks	**Cumulative Efforts (Staff-Hrs.)**
s1	3833	513
s2	4007	740
s3	5232	689
s4	2345	684

Noticing that s1 requires over 100 less staff-hours' worth of effort than s2 or s3, we see that in the long run, the fundamental solution is not only more effective than the symptomatic fix; it is less costly as well. The only question remaining is in comparing s1, "fundamental solution alone", with s4, "fundamental solution combined with symptomatic fix." A company will have to decide for itself whether the additional 151 staff-hours of efforts are worth the reduction in 1500 successful attacks. How such calculations are made is touched upon in related work, below. In any case, simulation allows the company to consider the effects of its actions, and choose its optimal course with these effects in mind.

By analogy with these results, when other variables of interest in the system have a similar evolutionary trend, the *Symptomatic Fixes* archetype might be manifesting itself. In that case, the situation must be diagnosed and the real cause and the corresponding solution must be examined; this solution has to be applied, thus fixing the real problem. Of course, the results of this shifting must be monitored over time,

to make sure that the diagnosis was correct and that the solution was correctly implemented.

Lastly, the above simulations show the applicability of the model as a decision tool, by allowing one to see the effects of different proposed solutions before implementing them. In the example presented here, the decision was regarding the allocation of effort to different security efforts (IDS and patches). The model might also be useful in exploring and making security policies, as well as for training security staff.

2.4 Other Instances of Symptomatic Fixes in Security

We have presented only one possible instance of *Symptomatic Fixes* here, and thus we have opened the door to many related opportunities. Our simulation model includes many security-related tasks not described here (such as user training, enforcement of the security policy, and maintaining tolerance measures such as backups, to name a few), and in place of the Patch Efforts described here, this simulation could be run with other tasks or some combination thereof, as well as considering more-sophisticated attacks. Just as different parties may see different tasks as "the" fundamental solution ([Sen94]), attacks of different sophistication may have different "fundamental solutions."

Additionally, [Sen90] finds that the best way to describe the history of a particular company's strategies is by combining the *Symptomatic Fixes* archetype with another archetype, namely *Limits to Growth*. Thus, the applicability of this combination and other archetype combinations should be considered in computer security as well.

A variant on *Symptomatic Fixes* described in [Sen90] and [Sen94] is known as *Shifting the Burden to the Intervener,* in which the fundamental fix involves the internal actors repairing problems, and the symptomatic fix involves outsiders. This brings to mind some sentiments in the security community about security being incorporated into system design at each step of the process, rather than ignoring security and relying on an expert to add security features shortly before release or deployment.

Lastly, there has been much discussion in the security community (see [Hun06]) regarding whether better security behavior should be taught to the users of a system, or placed entirely on the shoulders of the system administrator. Similarly, in a system where the roles of system administrator and security officer are divided, the interactions between them may follow archetypal patterns. We had begun to document anecdotal accounts of such interactions, and our model leaves room to add detail to its human-factors portion of the model, including the interactions between users, system administrators, and security officers. *Shifting the Burden to the Intervener* could thus shed light on these human interactions.

For additional information of *Symptomatic Fixes* as it pertains to security, please see [Ros06b], from which this chapter was excerpted.

Chapter 3: *Escalation* Archetype

3.1 Escalation Description

In the *Escalation* archetype, each of two parties makes efforts and achieves results towards reaching its own well-defined goals. However, each party desires greater results than its counterpart. Thus, each party continues to increase its efforts, with neither party achieving dominance for an extended period. This can theoretically continue *ad infinitum*.

As an instance of this archetype in security, we investigate the action-reaction effects of attacks on an organization's computer system and the organization's attempts to better defend its assets, all the while advertising its strengths in an attempt to attract more business. We begin with a company that spends little on security measures, but sustains few attempted attacks because it's not a very well-known or worthwhile target. While some simple "kiddy-script" attacks blindly go after any available computer system and can be seen as the ever-present "attack noise", other simple attacks (such as a "Zombie DDoS", see [Gib02]) are consciously directed at an organization by an attacker. These are more likely if the organization is better-known. Furthermore, an organization will be targeted by sophisticated attacks if its assets are valuable (e.g. credit card numbers stored on its servers), or if its defenses are considered formidable, in which case breaching them poses a worthwhile challenge.

We suppose that the organization decides to attract new customers by increasing its security spending and advertising its new security strength. As the prominence

and/or asset desirability of the organization rise, the motivation to attack its system is increased, raising both the quantity and sophistication of attempted attacks. To counteract these, the company increases security spending again. Alas, this furthers the motivation to attack, leading to another increase in attempts. This process can continue for several more rounds.

These effects are displayed in Figure 6, an influence diagram showing the effects of given variable on one another over time. (Similar influence diagrams are drawn for archetypes in [Bra02].)

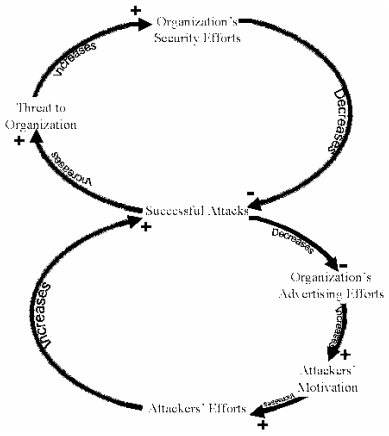

Fig. 6. Influence Diagram for *Escalation*

The upper loop in Figure 6 reads as follows: "Increasing the organization's security efforts will decrease the number of successful attacks against it. An increase in successful attacks leads to a greater threat to the organization. The greater the threat, the more security efforts will be added to counter it." Thus, if the attackers' efforts are constant, we would observe the following behavior: increased security

efforts will decrease the number of successful attacks, decreasing the threat to the organization, decreasing the need for additional security efforts. This forms a "balancing" or "negative" loop, as after several rounds of such behavior, no further efforts will be required.

A similar pattern is found in the lower loop: "Increased successful attacks cause the organization to advertise less. (We assume the resources that would have been funneled into advertising are now needed to recover from all of the attacks.) Advertising efforts increase the motivation to attack the organization, leading to more efforts on the part of the attackers, and therefore more successful attacks." Thus, the attacker behavior in and of itself should also form a balancing loop, as enough successful attacks will prevent any advertising, at which point the organization is no longer a very visible or worthwhile target, so attack efforts are not increased again.

However, in our scenario, both the organization and the attackers respond to one another, violating the assumptions we had made for balancing loops. Traversing the outermost loop of Figure 6 describes the overall behavior: an increase in the organization's security efforts increase its advertising efforts (or otherwise raise its prominence and asset desirability), increasing the motivation and therefore the efforts to attack the organization, leading to a rise in successful attacks. The organization feels threatened and therefore increases its security efforts, and the spiral continues from there. As both the organization's and the attackers' efforts continue to increase in time, this forms a positive loop. The number of successful attacks, however, reflects the ratio of attackers' efforts to the organization's security efforts, and thus should exhibit stable oscillations. [Wol03] describes this archetype as "Relative

Control", as each party's balancing loop is used in an attempt to gain control over the relative quantity "success of one party / success of the other party."

3.2 Simulation Setup

Clearly in our case, the number of successful attacks becomes the barometer of "success of attackers compared to success of defenders." Increased efforts by attackers over time can be modeled by an increasing number of attempted attacks, both simple and sophisticated. The organization's efforts can be fulfilled by: introducing countermeasures that were not previously present; changing the allocation of support staff-hours to various tasks; training the support-staff (which, to a point, increases their effectiveness); and increasing the staff-hours available for security tasks. The latter may require hiring in the long run, but in the short term may often be achieved simply by encouraging overtime, reassigning personnel within the company, etc.

In the simulation scenarios presented here, we have simplified by limiting the organization to one action, namely increasing staff-hours, and did not include other actions. We assume that all countermeasures are present, but they all begin with inadequate support staff. In time, increasing the staff-hours to each task will result in a greater number of attacks not successful. We have further simplified by scripting the actions of both the organization and the attackers as an automated series of "If-Then" rules, so the simulation runs without external intervention. The rules we use are based on our assumptions of how a company in such a situation would behave,

and they quantitatively capture the qualitative behavior described in Figure 6. These rules are as follows:

The organization decides to increase efforts:

FOR: e*very x_1 days*

IF: (Successful Simple Attacks > x_2)

THEN: increase staff-hours allocated to Antivirus, Firewall, IDS, Enforcement Actions, and Software Patches by $\vec{w} = \{w_1, w_2, w_3, w_4, w_5\}$, respectively.

These tasks begin with $\vec{w}_0$ staff-hours allocated at the start of the simulation.

These countermeasures and vulnerability-reduction tasks are very effective at preventing or detecting simple attacks. Faced with sophisticated attacks, however, their effects are diminished: the antivirus does not address these attacks, which aren't viruses; the IDS and firewall can sometimes be deceived; and enforcement actions and software patches can only reduce known vulnerabilities, whereas the sophisticated attacker may discover and exploit new vulnerabilities. Thus, the company responds to sophisticated attacks in a different way than to simple attacks:

FOR: every x_1 days

IF: (Successful Sophisticated Attacks > x_3)

THEN: increase staff-hours allocated to Encryption by v_1 and Tolerance by v_2. Tolerance and Encryption are allocated $\vec{v}_0$ staff-hours at the beginning of the simulation.

We assume that these countermeasures are no less effective against sophisticated attacks than against simple attacks. Today's commercial encryption is believed to be

unbreakable by any private individual with a handful of computers, no matter how clever, and tolerance works despite the success of the attack.

As some tasks may require more staff-hours than others to be done well, different numbers can be specified for each task. In any case, decisions to increase staff-hours are implemented as follows: *Any increase in staff-hours requires a d_1 day delay to reassign personnel. d_2 days after the increase occurs, the company advertises its added security efforts.*

This leads the attackers to launch additional attacks, according to the following assumed behavior: *Begin with y_1 simple attacks. Any day that advertising is present, increase the simple attacks by y_2%.*

Simple attacks can be increased rapidly, as this merely requires directing automated "kiddy-scripts" against the system. The number of sophisticated attacks, however, grows at a different (generally slower) rate: *Begin with y_3 sophisticated attacks. Any day that advertising is present, wait d_3 days as sophisticated attacks are prepared; then increase the sophisticated attacks by y_4%.*

In our execution, the number of simple attacks attempted is given by the above rules. To allow for some randomness, we chose to let the number of attempted sophisticated attacks vary by a (Gaussian) standard deviation of 5%. Additionally, if the number of successful sophisticated attacks is found to be between 0 and 1, then a random number is drawn to determine if the attack succeeds.

We simulate a system of approximately 200 machines, choosing a simulation period of six months (180 days). Keeping these numbers in mind, we have run the simulation with the following values: $x_1 = 7$, $x_2 = 4$, $x_3 = 1$, $\vec{w}_0 = \{1.8, 2.2, 6.0, 2.4,$

2.4}, $\vec{w}$ = {1.1, 1.3, 3.7, 1.5, 1.5}, $\vec{v}_0$ = {0.72, 0.87}, v_1 = 1.8, v_2 = 2.2, y_1 = 20, y_2 = 29.8, y_3 = 0.6, y_4 = 9, d_1 = 14, d_2 = 5, and d_3 = 2. In our opinion, these values, used with the above rules over a 180-day period, describe a linear progression from minimal attention to complete dedication vis-à-vis staff-hours for security tasks.

3.3 Results and Discussion

Successful attacks per day are used as our measure of "organization's efforts vs. attackers' efforts"; the results are shown in Figure 7.

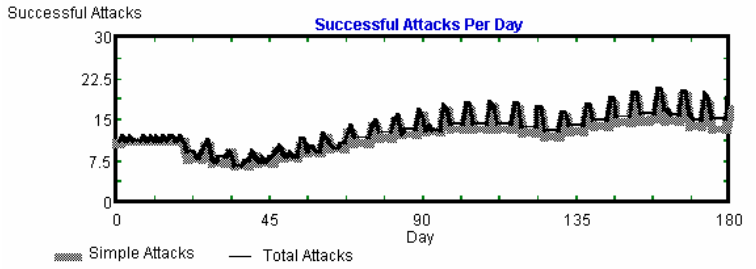

Fig. 7. Successful Attacks per Day, First *Escalation* Scenario

Certainly from Day 90 onwards, the system reaches a sort of equilibrium, as successful attacks hover around 13. This is a result of the matched opposing efforts of the organization and the attackers. Yet while the overall metric (i.e. successful attacks) does not change much, both efforts are ongoing. Figure 8 shows the efforts of the organization, in staff-hours per day dedicated to security tasks.

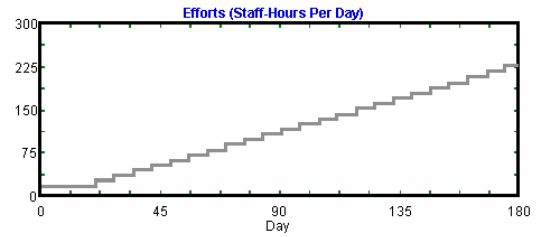

Fig. 8. Staff-Hours per Day, First *Escalation* Scenario

According to the rules and the specific values of the variables described above, the first decision to increase staff-hours occurs at Day 7, and is implemented fourteen days later; thus, the first increase is seen at Day 21. After that, increases can occur as often as every seven days: decisions to increase are made every seven days, and previous weeks' decisions will be implemented while waiting the fourteen days for this week's implementation. Overall, the organization's efforts grow, fairly linearly, up to approximately 220 staff-hours per day. Assuming eight-hour days, this translates into twenty-seven people, which is high but not unreasonable for a system of 200 machines. Of course, this growth is matched by the increase in both simple and sophisticated attacks. Figure 9 shows the attempted simple attacks.

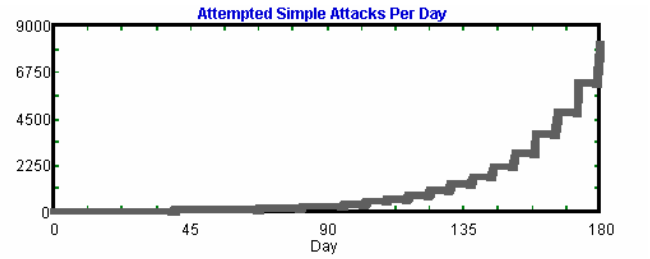

Fig. 9. Attempted Simple Attacks per Day, First *Escalation* Scenario

The number of attempted simple attacks can rise rather drastically, as this only requires that novice users unleash their automated processes against the system. This reaches almost 9,000 attempted attacks on the entire system per day, or 45 attempts, including viruses, per machine. While we stress the *behavioral trends* here much more than the specific numerical results, these numbers can be "reality-checked" against some empirical findings involving honeypots. [Dac04] observed attacks from 6,285 IP addresses over four months, averaging over two new attack sources per hour. Similarly, [Pou04a] observed 28,722 new attack sources over sixteen months. [Pou05] found 924 attack sources per day in Germany, and [Pou04b] mines a year of collected data and concludes with a very conservative estimate of 753 attack tools available to simple attackers. In light of these results, and considering that in our case, the organization has "begged for attacks" by advertising, our numbers seem fairly realistic (or in agreement with the existing empirical data.)

Figure 10 shows the daily average of attempted sophisticated attacks.

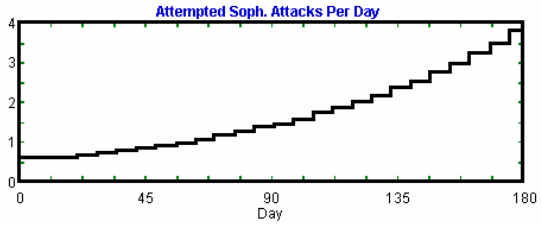

Fig. 10. Attempted Sophisticated Attacks per Day, First *Escalation* Scenario

The growth of attempted sophisticated attacks is much slower, as it requires higher human effort and expertise.

We also observe that the linear increase in the Organization's Efforts (i.e. staff-hours) can balance out the exponential increase in Attackers' Efforts (i.e. attempted

attacks). This is the case because in our model, a linear growth in countermeasure effectiveness leads to a lower percentage of successful attacks – an exponential decline.

Our simulation resulted in an overall relatively constant average number of successful attacks, an equilibrium of sorts between the results of the two striving parties (organization and attacker). Given these results, an organization may attempt to "beat" this escalation by increasing its efforts beyond the values given here; or it may consider cutting costs by reducing its efforts, if the results will be the same. We therefore ask how this equilibrium is affected if we modify the values representing the amount and frequency of increases in security efforts.

Firstly, we ask how much can be gained by the organization if it increases its efforts a bit more. In this scenario, when staff-hours increase, they increase not by $\vec{w}$, but by $1.1\,\vec{w}$ instead. Figure 11 shows the results.

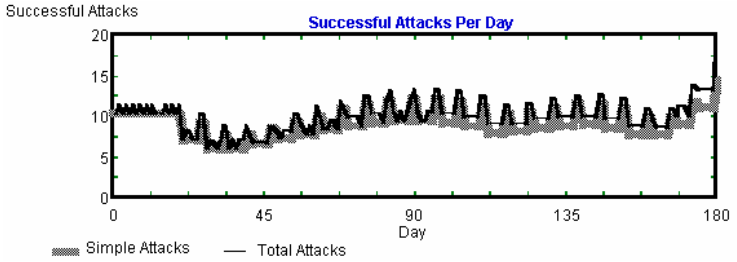

Fig. 11. Successful Attacks per Day: Results of 10% Increase in Efforts

Compared to Figure 7, Figure 11 has a similar overall shape, but the average number of successful attacks hovers around 10, versus the 13 of Figure 7. Thus, by increasing efforts by 10%, the organization can reduce its equilibrium by 3 successful attacks.

Secondly, we ask how much is lost if the organization does not increase its efforts as much. Now the increase in staff-hours is not $\bar{w}$, but $0.9\,\bar{w}$ instead. Figure 12 shows the results.

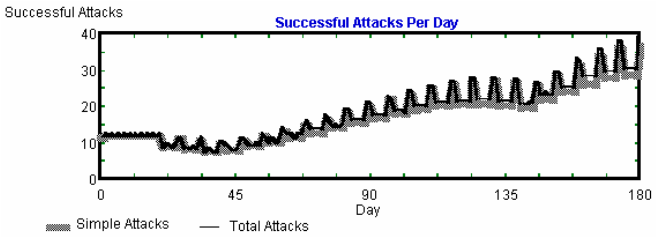

Fig. 12. Successful Attacks per Day: Results of 10% Decrease in Efforts

Suddenly, the equilibrium has risen to approximately 25 successful attacks. (It is not even clear whether an equilibrium exists by the end of the period, given the graph's steep climb from Day 140 onwards.) Thus, for a company considering changing its efforts, simulation here has shown that a small increase in efforts will not do much good, but a small decrease in efforts will cause much harm. This echoes [Sen90]'s discussion of "leverage", the large effects of small changes. A benefit of simulation is thus demonstrated.

Lastly, we test the sensitivity of this equilibrium by modifying a different value. Instead of the amount of the efforts' increase (i.e. $\bar{w}$), we change the frequency of increased efforts. x_1, the delay between increases (if increases are required), had been 7. We now change it to 6, and run the simulation. Intuitively, since the organization's reaction is more frequent, we expect the number of successful attacks to decrease. The results are shown in Figure 13.

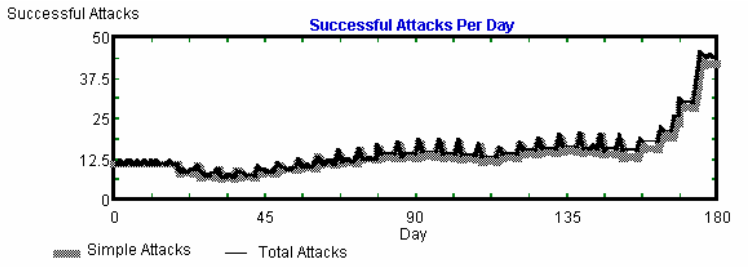

Fig. 13. Successful Attacks per Day: Results of More Frequent Efforts

Clearly, by Day 160, the equilibrium has been upset. The increase in attempted attacks is outpacing the increased efforts of the organization, and successful attacks begin to climb. This is due to the fact that the organization's more frequent efforts consisted of security spending followed by advertising, which attracted more frequent efforts of the attackers that it could not match. This example illustrates not only the utility of simulation for predicting the effects of small changes, but also the benefits of simulation in revealing unexpected behavior. This example also demonstrates a systems concept: sometimes the best way to survive an *Escalation* scenario is to not react as often, even if a reaction appears necessary. [Sen90] gives a case study of two manufacturers of a new design of stroller, both of which are making a respectable profit margin on their sales. Then the manufacturers entered *Escalation*, lowering their prices in an attempt to raise market share. Little time passed before both manufacturers no longer had a profit margin. The risk of reaction (reduced profit margin) had not been weighed against the risk of no reaction (reduced market share), and perhaps a slower reaction may have offered the greatest overall gain. Similarly, we have demonstrated the possibility that an organization can be out-escalated by its

opponent. Simulation thus grants the would-be entrant into *Escalation* the opportunity to pause and consider such outcomes.

Returning to our security scenario, we had described the increase in attacks as due to the organization's advertising. While some companies (e.g. search engines) cannot exist without high visibility, our results behoove an organization to consider the effects of its advertising and whether they outweigh the risk of additional attacks. Additionally, the automated "if-then" rule for advertising used here was to advertise anytime an increase in efforts is made. While the influence diagram of Figure 6 indicates that enough successful attacks will prevent further advertising, our if-then rules had assumed that point was not yet reached in the system, e.g. it only occurs when successful attacks reach 60 or higher.

Alternatively, a rule can be constructed which states, "Advertise only if successful attacks are below a certain threshold." Such a rule is included as part of the *Escalation* behavior in Chapter 4.

All of the above scenarios involved automated rules to govern the choices of both organization and attackers. As an alternative, the model also allows for a rule of "pause the simulation whenever a certain condition occurs." In our case, then, pauses may be configured, for example, whenever the successful attacks (simple, sophisticated, or sum of the two) exceed a threshold value. The simulation then pauses, and the end-user of the simulation may consider making changes such as introducing a new countermeasure or increasing staff-hours before resuming the simulation. The behavior over time of the aggregated attackers can also be paused and adjusted by hand; this feature may be useful to security researchers, but for a

company considering the impact of various choices that it could make, the attacker behavior is out of its hands and would thus presumably be represented by automated rules.

Lastly, here we assume that the organization is free to increase its efforts without any additional constraints. Practically, such increases may carry risks other than those of increased attacks; such risks are described by another archetype, *Limits to Growth*, and are described in the next chapter.

3.4 Other Instances of Escalation in Security

On the *Escalation* archetype, [Sen90] lists the international arms race as the most obvious example of *Escalation*, and [Hof05] specifies an "information technology security arms race." This arms race consists of advances in attack technology, which necessitate improvements in security technology. For example, [Hof05] argues that "with the advent of binary differs . . . patching is no longer a viable defense strategy", and instead advocates recent advances in Intrusion Prevention Systems. But this "race" develops over the course of a decade or longer: see [Dwa05] for a timeline from the 1980s to today. Given the vast unpredictability of long-term innovation, this is hardly something a single organization can simulate to aid its decision-making; we have thus chosen not to model it here.

[Sen90] also suggests a generic solution to this archetype's woes: often there can be an agreement to reverse the cycle, as each party agrees to simultaneously "ease off." While this may succeed in international politics (as it arguably did in détente),

the notion of “we’ll use less security technology if you agree to attack our computers less” is obviously not applicable in this case, particularly when the anonymous attacks, attackers, and motivations are myriad. This option is therefore not considered in our scenario.

For additional information on *Escalation* in computer security, please see [Ros06a], from which this chapter was excerpted.

Chapter 4: *Limits to Growth* and *Escalation* Archetypes, Combined

4.1 Limits to Growth Description

In the *Limits to Growth* archetype, a growing action is applied, which leads to increased gains or results. These gains encourage further growth, forming a reinforcing loop. However, the gains soon reach some natural limit, at which point the limiting process places downward pressure on further gains. Despite continued growth action, the gains will plateau and, in some cases, decline.

As an instance of this archetype in security, we consider the effects of security demands on an organization's computer staff of a fixed size. Suppose that an organization has a certain number of employees dedicated to various computer-related tasks such as technical support, hardware maintenance and upgrades, and security-related tasks such as monitoring a firewall or an IDS, or maintaining antivirus software. Initially, the organization pays modest attention to security, but then decides to make some investment in it. Whether the investment includes purchasing equipment (IDS, encryption or antivirus software, and the like), security training, overtime, or higher salaries for employees who focus on security, it always involves reassigning personnel to security. Encouraged by the noticeable gains in security, further investments lead to more reassignments of personnel to security. This continues to be a good strategy until insufficient personnel are available for non-security tasks. At this point, numerous non-security-related technical problems arise

in the computer system, forcing the security personnel to pause their efforts as these problems are addressed. Reassigning more employees to security (or demanding more of the current security employees) will bring no further gains; in fact, the additional technical problems as well as the support staff's decreased efficiency from facing demands it can not meet may result in a decline in gains. An influence diagram for this situation is shown in Figure 14.

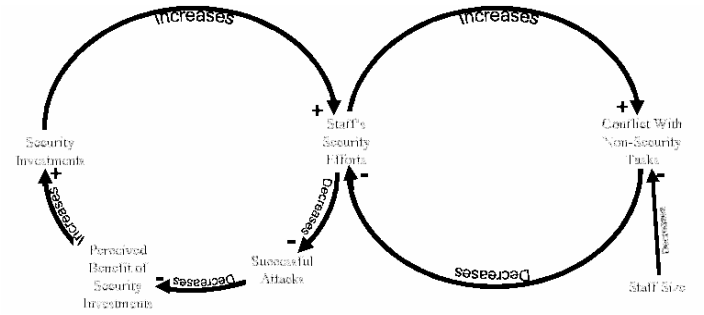

Fig. 14. Influence Diagram for *Limits to Growth*

Traversing the left side clockwise reads: "security investments increase the staff's security efforts, decreasing the number of successful attacks. More successful attacks would decrease the management's perceived benefit of security investments. More perceived benefits of investments leads to further investments." Reversing the double negative yields: "investments leading to efforts leading to gains in security (i.e. *less* successful attacks), increasing the perceived benefit of investments and therefore leading to further investments" – this is a reinforcing loop. The right-hand loop, however, describes how increasing the staff's security efforts can conflict with non-security-related tasks, due to a personnel shortage. As indicated by the upward arrow, this effect is decreased if the staff size is sufficiently large. Lastly, an increase of such

conflicts will cause problems that diminish the staff's security efforts. A balancing loop is thus formed, as security efforts will (unconsciously) decrease as long as the conflict of resources with non-security tasks is present. Given a constant number of attempted attacks, implementing this archetype should result in a continuous reduction of successful attacks (i.e. increase of gains for security investments) until insufficient personnel are available for other tasks; at that point, the number of successful attacks will cease to fall further, and may in fact begin to rise. [Wol03] includes this archetype in the category "Underachievement", as a reinforcing loop is desired for growth, but it is not successful.

The simulation model can incorporate this Limit to Growth with the following property: Some value p is the highest percentage of staff efforts that can optimally be reallocated to security with no ill effect. If total demand for security efforts exceed (p/100)*SysAdminCapacity, then the "effective" hours for security are given by the SysAdminCapacity, minus some constant k times the excess demand. In the simulation described below, we have used p = 23 and k = 1.2, believing these values to be a reasonable description of a typical system.

4.2 Combined Archetypes

While the use of an archetype can present a complex system in readily-grasped terms, a given scenario or story may not neatly fit into a given archetype. The general archetypes of [Sen90], [Sen94], and [Bra02] are unique only in that they have been frequently observed in diverse settings, and that they provide useful "building blocks" for other influence diagrams. For each given case study, [Sen94] recommends

beginning with the influence diagram of one easily-observed archetype (or simply a balancing or reinforcing loop), then "widening and deepening" the diagram by adding additional "loops" to describe the observed behavior. Thus, a combination of archetypes is often the simplest way to grasp a system's behavior when two or more different behavior patterns are exhibited simultaneously. (Such a combination, that of *Limits to Growth* with *Shifting the Burden*, can be found in [Sen90].)

Observe that both *Escalation* and *Limits to Growth* hinge on the organization's security investments and successful attacks; we thus connect their influence diagrams through these values. The resulting combined diagram is shown in Figure 15.

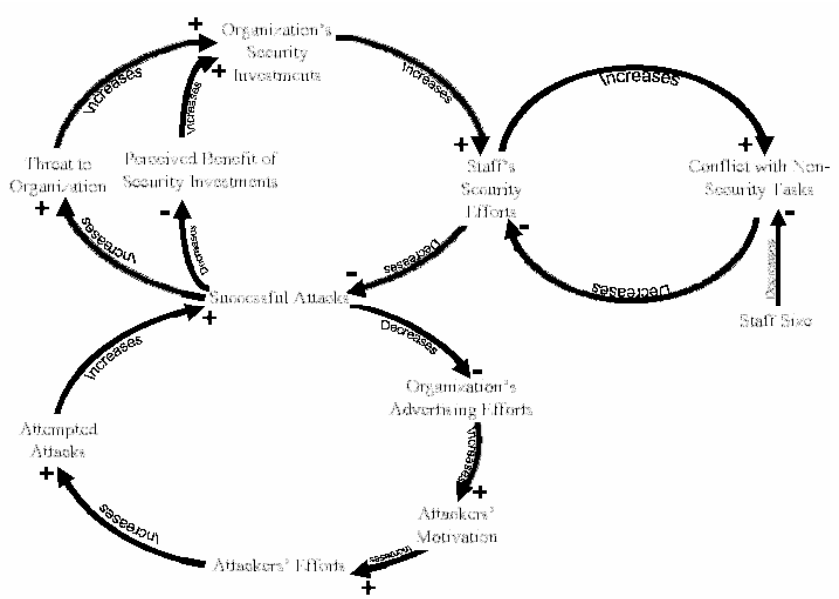

Fig. 15. Influence Diagram for Combined *Limits to Growth* and *Escalation*

Observe that the two influence diagrams largely address different issues, except for the upper-left-hand corner of Figure 15, which links successful attacks to security investments. While *Escalation* had assumed a "positive" effect (i.e. more successful

attacks increase the threat, increasing investments), *Limits to Growth* assumes a "negative" effect (successful attacks decrease perceived benefit, reducing investments).

In combining the two archetypes it becomes clear that both patterns may be true for different organizations with different cultures, or for different levels of management. Additionally, recall that an the influence diagram shows only "increases" and "decreases", but quantitatively some links may be stronger than others. Thus, both patterns may be present within a single organization; a visible shift from increases to decreases in investment, or vice versa, will occur at times when the weight of one pattern exceeds that of the other. For example, when the organization's management first invests in security, its perceived benefit is low, so further investments hinge on a reduction in attacks; later, security investments are believed an appropriate cure if successful attacks rise; finally, successful attacks may reach some upper limit at which point the management begins to lose its faith in investments and reduce them.

The overall trend of this combined archetype, when viewed in terms of successful attacks, will look as follows: a stable oscillation (due to *Escalation*) until security efforts exceed their optimal value (for the given staff size), followed by a rise in successful attacks (from *Limits to Growth*). At this point, several possibilities exist: the organization may continue (for a short duration) to advertise, leading to further attempted attacks; it may follow the "threat" pattern and push for more security investments; and/or it may follow the "perceived benefit" pattern and reduce security investments. Depending on these three options, the stable oscillation and rise in

successful attacks will be followed by either a leveling off or a rise in successful attacks; the former would occur if the organization halts both advertising and investments, keeping attempted attacks constant. The highest risk, leading to a significant increase in successful attacks, occurs if the organization continues advertising, raising the attempted attacks, as its continued investments cause more woes for its computer staff, further diminishing their effective efforts.

4.3 Simulation Setup

The behavior of the organization's management (which invests in security and demands staff-hours for it) and the aggregated attackers (who attempt the attacks) are then given by a series of rules (similar to those of Section 3.2), following the escalatory behavior described above. Here we demonstrate one possible outcome by assuming that the perceived benefit or "faith" in investments is held constant, and thus the decision regarding further investments is determined only by the threat to the organization. This decision is modeled by the following rule: *The simulation begins with an initial demand of w_0 staff-hours for security. Every x_1 days, { IF (Successful Simple Attacks > θ_1), THEN increase staff-hours demanded for "simple" security tasks by w. Additionally, IF (Successful Sophisticated Attacks > θ_2), THEN increase staff-hours demanded for "sophisticated" security tasks by v.} A delay of d_1 days is incurred for personnel reallocation.*

(The description of tasks as "simple" or "sophisticated", as well as the task-by-task composition of *w* and *v*, are unchanged from Section 3.2.)

The organization's advertising efforts are modeled by the following rule: *Every x_2 days, IF (Successful Simple Attacks < θ_3), THEN decide to advertise. A delay of d_2 days is incurred before the advertising occurs.*

Lastly, the aggregated attackers' response is modeled as follows: *The initial value of Simple Attacks Attempted / Day is a_0. Each day, IF (Advertising occurs), THEN a delay of d_3 days occurs as the word spreads and new attack tools are accumulated, where upon Simple Attacks Attempted / Day is increased by a%. The initial value of Sophisticated Attacks Attempted / Day is b_0. Each day, IF (Advertising occurs), THEN a delay of d_4 of days occurs as the word spreads and new attacks are engineered, whereupon Sophisticated Attacks Attempted / Day is increased by b%.*

We simulate a system of approximately 200 machines. We have chosen a period of six months (180 days) for our simulation. Successful attacks per day are used as our measure of "attackers' gains vs. organization's gains."

With these values in mind, we first simulated a "baseline scenario" characterized by the following values: $x_1 = 7$, $x_2 = 7$; $w_0 = 29.8$, $w = 9.1$, $v = 1.6$; $\theta_1 = 6$, $\theta_2 = 2$, $\theta_3 = 18$; $d_1 = 14$, $d_2 = 1$, $d_3 = 2$, $d_4 = 7$; $a_0 = 15$, $a = 26$; $b_0 = 0.6$, $b = 7$. These values describe, in our opinion, an organization's 180-day progression from minimal security efforts to full security efforts; a realistically aggressive advertising campaign; common delays for each action described; and a progression in terms of attack attempts from the minimal attack "noise" received by an inconspicuous organization to the high number that a prominent organization receives.

4.4 Results and Discussion

The results of this simulation are shown in Figure 16.

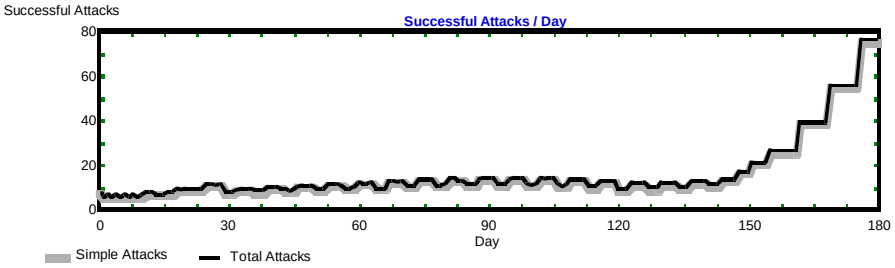

Fig. 16. Successful Attacks per Day, *Escalation* with *Limits to Growth*

The number of successful attacks seems to oscillate fairly stably until approximately Day 145, at which point it rises dramatically. Until Day 145, the number of successful attacks hovers at about 11, which is this system's equilibrium of escalation: security efforts, followed by advertising, followed by new attack attempts, followed by further security efforts. Around Day 145, however, the *Limits to Growth* archetype emerges: the demand for staff-hours exceeds the optimal load the staff can bear, the staff's performance deteriorates, and successful attacks rise. Note that successful attacks exceed θ_3=18, the organization's threshold for cessation of advertising, at approximately Day 155.

Correspondingly, the number of attempted attacks (simple plus sophisticated) is shown in Figure 17

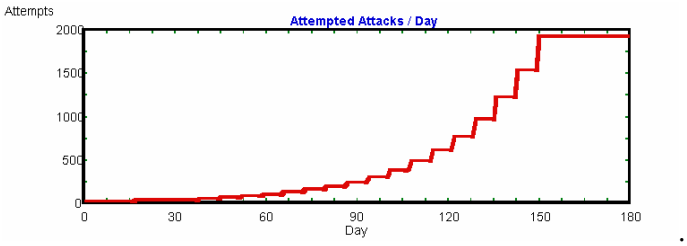

Fig. 17. Attempted Attacks per Day, *Escalation* with *Limits to Growth*

The number of attempted attacks escalates as often as every seven days (the organization's wait time between advertisements) until approximately Day 155. At that point, the company halts its advertising, and a constant 1925 attacks per day are attempted for the remainder of the period. Yet, returning to Figure 16, *successful* attacks are found to rise several times between Days 155 and 180. As the organization continues to reallocate staff to security and increase its demands on them, the personnel shortage for other tasks leads to more technical problems, sidetracking the increasingly overwhelmed security staff; attempted attacks thus become successful as the state of the countermeasures deteriorates and system vulnerability rises. *Limits to Growth* leads here to a decline in gains, not to a plateau.

By examining the behavior of the system, one can realize the problem of the increase in "successful attacks" around day 145. In response to this problem, the organization should take some action. Below we show how the use of simulation can support decisions regarding what action best fits the goals and context of a given organization.

Firstly, as our system was described, the increase in attempted attacks came not directly as a result of increased security efforts, but as a result of the organization's

advertising. While this may not be the case for all organizations, certainly any organization considering advertising must weigh potential benefits (such as increased clientele) against the possibility of (and its preparedness for) *Escalation*.

Secondly, even when *Escalation* is called for, it may be wise to escalate less strongly. The organization's rule for increasing security efforts was given as: "Every x_1 days, if successful (simple, sophisticated) attacks are greater than (θ_1, θ_2), increase efforts by (w, v)." Increasing the period x_1 (i.e. reducing the frequency of possible escalation), raising the thresholds θ_1 and θ_2 (reducing the frequency of when escalation is called for), and/or reducing w and v (the quantities of escalation when it is employed) are all possible solutions. When a threat is perceived, the effect of reaction must be weighed against the risk of no action, and sometimes the greatest overall gain is achieved by a slower or weaker reaction. Similarly, when we turn to *Limits to Growth*, it is noted that if the limits will not be (or cannot be, as in [Mar03]) removed, then reducing the growth action will delay the onset of the limiting factors, as well as slowing the deterioration of growth once the limits manifest themselves. A reduction solution thus heeds both archetypes. To see which of these three reductions is most effective here, all three were simulated: reducing the frequency of increased efforts, raising the threshold for increased efforts, and reducing the quantity of efforts. Experimenting with each solution individually as well as combined with others, we found that our system responded most favorably to simply reducing the quantity of escalation *w* by 30%: each time the organization decides to increase its security efforts, it does so by 6.4 staff-hours, as opposed to the 9.1 of the baseline case. The results are shown in Figure 18.

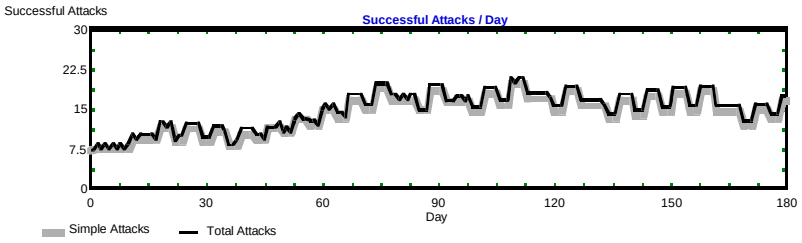

Fig. 18. Results of Reduced Escalation (Successful Attacks per Day)

Note that the equilibrium number of successful attacks has risen to about 15 (as opposed to the 11 in Figure 16), but there is no dramatic climb in successful attacks by Day 180. The weaker reaction has not pushed demands on the staff beyond their point of optimality in these six months. This approach can thus be thought of as "partly losing *Escalation*, but winning *Limits to Growth*." Note as well that the number of successful attacks reaches θ_3=18, the advertising threshold, several times, leading to less advertising and thus less attempted attacks. Integrating Figures 16 and 18, we find that the number of cumulative successful attacks is less for reduced escalation (~2650) than for full escalation (~2800). Given a particular organization's structure, goals, and priorities, the above tradeoffs (equilibrium number of attacks, rise in attacks, advertising opportunities, cumulative attacks) should be considered to find whether reduced escalation is more in its interest than full escalation.

Thirdly, a solution commonly found for *Limits to Growth* is to cease the growth action, and instead concentrate on removing the limiting condition. In our case, this would translate into hiring additional support staff. [Sen90] stresses the concept of "leverage", i.e. an organization's efforts will yield maximal gains if it carefully chooses where and when to apply those efforts. While hiring too early is prohibitively expensive, if hiring is delayed too much, the limit will set in and deterioration of

gains will begin. Additionally, the stronger the limit has become, the harder it is to remove it; in our case, once the support staff is overwhelmed with demands, it will not have time to introduce new hires to the intricacies of the computer system. Thus, the point of highest leverage for hiring is *when it will take effect just before the demands on personnel exceed their optimal load.* This requires great prediction skills on the part of the manager, including a sense of "feedback" regarding the support staff's load. Otherwise, the best strategy is to hire *as soon as possible once a decline in gains is visible.* This also requires the manager to realize that indeed, gains have diminished since the optimal personnel load was reached. As opposed to the previous strategies, which are executed before-the-fact, this strategy describes how an organization might now respond to problems. Following full escalation, Figure 16 showed a rise in attacks around Day 145. Figure 19 shows the results if the organization responds rapidly and additional personnel are available as of Day 155.

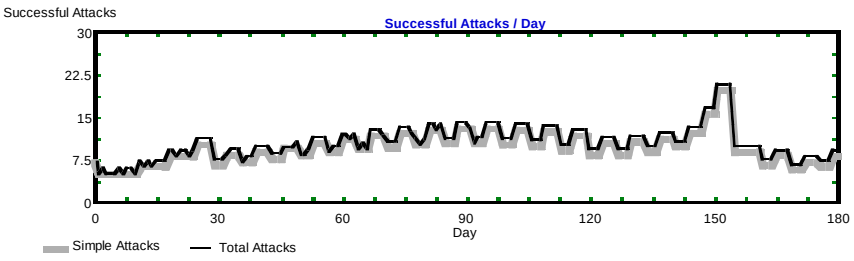

Fig. 19. Full Escalation, with Hiring at Day 155

The oscillations and steep rise occur as in Figure 16, followed by a steep drop in attacks due to the hiring. Integrating, we find a total of approximately 1,880 successful attacks, far less than in full or reduced escalation. (Of course, this benefit

comes with the cost of hiring.) The sooner the hiring, the less of the peak around Day 150; the longer the wait to hire, the greater the peak.

Figures 16, 18, and 19 have shown the results of the three above scenarios in terms of the number of successful attacks. An organization must also consider factors such as labor costs, and thus Figure 20 displays the cumulative staff-hours employed for each scenario: "baseline" (full escalation, without hiring), "reduced escalation", and "escalation with hiring." Note that the efforts of "baseline" and "hiring" will coincide until Day 155, at which point the curve for "hiring" will grow more steeply.

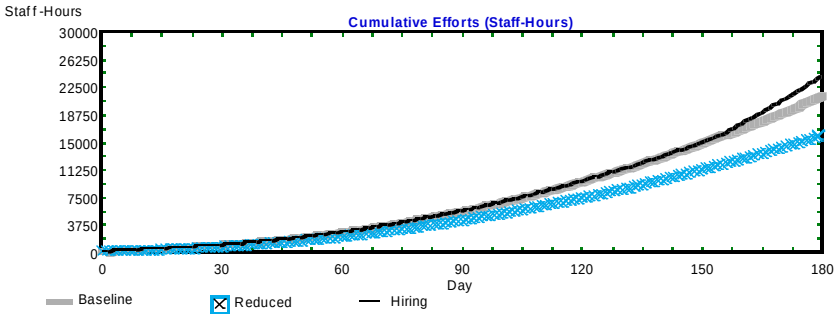

Fig. 20. Cumulative Staff-Hours for Each Scenario

Thus, given the number of successful attacks and number of staff-hours employed, both per-day and cumulatively, an organization can consider its best options as it encounters this combination of archetypes.

For additional information on the combination of *Escalation* and *Limits to Growth* as it occurs in security, please see [Ros06c], from which this chapter was excerpted.

Chapter 5: Related Work

5.1 System Dynamics and Archetypes

System dynamics thinking is introduced in [For61]. An introduction to archetypes can be found in [Sen90], with added details and recommendations in [Sen94], while [Bra02] extends this work to list ten different archetypes. [Wol03] argues that all existing archetypes can be included in one of four "core generic" archetype categories such as "Underachievement" or "Out-of-Control"; however, [Wol03] acknowledges that the more-specific, more-familiar ten archetypes (such as "*Escalation*" or "*Limits to Growth*") are more rapidly applied to real systems, and we have thus used them here. [Mar03] applies systems thinking and archetypes to safety engineering. Some archetypes found in safety are clearly those seen elsewhere (such as "Eroding Goals"), but others seem unique to safety. (This is the case partly because safety measures can be a victim of their own success – when no accidents occur, there can be pressure to reduce safety measures.) For now we have focused on the more-common archetypes of [Bra02] regarding security, but future work may find that new archetypes apply to safety as well.

5.2 Sources of Data

Empirical data regarding computer security are still fairly rare as of now. Anecdotes detailing attacks and their responses, such as [Gib02], are very illustrative of the attacker/defender interaction, but few such anecdotes have been published.

Some information regarding what is general practice in the security world today can be found in [Gor05a], a survey of several hundred organizations. For example, our model includes IDSs but not biometrics because the former is found to be significantly more prevalent in real life today.

Most data on attacks are gathered from analyzing “honeypots” or “honeynets”, systems designed to be attacked. Such studies include our own laboratory’s [Pan05], as well as [Dac04], [Pou04a], [Pou04b], and [Pou05].

Hypothesized attacker behavior is described in [Jon97], based on empirical findings from controlled attack experiments. This focuses on the behavior of the individual attacker, while more data are needed on the aggregated effects of multiple attackers.

To help meet the dearth of empirical data regarding security, nine teams are collaborating on the projects DETER and EMIST [Baj04]. DETER involves building a massive (currently approximately 200 machines, intended to reach 1000 machines) “researcher- and vendor-neutral” network testbed for emulating various types of attacks, countermeasures, and network topologies. Meanwhile, the EMIST project seeks to formalize methodologies for measuring these effects. Combined, these projects should provide a wealth of useful, unbiased, and well-accepted emulated attack data. Both studies will enrich our model with quantifiable values, e.g. honeynet findings might show that 20 buffer-overflow attacks of a certain type are attempted each day, and the DETER/EMIST findings would tell us that the attack will succeed 80% of the time if the network has Topology A but only 60% of the time with Topology B.

Regarding user factors, [Lar03a] uses surveys to understand Internet usage, and [Lar03b] conducts studies with test websites to investigate users’ privacy behaviors online. The authors of these papers have indicated that their future work will analyze

user behavior regarding network security, which should be applicable to our user model.

Comment [SNR1]: This better?

5.3 Economics and Security

[Cam03] considers the effects of public disclosure regarding security breaches on a company's stock prices. [Gor02], [Gor05b], and [Bod05] all use economic analysis in determining how much security investment is worthwhile for a company, given its priorities; however, details are not provided as to what should be done specifically with the investments. This provides the connection point to our model.

Economic requirements are also used to lead to assumptions or specifications for related computer security, e.g. determining the subjective cost and total welfare regarding network routing [Fei05] or requirements on trusted platforms placed by digital rights management [Ber04], [Ber05].

5.4 Other Modeling Approaches in Security

One approach in security has been to probabilistically quantify an attacker's behavior and its impact on a system's ability to provide certain security-related properties. Attempts have been made to build models that take into account both the attacker and the system being validated. A general model of an intrusion-tolerant system is proposed in [Gon01] to describe security exploits by considering attack impacts; the system state is represented in terms of failure-causing events. [Jha01] proposes a combination of state-level modeling, formal logic, and Bayesian analysis

quantify system survivability. Finally, Ortalo et al. [Ort99] propose modeling known vulnerabilities in a system using a "privilege graph". By combining a privilege graph with simple assumptions concerning an attacker's behavior, the authors then obtain an "attack state graph." Parameter values for such a graph have been obtained experimentally; once obtained, an attack state graph can be analyzed using standard Markov techniques to obtain several probabilistic measures of security. [Ste04] uses a probabilistic model for validating an intrusion-tolerant system that combined intrusion tolerance and security, allowing the designers to make choices that maximize the intrusion tolerance before they implement the system. Compared to these models, the model presented here is more generic in its inclusion of other human elements such as users and system administrators. Additonally, other than [Ort99] which uses data collected empirically to assess some of the parameters values in the model, the other ones are not developed to easily be linked to empirical data.

Cyberciege ([Nav06], [Irv05]), developed by the Naval Postgraduate School, is a computer game with a very engaging user interface and virtual world, intended for training students to understand security engineering. Cyberciege focuses on detailed access control, user-by-user, for a small number of users. Each piece of hardware is hand-selected from a list of fictional brands (e.g. "BitFlipper router"), and physical security measures are implemented on a user-by-user basis. The determination of whether an attack succeeds is by comparing asset desirability and how well standard procedures have been followed. Cyberciege's level of detail models the role of an individual security officer who might oversee a dozen computers at most, while our

model abstracts one level higher, to the manager who oversees several hundred machines.

In a similar vein, Fred Cohen & Associates offer a security simulator [Coh06] on their website (http://all.net/games/index.html). Fully described in [Coh99], this simulator gives examples of how a single attack of varying sophistication might succeed against different computers with different countermeasures. The defender strength, i.e. to what degree the defender does the right thing, is specified as a percentage by the user before running the simulator. If an attack succeeds, the dollar loss due to the attack is estimated based on the attacker profile, e.g. how much will a successful attack by a private investigator cost? Our approach attempts to add in more empirical data, as described in Section 5.2. Additionally, our work extends the "defender strength" idea by allowing for strengths of each countermeasure: a system may have a 90% effective firewall but only a 70% effective IDS. Furthermore, rather than specify a value for defender strength, the user of our model inputs managerial decisions such as how much effort is allocated to which security tasks and how skilled the staff is – the model then uses these inputs to determine the resulting defense strength for each countermeasure.

Chapter 6: Conclusions and Future Work

6.1 Conclusions

The archetype and results of simulation execution presented here show the value of systems dynamics modeling for enterprise security. The evolution over time of two slightly different "what-if" scenarios may result in very different pictures, reinforcing the value of simulation. Systems thinking, combined with simulation, can assist an organization in placing its efforts in the places that will give the most "leverage" to their goals, and in diagnosing and solving problems. This approach thus leads to a more enlightened weighing of costs vs. benefits for the proposed decisions that an organization might make.

System dynamics simulation is also an intuitive and powerful tool for understanding computer security, as well as for training professionals. In time, our model will mesh with much other research currently being done by others, leading to gains in a wide variety of directions.

6.2 Future Work

A great deal of future work remains as well, including:

- "Deepening" the simulation model with more detail, e.g. where linear rates had been assumed, perhaps logarithmic or exponential would be more accurate. The documentation of the simulation model already reveals several ways it can be deepened.

- "Broadening" the model to include such factors as:
    - User details describing their interaction with the security policy.
    - Asset properties. Currently we only show successful attacks; future work can link this to system availability, confidentiality, and integrity.
    - Internal attacks. Currently it is assumed that the firewall is X% effective against all simple attacks, for example, which assumes that all simple attacks come from outside the firewall.
- Obtaining additional empirical data for use as parameters in the simulation model. Sources for such data, including work from our own research group, are described in Section 5.2.
- Modeling other instances of the above archetypes, modeling other archetypes, other combinations of archetypes, and looking for new archetypes. Appendix I gives a few ideas for modeling other archetypes.
- Documenting real-world case studies in security, using archetypes to explain the situations, and using simulation to suggest improvements. (For example, [Sen94] first describes the story of an airline's failure, applies archetypes to describe it, and then builds a simulator through which it is shown, for example, that had the airline not cut its ticket prices quite so steeply, it would not have gone bankrupt.) We have already begun interviewing one system administrator and documenting his case study, but obtaining the necessary details, applying archetypes, and simulating the case study are all left for future work.

Appendix I: Archetypes

Here we briefly describe each of the ten archetypes of [Bra02], giving one possible example from security.

Shifting the Burden, or **Symptomatic Fixes**. We witness a problem symptom, and rather than think about the root cause, we try to fix the symptom. Doing so distracts us from the actual cause of the problem, or masks the symptoms so it's harder to diagnose the problem. Suppose a system is continually falling victim to successful "script-kiddy" attacks (symptom). The company may install an I.D.S. to catch the attacks (symptomatic fix), when in reality the attacks wouldn't make it into the system if the company had a good firewall, and wouldn't succeed if they kept their vulnerabilities down. (Fundamental fixes.)

Fixes that Fail. Here, the attempted fix actually worsens the underlying problem in time. The newly-installed I.D.S may have a high false-alarm rate and require a great deal of the sysadmin's attention. The sysadmin is now too busy to attend to other duties (such as addressing vulnerabilities), so the number of successful attacks actually increases.

Success to the Successful. There is a tendency to believe that if putting some money into Approach A yields good results, then putting more money into Approach A (and ignoring Approach B) will further improve results. For example: for an investment of $100, a Host-Vulnerability-Scanner will yield more improvements than an IDS. But continued investment into the Host-Vuln-Scanner (diverting funds from the IDS) will not help much if at all.

Limits to Growth. Increased efforts and investments produce increased results, until the system reaches its natural limit. At that point, results will either plateau or decline. For example, given an inexperienced sysadmin staff of a fixed size, training them will result in significant gains to the network's security. But eventually, their size (rather than skill) becomes the limiting factor, so further training will accomplish nothing.

Attractiveness Principle. Increased efforts are no longer producing results, with two different limits fighting growth. The manager must decide which limit to address first/more. Suppose we have a simultaneous investment in both more/better sysadmin staff, and some technology (maybe a firewall). At some point, the Return on Investment will drop; at that point, we must decide which factor is more of a limiting one.

Growth and Underinvestment. A successful approach may initially seem to fail if it wasn't given proper investment/support/capacity. For example, a company may double its system size; if the SysAdmin size (which is the capacity in our case) is kept constant, overall performance will drop. If, instead, the SysAdmin size is properly increased, the company will see a gain.

Eroding Goals. If a goal is not immediately met, it can be tempting to reduce the initial goal. A manager may try for an Availability (or confidentiality, etc.) Level of 3, find that the expenses next month are too high, so s/he drops the goal to Level 2. The next month, the company is hit with a massive attack, causing more loss than had it held the course at Level 3. (Another example would be, "We want an IDS that

catches 100% of all attacks. What, that gives too many false alarms? Okay, maybe 90%. Still too many alarms? Okay, maybe set it to 80%.")

Escalation. Party A puts in more efforts, yielding more results; this threatens Party B, who does likewise, and so on. (The U.S./ U.S.S.R. arms race during the Cold War is a good example.) If a company increases its security efforts and publicizes how secure it is, or otherwise makes itself more of an attractive target, it will receive more sophisticated attacks, which will require more security investments, and so on.

Accidental Adversaries. Two parties initially agree towards cooperation, but then Party A perceives an offense (often unintentional) from Party B; it then retaliates, and the situation escalates from there. An example here would be the SysAdmin and User, who agree they want the company to succeed, but then the user accidentally breaches the security policy, leading the SysAdmin to impose a harsher security policy and other enforcement measures. The user (or another user) may become annoyed and retaliate.

Tragedy of the Commons. If two efforts independently consume a common resource without respecting one another, both will see reduced gains as the resource runs out. In our case, if a company decides to invest more in IDS as well as Host-Checking-Tool, but maintains the size of its SysAdmin (which is the "common" resource consumed), both will not yield full results.

Appendix II: Model Screenshots

The basic building blocks for continuous modeling in the Extend simulation environment include holding tanks, constants, and equations, to name a few.

As one example, we show a simplified version of how the antivirus software effectiveness is modeled. Suppose that this system needs its virus definitions updated on a daily basis; if so, an antivirus that has been totally neglected for too long of a period will become close to useless, as it fails to catch the majority of viruses circulating the Internet today. Thus, antivirus effectiveness is reduced each day by some average "daily loss rate" which describes the occurrence of new viruses, and increased each day by the number of staff-hours updating its definitions (or otherwise maintaining it) that day. The effectiveness is then measured on a 0-to-1 scale and output. This is modeled in Figure 21.

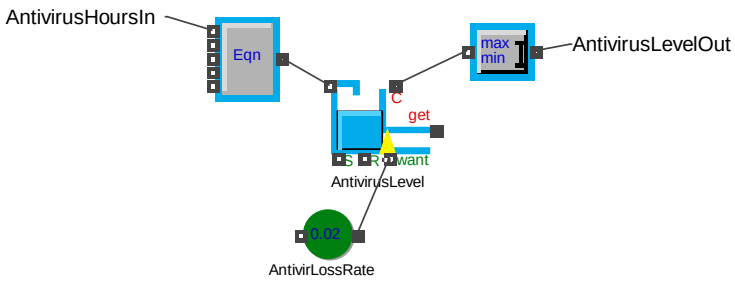

Fig. 21. Sample Screenshot of Holding Tank, Equation, and Constant Blocks

Notice the number of staff-hours in, subject to some function, the holding tank for the daily antivirus effectiveness, and the daily loss rate. Each day, the contents of the

tank are given, limited to the range between 0 and 1, and output as today's "antivirus level."

To allow for greater abstraction, all of the above blocks can be inserted into a custom-built "hierarchical blocks", such as the one shown in Figure 22.

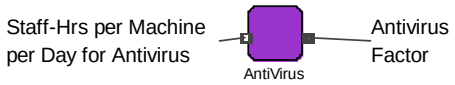

Fig. 22. Sample Hierarchical Block, Antivirus

Here we see only the input and the output; the remaining holding tank, equation block, etc. are all hidden inside the hierarchical block.

In our model, a certain number of attacks of a given sophistication level are attempted each day. Depending on the effectiveness of the various countermeasures and the system's vulnerabilities, a certain number succeed. Another hierarchical block, which performs this evaluation, is shown in Figure 23.

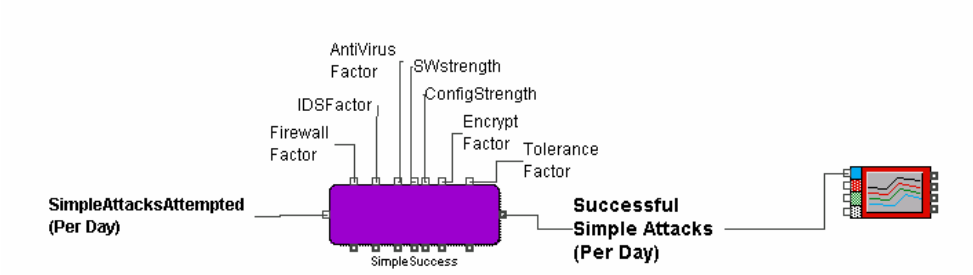

Fig. 23. Sample Hierarchical Block, Simple Attack Success

The block on the far-right of Figure 23 is an output plotter, used to generate many of the figures presented in this thesis.

The model has a great deal of constant parameters; for example, the antivirus daily loss rate of Figure 21. These are listed in a spreadsheet such as the one displayed in Figure 24.

ConfigVulnLevel, loss if ignored	.05, linear
", staff-hrs needed to maintain	1.5 / machine
NetVulns, loss if ignored	.04, linear
", staff-hrs needed to maintain	0.6 / machine
AppVulns, loss if ignored	.004, linear
", staff-hrs needed to maintain	0.13 / machine
AppVuln, loss from new S/W	0.8
ToleranceLevel, loss if ignored	.1, linear
", staff-hrs needed to maintain	0.67 / machine
EncryptionLevel, loss if ignored	.001, linear
", staff-hrs needed to maintain	0.067 / machine
Antiviruslevel, loss if ignored	.02, linear
", staff-hrs needed to maintain	0.4 / machine
FirewallLevel, loss if ignored	.033, linear
", staff-hrs needed to maintain	0.66 / machine
IDSLevel, loss if ignored	.05, linear
", staff-hrs needed to maintain	2 / machine

Fig. 24. Sample from Spreadsheet with Parameter Values

Lastly, while certain parameter values (such as antivirus loss rate) reflect the reality of the system, others (such as machine size, staff size and the presence of countermeasures) reflect decisions that a manager might make. To allow for easy "what-if" simulation, these parameters were extracted to a user-friendly Graphical User Interface, such as the one seen in Figure 25.

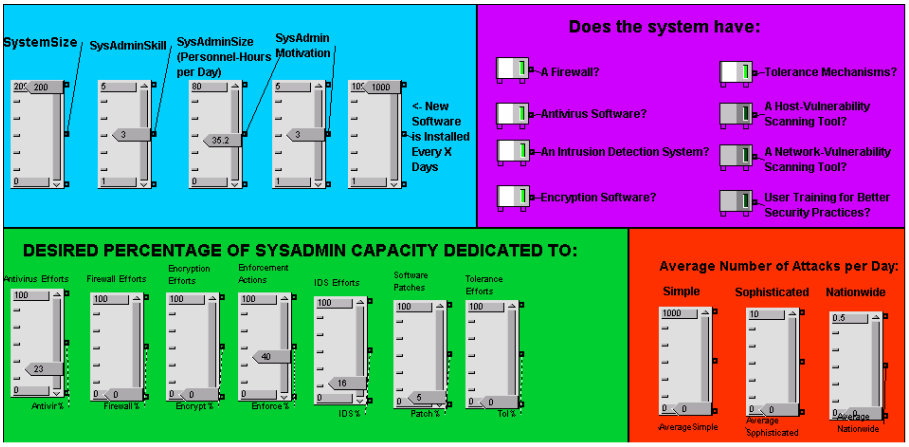

Fig. 25. Graphical User Interface Screenshot

Appendix III: Model Documentation

OVERVIEW: The end-user of the model sets several sliders and switches to describe the *system, countermeasures, allocation of sysadmin to various tasks,* and *attacks.* The end-user can then see the *costs* of this configuration. A certain number of attacks are then attempted on the system each day; given the details of the system and its countermeasures, the end-user can see how many of those attacks succeed, or how many were blocked by a given countermeasure. The end-user can also track the effectiveness of a given countermeasure over time.

SYSTEM INPUTS: These sliders describe the system and staff, and are listed in Table II.

Table II: Slider Inputs for the Model Graphical User Interface

Name in Model	**Type**	**Meaning**
SystemSize	Slider, 0-80	(# of machines)
SysAdminSize (Personnel-Hours per Day)	Slider, 0-80	Personnel-hours (or "man-hours") of System Administration and Security Officer staff employed per day. A SysAdminSize of 40 describes 5 people working 8 hrs/day each day, or 10 people working 4 hrs/day, etc.
SysAdminSkill	Slider, 1-5	Average overall skill of the System Administration and Security Officers Staff. The 1-5 scale is ours.
SysAdmin Motivation	Slider, 1-5	How motivated the SysAdmin staff is to protect the system; we impose a 1-5 scale.
New Software is Installed Every X Days	Slider, 1-1000	Interval (in days) between installation of new software (which contains new vulnerabilities). Patches are not included here.

Further descriptions of the system, e.g. Windows vs. Linux, would be a critical step in adding detail to the model; it will hopefully be considered in a future implementation.

COUNTERMEASURE INPUTS: We include several common countermeasures. In the 2004 CSI/FBI Computer Crime and Security Survey of 494 U.S. corporations, universities, government bodies, etc., the most common security technologies used (Fig. 16), by percentage of respondents, were: Antivirus software (99%); Firewalls (98%); Server-based access control lists (71%); Intrusion detection (68%); Encryption for data in transit (64%). We view the access control lists as part of the "SysAdmin's Enforcement Actions" and not a separate technology *per se*, as it is built into most operating systems today. For simplicity's sake, we chose to include both data-in-transit encryption and file-encryption as "encryption software."

A significant countermeasure not described directly in the CSI/FBI survey is the emerging field of attack *tolerance* (as opposed to prevention or detection). This could include designs for graceful degradation under attack; redundancy and diversity (in some cases); and other technologies allowing the system to succeed despite the attack. We thus include a countermeasure entitled "tolerance mechanisms."

Additionally, as 70% of the survey respondents (Fig. 17 in the FBI survey) identified some type of network security training for their users as important, we have included "user training for better security practices."

Lastly, we have included vulnerability-scanning tools which can assist the system administrator in finding vulnerabilities to fix. These include host-configuration vulnerability scanning tools, such as FERRET; and network-vulnerability scanning tools, such as NESSUS.

For all of the above countermeasures, we presently assume that they are either present in full strength, or not at all. (They're controlled by binary switches.) Future implementations of the model may modify this. The countermeasures are given in Table III.

COST/EXPENSE EQUATIONS & OUTPUTS: Given the above descriptions, we can now compute the system's expenses. (For now, we simply tally the number of successful attacks, rather than describing the monetary loss they cause the company; this too will hopefully be improved in a future model.)

Table III: Countermeasures Included in the Model

Name in Model	Type	Meaning
A Firewall?	Y/N Switch	"1" if the system has a firewall; "0" if it doesn't.
Antivirus?	Y/N Switch	"1" if every system has antivirus software installed.
An Intrusion Detection System?	Y/N Switch	"1" if an Intrusion Detection System is present.
Encryption Software?	Y/N Switch	"1" if encryption software is installed.
Tolerance Mechanisms?	Y/N Switch	"1" if tolerance mechanisms are present.
A Host-Vulnerability Scanning Tool?	Y/N Switch	"1" if the sysadmin uses a tool such as FERRET to check host-configuration vulnerabilities.
A Network-Vulnerability Scanning Tool?	Y/N Switch	"1" if the sysadmin uses a tool such as NESSUS to check for network vulnerabilities.
User Training for Better Security Practices?	Y/N Switch	"1" if the users are trained regarding network security.

Expenses reflect all the money spent on the system over the duration of the simulation (usually ~100 days). *StaffCost* is the cost per day of employing the sysadmin staff. *PurchaseCost* is the cost to purchase the various countermeasures, which we assume is a one-time payment. We then have:

Expenses = *(StaffCost* * Time) + *PurchaseCost.*

($) = *($/day) * (days)* + *($)*

In Extend terms, *Expenses* is an accumulating tank; *StaffCost* is the input, and *PurchaseCost* is the initial level.

StaffCost = STAFFCOSTPERHOUR * *SysAdminSize.*

($) = *($/hr) * (personnel-hours)*

The cost of employing the sysadmin staff per day. We assume an average cost of $35 per personnel-hour.

For *PurchaseCost,* we assume that *Tolerance Measures, Encryption Software,* and *an Antivirus* must be purchased for every machine in the system to be effective. (The effects of installing an antivirus on only half, 1/3, etc. of the machines would be another interesting question for future work.)

Per-system purchase costs = *SystemSize* *

{ (Tolerance Measures?)*TOLCOST + (Encryption Software?)*ENCRYPTCOST

+ (Antivirus?)* ANTIVIRUSCOST }.

($) = (# machines) * Σ{(1/0)*($/machine)}

We simply assume for now that tolerance measures cost $300/system. For encryption software, PGP is very commonly used (try Google searches for "encryption software" and the like); the most basic version of PGP Desktop Professional 9 costs $200; we have used the value $220 to allow for a few more features. For the antivirus, Norton Antivirus, one of the most popular products on the market, costs $40 /machine in the 5-user pack. *(Sources: manufacturer's websites.)*

We do not include the host-configuration or network-vulnerability scanning tools in costs or expenses, as the most popular products used (i.e. FERRET and NESSUS) are available for free. The remaining two *PurchaseCost* items are the firewall and IDS, whose cost is independent of the size of the system behind them.

PurchaseCost = per-system purchase costs + (*A Firewall?*)* FIREWALLCOST + (*An IDS?*)* (IDSCost).

($) = *($)* +*(1/0)*$* + *(1/0)* $.*

We assume that a high-quality firewall costs $10,000, given Dr. Cukier's experience with proprietary firewalls. For the IDS cost, we take the price of the Cisco 4250, which is $30,000.

SYSADMIN ALLOCATION: We describe the SysAdmin staff's "capacity" to maintain and protect the system as a function of its size, skill, and motivation:

TotalSysAdminCapacity = *SysAdminSize* * *SysAdminMotivation* * ln(*SysAdminSkill* + 1).

(Note that TotalSysAdminCapacity is measured in pseudo-personnel hours, as it can be increased by motivation and skill.) (The logarithm is used to reflect the

phenomenon that beyond a certain point, additional training accomplishes very little. We use (skill+1) so that a skill level of 1, the lowest, doesn't result in an ln(1) = 0 term.)

The end-user then decides what percentage of the *TotalSysAdminCapacity* should be dedicated to what task, using the sliders in the green box. The sysadmin needs to spend time and attention to deal with any given countermeasure (or its side effects!). We refer to these as "countermeasure efforts." Obviously, more efforts are needed during deployment than afterwards, but for now, we simply describe "efforts-per-day." (One approach would be to consider an average effort over the product's lifetime, including its deployment, but this again is for future work.) The order of the various efforts is consistent with that of the model, but it has no particular significance.

"Antivirus Efforts" consist primarily of keeping all of the antivirus definitions up-to-date. The percentage of *TotalSyadminCapacity* dedicated to Antivirus Efforts is called *Antivir%*.

"Firewall Efforts" consist of tasks needed to maintain the firewall, primarily through applying new patches as firewall vulnerabilities are discovered. *(Firewall%).*

"IDS Efforts" consist of maintaining the intrusion detection system, mostly by downloading new signatures. *(IDS%).*

"Encryption Efforts" consist of updating and maintaining the encryption software (quite possibly including helping users who run into difficulty using it). *(Encrypt%).*

"Enforcement Actions" include setting proper access control; monitoring the system for noticeable oddities; and developing and enforcing a security policy for the

users. For example, if a user tried using a "weak" (i.e. easily guessed) password such as "joe", a vigilant sysadmin would prevent him from doing so. (*Enforce%).*

"Software Patches" reflects the time spent per day on finding and installing patches for newly-discovered vulnerabilities in any of the system's netware, operating systems, or applications. *(Patch%).*

"Tolerance Efforts" depend on the particular tolerance measure; some measures are relatively low-maintenance (e.g. if graceful degradation has been built-in, then no further action is needed), but some are high-maintenance (e.g. if the system has a backup web server that runs a different operating system, the backup server has to be maintained as well). *(Tol%).*

"Addressing Alarms" refers to the alarms raised by the firewall and IDS; sometimes these were in fact attacks, but often they were legitimate actions. A good sysadmin should sort through these. *(Alarm%).* In the new versions, we've gotten rid of "addressing alarms" as its own task; it's now included in either "IDS Efforts" or "Firewall Efforts."

Comment [i2]: Page: 1
I suggest dropping the word "false" when we talk about alarms

The various desired percentages, as well as the *TotalSysAdminCapacity* and *SystemSize*, are input into the *HoursForTasks* block. The outputs of this block describe how many SysAdmin pseudo-personnel-hours (or more precisely, skill-motivation-personnel-hours each day) are actually allocated to each task.

If the various desired percentages (inputs) add up to 100 or less, then all of the desired demands can be met, and the process is simple:

Hours allocated to Firewall = (*Firewall%* / 100) * *TotalSysAdminCapacity.*

Hours allocated to IDS = (*IDS%* / 100) * *TotalSysAdminCapacity.*

The firewall and IDS are independent of the size of system behind them. (Or are they? Once we include analyzing alarms in Firewall Hours & IDS Hours, well, the bigger the system, the more alarms likely. In the new paper, I assumed staff-hours per machine for these as well.) For the other efforts, however, we must factor in the system size; after all, to spend a total of two hours per day on updating antivirus definitions for a single computer is certainly sufficient; for a thousand computers, it probably won't be. We thus talk of "hours allocated per system [per day]." Note that for now, we assume that doubling the system size will simply halve the personnel-hours available for a given task; in reality, larger system sizes tend to come with mechanisms for better management, so we might in the future consider a factor such as log(systemsize). For the moment, though, we've kept the divisor linear. The following hours are per-machine:

Hours allocated to Antivirus = (*Antivir%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Encryption = (*Encrypt%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Enforcement Actions = (*Enforce%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Software Patches = (*Patch%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to Tolerance = (*Tol%* / 100) * *TotalSysAdminCapacity / SystemSize.*

Hours allocated to (False) Alarms = (*Alarm%* / 100) * *TotalSysAdminCapacity / SystemSize.*

(A larger system will generate more alarms, and thus needs more attention.)

IF, however, the end-user specifies a series of percentages with the sliders that sum to >100%, not all of the desired hours will actually be allocated that way. A prompt can inform the end-user that the values have exceed 100%, and that s/he may wish to modify values before running the simulation. (This prompt was built into several earlier models; it was omitted from the Aug. 17 version for simpler presentation, but can be reincorporated if desired.) If the end-user chooses to continue, priority will be assigned from left-to-right, i.e. first *Antivir%* of the *TotalSysAdminCapacity*, if available, will be allocated to antivirus; then up to *Firewall%*, if available, to firewall, and so on.

(Extend description: looking inside the HoursForTasks block, we see a series of equations, converting the percentages into *HoursDemanded*. Below that, we see a series of holding tanks. All of the tanks are reset to their starting values at the end of each day by a periodic pulse. The first tank has starting value *TotalSysAdminCapacity*; the other tanks have starting value zero. At the beginning of each day, *TotalSysAdminCapacity* flows into the first tank; *AntivirusHours Demanded* is "wanted" from that tank; the quantity "gotten" is the hours actually allocated to antivirus. The remaining contents of the first tank flow into the second tank, where again a demanded quantity is "wanted"; up to that quantity is "gotten", and the remainder flows into the third tank; and so on. Hours "gotten" are either used directly (firewall & IDS) or divided by the systemsize.)

For the version in the archetypes paper: a new allocation block was built, in which all demands are met if they sum to less than 100%; if they exceed 100%, then they are doled out in proportion to their demand, i.e. if the demands are: {30, antivirus; 40, tolerance; 50, IDS}; but the SysAdmin capacity is only 60; then it will allocate 15/20/25, respectively.

Additionally, the archetypes version adds in the factor of sysadmin inefficiency if pushed beyond optimal capacity. Hours allocated to a given task are decreased by a linear multiple of the total demand's exceeding the optimal capacity. (Note that we haven't yet included a factor to describe the inefficiency of X net demanded hours for ten different tasks, which has greater inefficiency than X net demanded hours for one task.)

I actually didn't use the allocation system in the new (*Escalation*) paper, I just "fed" each task directly as many hours as were desired.

The newest paper (DSN) once again made use of the allocation system – pushing the Sysadmin too far resulted in the "limit to growth." In this paper, the limit set in much more quickly, as we rephrased things: "sysadmin hours" were for all tasks; as soon as security demands take too many of those hours, other things go wrong because of ignored tasks. The rule used was thus the following (the numbers used here: 23%, 1.25, etc., were a combination of guesses on my part, and what made the graphs come out okay, i.e. GIVEN our guesses for how to describe a 200-machine system going from nominal attention to full attention to security over 180 days, we

wanted the limit to kick in towards the end of this period, and that the limiting effect be fairly strong.)

Maximum optimal security load = OPTPERCENT * TotalSysadminCapacity.

IF (total hours demanded for security > maximum optimal load), EfficiencyStretch = STRETCHCONSTANT * (total hours demanded for security – maximum optimal load).

AvailableCapacity = TotalSysadminCapacity – EfficiencyStretch.

VULNERABILITIES: Many attempted attacks will only succeed if the system has (known) vulnerabilities. These are grouped into four categories:

"Mistakes" includes all user mistakes, such as not logging off, downloading a virus, and using weak passwords. *(Mistakes* block outputs *MistakeFactor.)*

"Host-Configuration Vulnerabilities" include settings that the sysadmin didn't set properly, such as leaving ports open, allowing everyone access to sensitive files, etc. (*ConfigVulns* block outputs *HostConfigVulnsFactor.)*

"Network Vulnerabilities" include those flaws that have been discovered in the network software, which could be exploited by an attack; these can be corrected with patches. *(NetVulns* block outputs *NetVulnsFactor.)*

" Application Vulnerabilities" include those flaws discovered in application software, which could be exploited by an attack (e.g. a flaw in Apache could be exploited for a denial-of-service attack; a flaw in Outlook Express might be exploited

to cause remote execution of code). These are also corrected with patches. *(AppVulns* block outputs *AppVulnsFactor.)*

We measure each of these subclasses as a "factor" between 0 and 1, where 0 is best (no known vulnerabilities of this type exist on this system) and 1 is worst (i.e. the system is permeated with vulnerabilities of this type). An overall "vulnerability factor" (*VulnFactor),* also between 0 and 1, is computed from these:

*VulnFactor = Min{1, [(*MISTAKECOEFF * *MistakeFactor)*

+ *(*HOSTCONFIGCOEFF * *HostConfigVulnsFactor)*

+ *(*NETVULNSCOEFF * *NetVulnsFactor)* +

+ *(*APPVULNCOEFF * *AppVulnFactor)] }*

The "Min" function keeps the overall *VulnFactor* to a maximum of 1. Note that it is possible to reduce one or two vulnerability subfactors, and yet still have an overall factor of 1 if the other subfactors have been ignored. We believe that this reflects the reality of system vulnerabilities. We have weighted host-configuration vulnerabilities most heavily, followed by mistakes and application vulnerabilities, and then finally network vulnerabilities. This was Rosenfeld's impression of the most-frequently exploited vulnerabilities. (The host-configuration vulnerability is particularly pernicious, as an attacker often need not "breach" any part of the system to perform an attack; therefore, such an attack is often not detected by an I.D.S.) [How did you obtain such a ranking? Is it more based on the number of vulnerabilities of each type present or on the impact that each of these vulnerability types has?] I was thinking impact, e.g. a single config error could be more dangerous than a single app

vulnerability. Thus my comment about "more pernicious as it doesn't require a breach." Again, this is all my judgment here.

Archetypes Version: To show the difference between those vulnerabilities fixed by enforcement (i.e. host config and mistakes), and those fixed by patches (i.e. netvulns and appvulns), two more derived values were created: (A "strength" of 1 is best.)

ConfigStrength = 1 – (CONFIGHOSTCOEFF* *ConfigVuln*) – (CONFIGMISTAKECOEFF* *MistakeFactor)*;

SWStrength = 1 – (SWAPPCOEFF* *AppVulnFactor)* - *(*SWNETCOEFF * *NetVulnFactor);*

Comment [i3]: Page: 1
I don't understand this

SWStrength is a straight average of how well the apps and the netware/OS has been patched. (Again, with the "1 minus thing" to switch a 1->0 scale (vuln of 0 is best) to a 0-> 1 scale (strength of 1 is best).) For ConfigStrength, I weighted the average 60/40 between ConfigVulns and Mistakes; again, just my judgment as to how dangerous ConfigVulns are.

Newest version (*Escalation* paper): I left the "SWStrength/ ConfigStrength" same as the previous paper, as it worked perfectly well for my purposes here.

We now describe the workings of the individual vulnerability subfactors.

Mistakes. User mistakes are given as a factor of three conditions: The users' *Awareness* of security issues, as a value 0-to-1, where 0 is no awareness, 1 is very high awareness; the users' *Concern* for security issues, 0-to-1, (we assume at this

point that the lowest level of concern is 0, i.e. no concern; the issue of deliberate sabotage, where the user is "negatively concerned" with actively damaging the system, has not yet been incorporated into this model); and the sysadmin's *Hours Allocated to Enforcement Actions*, in pseudo-personnel-hours per machine per day. With proper sysadmin enforcement actions in place, the users' ability to make dangerous mistakes can be sharply minimized or eliminated altogether. We then compute an overall *Mistake Factor*, as a value between 0 (no dangerous user mistakes) and 1 (dangerous user mistakes happen all the time).

MistakeFactor = 1 – [{(AWARENESS + CONCERN*)* / USERMISTAKEDIVISOR}

+{ *Hours Allocated to Enforcement Actions* / ENFORCEMENTMISTAKEDIVISOR}]

Archetype version: the 0.7 was changed to 1.5. [Where are these numbers coming from?]

The *MistakeFactor* is then limited by a max of 1 and a min of 0. The numbers were designed as follows: even given perfect awareness and concern, i.e. *Awareness + Concern = 2*, if there are no enforcement actions, the mistake factor will still be .091 (not 0) to account for human error; (for example, this author recalls once downloading a virus simply because he accidentally clicked the wrong button.) Conversely, given sufficient enforcement actions (we assume .7 pseudo-hours per system per day would suffice), the mistake factor will go to zero, regardless of user awareness or concern. Note that the mistake factor is "memoryless" and employs no holding tanks; we assume that if awareness, concern, or enforcement were to suddenly decrease today, the effects would be felt immediately.

Comment [i4]: Page: 1
is this realistic?

While we would like to make *Awareness* and *Concern* variables that the end-user could adjust, for now we simply set both *Awareness* and *Concern* to 0.2. The *User Training for Better Security Practices?* switch adds TRAININGEFFECT to *Awareness*, raising it to 1. [Where are these numbers coming from?] (Future implementations may describe the effects of training over time, i.e. a gradual rise in awareness.) These numbers are all just guesses on my part about the "average" user's awareness and concern, and how much training can help. (I wanted to demonstrate the effectiveness of full training, so I let it raise Awareness all the way to 1.)

ConfigVulns has two inputs: *Hours Allocated to Enforcement Actions* (in pseudo-hours per system per day), and *A Host-Vulnerability Scanning Tool?* (0 if not present, 1 if present). A *HostConfigVulnsFactor* of 0 is best. The scanning tool assists the sysadmin by finding the vulnerabilities present; however, the sysadmin must still spend time fixing these vulnerabilities! We thus model the scanning tools as an increase in the "effective hours" (or "virtual enforcement hours") available for fixing vulnerabilities. If the scanning tool is not present, "virtual enforcement hours" = hours allocated to enforcement actions. If the scanning tool *is* present, "virtual enforcement hours" = CONFIGTOOLMULTIPLIER * hours allocated to enforcement actions.

Yes, it was my assumption for the present that the tool can double the sysadmin's effective hours here; just picked a number to try.

The vulnerability level is then described using (1 – the level of a holding tank), i.e. the holding tank is "full" when all vulnerabilities have been patched, and "empty"

Comment [i5]: Page: 1
For future versions we might want to elaborate on this, make it more similar to the attcks model, i.e., deal with number of vulnerabilities rather than with vulnerability level

when no vulnerabilities have been patched. (All of the holding tanks in this model have the default setting of "want" connector not being able to reduce the tank value below zero.) The holding tank has a "loss rate"; this describes the fact that over time, new vulnerabilities are discovered; additionally, new user accounts are created, etc., all requiring attention from the sysadmin to prevent additional vulnerabilities. We set the loss rate here to .05, i.e. if the sysadmin configured the system perfectly, but then ignored it for twenty days, it would now look very vulnerable. (For now we assume a linear loss rate. Further detail may modify this in the future.) (Yes, all assumptions of mine which could use validation.) The next question we ask is, how many pseudo-personnel hours are required to maintain the tank at its "full" level of 1? For configuration vulnerabilities, we assume it to be 1.5 pseudo-hours per system per day. (Or with the scanning tool, .75 pseudo-hours per day.) We then have a "divisor" of (1.5 / .05) = 30, i.e.: input to the tank = *Virtual Enforcement Hours* / 30.

The tank is designed that if the current level of the tank is already 1 (full), additional input (i.e. additional hours) will not raise the tank level further. Lastly, the initial level of the tank is also decided by the number of virtual enforcement hours. If 1.5 or more pseudo-hours are available, the initial level will be 1. Otherwise, the initial level will be (pseudo-hours allocated / 1.5). Expressed in terms of the "divisor" and "loss rate" constants, this is:

StartLevel = Min{1, [*Hours / (Divisor * LossRate)*] }.[I don't understand this discussion on the tank. What is the main message?]

What happens, as given, is that if the model starts out with enough staff-hours to keep the configvulns "happy", it will start at "full" and stay that way. Otherwise, it will inevitably decline to zero.

This raises the question: if you go through a year of only spending half the time you should on patches (or tolerance, etc.), by the end of the year, how vulnerable is your software? Totally? 50% Not sure.

In the old models, anything that required sysadmin hours was designed that if it didn't get enough of them, it would ultimately decline to zero no matter what in the long term. In the *Escalation* paper, this was changed for IDS, Firewall, NetVulns and ConfigVulns: for all of them, the effectiveness TODAY of a given countermeasure is given by a maximum of two values: the tank level (which reflects what it had been given in the past), AND the number of hours given TODAY divided by the number of hours required to be fully happy. Thus, if I patch well for a long time, then ignore patches for a few days, the patch level will be 0.9 or so. On the other hand, I could have totally ignored patches for years, but if I spend some time on them today, patches will be somewhat effective today. Please let me know what you think of this.

For the archetype version, to get better-looking results we often changed the start level to something specified, e.g. 0 or 1 or some constant in between that worked nicely. This describes a scenario of "new sysadmin walks in on a system that had been totally ignored for a long time", or "incompetent sysadmin ruins a system that had been fine."

The above scheme of holding tanks, loss rates, divisors, and starting levels, will be found repeated throughout other parts of this model.

NetVulns are designed very much like configuration vulnerabilities, except here the inputs are *Hours Allocated to Software Patches* and the presence of *A Network Vulnerability Scanning Tool?* Once again, the scanning tool increasesthe virtual hours available for patching network vulnerabilities as follows:

Effective Netvuln-fixing Staff-Hrs. = NETTOOLMULTIPLIER **Hours Allocated to Software Patches*. The *NetVulnsFactor* is similarly given as (1 – tank level). (The output from the tank is limited to the range 0 to 1. However, the feature "if tank level = 1, don't allow further input" was not added to the *NetVulns* block (or any other block in this model yet) due to time constraints; thus, as is, the theoretical tank level can exceed 1, but the most it will read out is 1.) LossRate = 0.04 (i.e. totally vulnerable if ignored for 25 days), divisor = 15 (i.e. fully patched if given .6 pseudo-hours per system per day), initial level = virtual hours / 0.6, limited between 0 and 1. Archetype Version: Divisor was changed to 32. [Where is all that coming from?]

Comment [i6]: Page: 1
what does this mean in terms of the real world?

AppVulns has inputs *Hours allocated to Software Patches* and *New Software is Installed Every X Days.* As before, a tank 0-1 describes the "strength" of the software; it is replenished by "hours allocated to patches", with a divisor of 33.33. The loss rate is .02. ADDITIONALLY, anytime new software is added, this causes an additional loss of 0.8. The addition of new software is modeled as an event that occurs every Y days, where Y is Gaussian, mean *New Software Installed Every X Days*, std. deviation 30%. Archetype Version: LossRate is 0.004. [Explain why you selected these values.]

COUNTERMEASURES: Countermeasures behave much like vulnerabilities, only a factor of 1 is best (countermeasure is fully effective), 0 is worst. Each countermeasure

outputs its factor, 0-to 1; and has inputs for the hours allocated to it, as well as a binary value indicating whether it is present. (If the switch, e.g. *Antivirus Software?*, is off, the factor output will always be 0.) All of the countermeasures are represented by holding tanks. A "limit" block applied to the tank's contents level ensures that the output will be between 0 and 1. *However*, the feature "tank itself can not exceed 1", i.e. "if tank capacity = 1, today's input = 0", was not yet built into the countermeasures as it was into the vulnerabilities. (This can be easily changed.) Thus, for now, if the sysadmin were to put "super" efforts into a countermeasure for a while, the tank level would exceed 1; the factor output will still be 1; however, the sysadmin could ignore the countermeasure for a short time and it will still have a factor of "1", as the loss rate drains the tank from a value greater than 1 to 1. (Eventually, though, the tank will drain below 1.)

Comment [i7]: Page: 1
I don't understand

Comment [i8]: Page: 1
I believe THIS IS realistic

Starting level of the tank, if not manually adjusted, will follow the same equation as the vulnerability tanks: (hours allocated) / (total hours needed for the countermeasure to be "happy", = divisor * lossrate). (This is then limited between 0 and 1.)

Tolerance Mechanisms. These can be high-maintenance, as this includes diversity. Loss rate 0.1, divisor 6.67. [Why?] It was just assumed that tolerance measures are high-maintenance, especially if we include diversity; so I picked values these values: if tolerance measures are ignored for ten days, they become useless (loss rate 0.1); and that 2/3 pseudo-staff-hours per machine are required to keep these tolerance measures fully maintained.

Encryption Software. Once in place, this is fairly low-maintenance. Loss rate 0.001, divisor 66.6667. [Why?] Again, I just picked numbers that would imply low-maintenance, i.e. an encryption system, once ignored, takes > 2 years (i.e. 1000 days) to become useless (there still may be bug fixes, updates, and the like); and it doesn't take much work to keep the encryption up (or deal with users having problems with it), so I just figured an average of .067 pseudo-staff-hours per machine per day.

Antivirus Software. We assume that new definitions must be installed by the sysadmin. Loss rate 0.02 (i.e. useless after fifty days, given that ~2.5 new viruses come out each day, looking at a list from McAfee or the like.) Divisor 20. How much time per day per machine is needed to keep the antivirus up-to-date? I assumed 0.4 pseudo-staff-hours / machine / day.

Firewall and IDS [Indeed, they should be separate.]. We described "firewall efforts" and "IDS efforts" each as separate from "hours for analyzing (false) alarms", which includes the alarms generated by both. Thus, the *FirewallIDS* forms one unit, with inputs: *Hours Allocated to Firewall, Hours Allocated to (False) Alarms, Hours Allocated to IDS,* and the binary switches *A Firewall?* and *An Intrusion Detection System?*. Outputs are *FirewallFactor* and *IDSFactor*, both within [0,1]; and *FalseAlarms,* measuring how many staff-hours-per-machine's worth of alarms are generated on a given day.

Comment [i9]: Page: 1
why are these not separate?

Comment [i10]: Page: 1
why? Shouldn't they be separate?

Firewall effectiveness and IDS effectiveness each start off as independent holding tanks similar to those of the other countermeasures. Thus, *FirewallFactor* and *IDSFactor* are simply the contents value of their respective tanks. Firewall has

LossRate 0.0333 and Divisor 20; IDS has LossRate 0.033 and divisor 8. [Why these numbers?]

However, both the IDS and firewall generate more alarms as they become more effective.

*AlarmRate = (0.3 * FirewallFactor)+ (0.6 * IDSFactor).[No, they should not be mixed.]*

If these alarms are not addressed, they become *Ignored Alarms.*

IgnoredAlarms = AlarmRate – AlarmHours, with a minimum of 0. All of these are measured in staff-hours per machine per day.

IgnoredAlarms leads to a steep decline in the effectiveness of the IDS and firewall, with several days' delay.

The "want" (i.e. drain) on firewall effectiveness is the "natural" loss rate due to the need for routine maintenance, patches and the like, which was given as 0.0333; plus 0.25 * *IgnoredAlarms,* with a five-day delay on the latter.

Similarly, drain on IDS effectiveness is 0.0333 (natural loss rate), plus 0.33 * *IgnoredAlarms,* with a three-day delay on the latter.

Based on this, running the model with a high number of hours dedicated to the IDS or firewall, but few hours to analyzing alarms, will result in IDS and firewall effectivenesses that show decaying nonnegative oscillations, i.e. high, then low, then medium, then low, and so on, until they reach a level of zero.

Archetype Version: Here, we wanted to show gains per effort for a single variable, so we included alarm analysis into the IDS efforts (and hours). (The firewall model was unchanged, as our archetypes did not include a firewall.) We now have a holding

tank with a starting level of zero, and an input divisor of 40. Loss rate is now entirely a factor of the IDS effectiveness: LossRate = 0.35* IDSlevel [i.e. the contents of the holding tank], with a 15-day delay. This causes the oscillations seen in the attack success rate of the *Shifting the Burden* IDS scenario.

We then argue that even an ignored IDS will still catch *some* attacks; this assumption also keeps the oscillations in the archetypes paper from being too extreme. This is accomplished by simply letting *IDSFactor = (contents of holding tank) + 0.6*, with a maximum of 1. This means that a totally ignored IDS will still have 60% the effectiveness of a well-maintained one. (The author claims no sources in the literature to support this, other than "it made the graph look nice.")

Newest version: we've kept everything separate: addressing IDS alarms goes into IDS Efforts; addressing Firewall alarms goes into Firewall Efforts. Firewall has loss rate .033 (i.e. useless if ignored for 30 days, just my assumption) and divisor 20 (i.e. for full effectiveness, firewall should have 0.66 staff-hours / day; in this paper, I assumed .66 staff-hours per day per machine. (That's too high, isn't it? Again, that's skill-motivation-staff-hours, which is easily double the number of actual staff-hours.) For the IDS, loss rate 0.05 (i.e. useless if ignored after 20 days, again my assumption), and a divisor of 40, i.e. best to provide the IDS 2 staff-hours per day (per system).

ATTACKS: We divide the attacks into three categories by their sophistication. (This three-way division is found in some DARPA presentations that have not yet been published.) *Simple Attacks*, (or "kiddy-scripts"), almost always rely on known

vulnerabilities and require little action from the attacker other than downloading and running the attack. A "sitting-duck" server may be subject to 50 or more simple attacks per day. Dr. Cukier's empirical findings support roughly this number. (Though his ~50 did not include viruses.)

"*Sophisticated Attacks"* may involve finding new vulnerabilities, can often defeat many countermeasures, and usually come from a single knowledgeable attacker (such as one who might actually write the "kiddy scripts" used in the first category). The average company will sustain only a handful, at most, of sophisticated attacks per day. (Yup, just an assumption; Dr. Cukier is trying to get sophisticated attackers to hit his systems, but not much luck yet. Wasn't there a quote from Dr. Cukier about 95% simple / 5% sophisticated or something like that?) Certainly we must include computer viruses, the most costly computer-security breach as

reported in the CSI/FBI survey, in our discussion. While a computer virus does require a sophisticated author if it will spread, it spreads in fairly simple, predictable ways, and is easily defeated by simple countermeasures (antivirus) and patching vulnerabilities; we therefore include viruses in the simple, "script-kiddy" category.

Lastly, we have *Nationwide*-level *Attacks***,** which may be part of a war effort, global terrorism, possibly a multinational corporation attacking a competitor, and so on. Most companies will only see one of these every few months or so, if at all. (That seemed like common sense.) Attacks of this sophistication do not rely on vulnerabilities as they can "brute force" through most software; they can also defeat most countermeasures.

For the time being, we do not differentiate attacks other than their categories of *Simple, Sophisticated,* and *Nationwide.* The model's end-user inputs *AverageSimple, AverageSophisticated,* and *AverageNationwide* via sliders. The outputs of the respective "attack generator blocks" are *Simple/Sophisticated/Nationwide Attacks Attempted*. To add realism to our model, some randomness occurs between the input *Average*s and the output *Attempted*s:

All of the above behave the same way. If *Average* >= 1, then a number Y is output, where Y represents the number of attacks of that type attempted per day. Y is given by a Gaussian distribution, with mean *Average* and a standard deviation of 0.2 * *Average.* (i.e. "a standard deviation of 20%.")

If *Average* < 1, then exactly one attack is attempted every Z days, where Z follows a Gaussian distribution with a mean of (1/Average) and a std. dev. of 30%.

Archetype Version: for simplicity, and to prevent oscillations in the graph due to randomness, we simply let *AverageSimple* = *SimpleAttacksAttempted* = 100. (We circumvent the "attack generator block.") All other attacks are set to 0.

In the new (*Escalation*) paper, I had no randomness in Simple Attacks, but a 5% standard deviation in Sophisticated Attacks. Just numbers I picked to demonstrate some randomness; I don't know how much the numbers vary day-to-day in real-life.

ATTEMPTED VS. SUCCESSFUL ATTACKS, or ATTACK DEFENSES. Even if perfectly effective, a given countermeasure is only so successful at thwarting attempted attacks. For example, if we say 100 attacks are attempted per day, we include a certain number (call it X) of viruses. The best antivirus in the world will

thwart all X of those viruses, i.e. X% of the total attacks, but it can not defeat more than X% of the attacks because (100-X) attacks are not viruses. As for what percentage of attacks are not successful due to a given countermeasure, the only numbers available are those of experts' opinion and the CSI/FBI survey. The CSI/FBI survey is of limited use, however, as it records what percentage of correspondents reported observing a given type of attack on their system. Thus, we know that 78% of the businesses surveyed detected a virus last year, and 37% detected a DoS; that does not mean that 78% of the attacks out there are viruses or that 37% of them are DoSs! (Otherwise, the numbers exceed 100% quite rapidly.) Nonetheless, the numbers can be used as a very rough approximation for the prevalence of a given attack. Otherwise, the numbers given here represent the author's numerical interpretations of M. Cukier's descriptions of "fully effective", "partially effective", or "not effective" for each countermeasure against each category of attack.

Similarly, certain countermeasures may be very effective against simple attacks, but not against sophisticated ones. We therefore have three different blocks labeled *SimpleSuccess*, *SophistSuccess*, and *NatnwideSuccess*, respectively. (Extend's limits on the number of characters in a hierarchical block's name necessitated some creative spellings here.) Each of these takes as inputs *Attempted XYZ Attacks*, where XYZ is simple/sophisticated/nationwide. They also have inputs for the factors of all relevant countermeasures and vulnerabilities. The primary output is the number of successful attacks of a given category. The other outputs appear on the bottom of the AttackSuccess block, directly beneath the inputs for the various countermeasures and vulnerabilities. These outputs show the number of attacks per day not successful due

to the corresponding countermeasure or lack of vulnerability. Additionally, it appears that a “thinner” block has been attached to the bottom of each Attack Success block. This functions as an accumulator, showing how many attacks have been attempted, successful, or not-successful-due-to-a-given-factor, over the entire simulation period.

Each *AttackSuccess* block is designed in the same way, as a linked series of holding tanks: at the beginning of each day, all tanks are reset to zero. Then, a certain number of attacks (attempts) are input to the first holding tank; some are removed by the first countermeasure (in proportion to how effectively it is functioning, e.g. is the *AntivirusFactor* 1, i.e. it has been well-maintained, or something lower?); the remaining tank contents (i.e. remaining attacks) are transferred to the second tank, where some are removed by the second countermeasure, and so on; those that remain after all the tanks are done are deemed *Successful Attacks*.

For simple attacks, we have the following procession: As an attempted attack enters the system, it first encounters the firewall, then an IDS; if it passes those, it will be scanned by an antivirus. If it still passes through, it may be designed to exploit a given vulnerability in the system; if that vulnerability is not present, it will be thwarted here. If it still succeeds, encryption may sometimes help as follows: even if the system is breached and data is illegally accessed, an attacker will find the encrypted data meaningless; confidentiality is thus maintained. Finally, if all else fails, tolerance measures will mitigate the damage in many cases. Thus, starting with attempted simple attacks, we have the following: [Not completely right. In particular, the antivirus focuses mainly on email attachments. Otherwise, the antivirus can detect

Comment [i11]: Page: 1
Michel, what do you think?

the corruption of the computer. We can work this out during our next meeting, OK?] We're still working on this, but the models haven't changed it yet.

Remove (*FirewallFactor* * 90%*)* of the attempted attacks. I.e. if the Firewall is fully effective, it will catch 90% of the attempted simple attacks; if it's only 50% effective (supposing it hasn't been well-maintained), then it will catch only 45% of attempted attacks. Of those remaining, remove (*IDSFactor* * 60%); of those remaining, remove (*AntivirusFactor* * 78%); of those remaining, remove ((1 – *VulnFactor*) * 90%); this represents those attacks that were designed to exploit a given vulnerability; if that vulnerability is not found, the attack will not succeed.

Archetypes Version: in order to differentiate between the results of enforcement actions (which influence config vulns and mistakes) and patches (which influence NetVulns and AppVulns), we have each defeat attacks separately, rather than taking 90% * (1 – VulnFactor). Instead, remove (*SWStrength* * 60%), then (*ConfigStrength* * *80%).* In displaying those attacks defeated by *ConfigStrength,* we adjust the equations to show total attacks defeated by *ConfigStrength*, not those attacks defeated by *ConfigStrength* that were not previously defeated by *SWStrength*.

Then remove (*EncryptFactor* * 40%). (Encryption is only useful in preventing theft of data; it does very little, for example, against a DDoS attack.) Lastly, remove (*ToleranceFactor* * 75%). Take this result and apply the "floor" function, i.e. largest integer that is less than or equal to it. (Thus, if after all the countermeasures, we have 2.2 attacks succeeding, count that as 2. If we have 0.9 attacks succeeding, count that as 0.) Archetype version: to make the lines smoother, we leave out the floor, and instead interpret the results simply as "percentage of attacks succeeding." We now

have the number of *SuccessfulSimpleAttacks*. (The various summed-over-time outputs are found simply by inserting accumulation tanks at the appropriate point in the chain.)

All of these percentages were either my assumptions, some comment from Dr. Cukier about "very effective/somewhat effective/not effective", and occasionally, the survey (see above about 78% saw viruses.)

For Sophisticated and Nationwide attacks, many less countermeasures are effective. Furthermore, even a single attack stands a good chance of succeeding. This is represented as follows: after reducing the appropriate percentages due to countermeasures and vulnerabilities, we are left with what should be X successful attacks. If X >=1, round X to the nearest integer; that is how many attacks of this type are successful today. If $0 < X < 1$, one attack will succeed an average of X% of the time. This is accomplished by selecting a random value r uniformly distributed on [0,1]; if $r < X$, the attack succeeds; otherwise, it does not.

For sophisticated attacks, the antivirus is ineffective because all viruses are treated as simple attacks. An IDS can be defeated by a clever attacker, so it is not included. Encryption (which we assume can not be defeated without a supercomputer of some type (Dr. Cukier agreed with this; I've heard in the news that every now and then a team of experts with 100 computers has cracked a given file encrypted with RSA, after working on it for a few months.) which is beyond the reach of a single sophisticated attacker) is still as effective as with simple attacks; the same goes for tolerance. A firewall is effective, but less so because it can sometimes be defeated.

Lastly, some sophisticated attacks are designed to exploit known vulnerabilities, but often a sophisticated attacker can find his/her own new vulnerabilities in the software. We thus are left with the following:

Remove (*FirewallFactor* * 30%) of attempts; of the remaining, remove ((1 – *VulnFactor*) * 50%); of the remaining, remove (*EncryptFactor* * 40%); lastly, of the remaining, remove (*ToleranceFactor* * 75%). The remaining value is rounded to the nearest integer if it is >= 1, or used as a probability if it is < 1, as described above. The result is the number of *SuccessfulSophisticatedAttacks*. Only source other than Dr. Cukier's comments or my guesses I can add here is this: Encryption is helpful for whatever percentage of attacks sought to steal sensitive data. What is that percentage? Survey talks dollar costs of various attacks (e.g. theft of data vs. DoS), but not the percentage breakdown of the number of attacks themselves.

In the case of Nationwide attacks, we assume that network and application vulnerabilities are irrelevant as the code is subject to "strong smart force", the nationwide-scale attackers may have access to the code being used; similarly, the nationwide attacker possesses a supercomputer, quantum computer, or some other method of defeating commercially available cryptography. The only countermeasures that are effective (and partially at that) are the firewall (if it is a hardware firewall of proprietary design, as M. Cukier described in an experience of his) and tolerance measures.

Remove (*Firewall Factor* * 20%), then of the remaining, remove (*ToleranceFactor* * 50%). Apply the rounding or probability as described above; the result is the number of *SuccessfulNationwideAttacks*. Dr. Cukier had said something

about Tolerance being fully effective against simple & sophisticated attacks; halfway effective against nationwide attacks.

Lastly, the three categories of successful attacks can be summed; each AttackSuccess block is connected to an addition block. The result is *All Successful Attacks (Per Day).* Similarly, if one wishes to see all successful attacks over the entire simulation period, the various *Successful ABC Attacks (Sum Total),* for ABC = {Simple, Sophisticated, Nationwide}, sum to *All Successful Attacks (Sum Total).*

Bibliography

[Baj04] R. Bajcsy et al, "Cyber defense technology networking and evaluation," *Communications of the ACM*, vol. 47, no. 3, March 2004, pp. 58—61.

[Ber04] D. Bergemann, J. Feigenbaum, S. Shenker, and J. Smith, "Towards an economic analysis of trusted systems," presented at Third Annual Workshop on Economics and Information Security (WEIS '04). Minneapolis, May 13—14, 2004.

[Ber05] D. Bergemann et al, "Flexibility as an instrument in DRM systems," presented at Fourth Annual Workshop on Economics and Information Security (WEIS '05). Cambridge, MA, June 2—3, 2005.

[Bod05] L. Bodin, L. Gordon, and M. P. Loeb, "Evaluating information security investments using the analytic hierarchy process," *Communications of the ACM*, vol. 28, no. 2, Feb. 2005, pp. 79—83.

[Bra02] W. Braun, "The System Archetypes," (2002), Available at http://www.uni-klu.ac.at/gossimit/pap/sd/wb_sysarch.pdf

[Cam03] K. Campbell, L. A. Gordon, M. P. Loeb, and L. Zhou, "The economic cost of publicly announced information security breaches: empirical evidence from the stock market," *Journal of Computer Security*, no. 11, 2003, pp. 431—439.

[Coh99] F. Cohen, "Simulating CyberAttacks, Defenses, and Consequences," (March 1999), Available at: http://all.net/journal/ntb/simulate/simulate.html

[Coh06] Fred Cohen & Associates, "Strategic Games", (2006), http://all.net/games/index.html

[Dac04] M. Dacier, F. Pouget, and H. Debar, "Honeypots: practical means to validate malicious fault assumptions," In *Proc. 10th IEEE Pacific Rim Int'l Symposium on Dependable Computing (PRDC '04)*, 2004, pp. 383—388.

[Dan04] D. Danchev, "Reducing 'Human Factor' Mistakes." WindowSecurity.com, (2003), www.windowsecurity.com/articles/Reducing_Human_Factor_Mistakes.html

[Dwa05] Z. Dwaikat, "Attacks and countermeasures," *CrossTalk: The Journal of Defense Software Engineering*, Oct. 2005, Available at: www.stsc.hill.af.mil/crosstalk/2005/10/0510Dwaikat.html

[Fei05] J. Feigenbaum et al, "Subjective-cost policy routing," in *Proceedings of the Workshop on Internet and Network Economics, Lecture Notes in Computer Science*, vol. 3828, Berlin: Springer, 2005, pp. 174--183.

[For61] J. W. Forrester, *Industrial Dynamics*, Cambridge: The Wright-Allen Press, 1961.

[Gib02] S. Gibson, "The Strange Tale of the Denial of Service Attacks Against GRC.com." Gibson Research Corporation, (March 2002), Available at: www.grc.com/files/grcdos.pdf

[Gon01] F. Gong, K. Goseva-Popstojanova, et al, "Characterizing intrusion tolerant systems using a state transition model," In *Proc. DARPA Information Survivability Conference and Exposition II* (*DISCEX '01)*, 2001.

[Gor02] L. A. Gordon and M. P. Loeb, "The economics of information security investment," *ACM Trans. Information and System Security*, vol. 5, no. 4, November 2002, pp. 438—457.

[Gor05a] L. A. Gordon, M. P. Loeb, et al, "Tenth Annual CSI/FBI Computer Crime and Security Survey," Computer Security Institute, (2005), Available at: www.gocsi.com/forms/fbi/csi_fbi_survey.jhtml

[Gor05b] L. A. Gordon and M. P. Loeb, *Managing Cybersecurity Resources: A Financial Perspective*, New York: McGraw-Hill, 2005.

[Hof05] S. Hofmeyr, "The Information Technology security arms race," *CrossTalk: The Journal of Defense Software Engineering*, Oct. 2005, http://www.stsc.hill.af.mil/crosstalk/2005/10/0510Hofmeyr.html

[Hun06] C. L. Huntley, "A Developmental View of System Security," *Computer*, vol. 39, no. 1, pp. 113—114, January 2006.

[Ima05] Imagine That, Inc., *Extend.* (Version 6.07). [CD-ROM]. [Windows 98/ME/NT4/2K/XP]. San Jose, CA, 2005.

[Irv05] C. E. Irvine, M. F. Thompson, and K. Allen, "CyberCIEGE: gaming for information assurance," *IEEE Security & Privacy Magazine*, vol. 3., no. 3, May-June 2005, pp. 61—64.

[Jha01] S. Jha and J. M. Wing, "Survivability analysis of networked systems," in *Proc. of the 23rd International Conference on Software Engineering (ICSE '01)*, 2001, pp. 307—317.

[Jon97] E. Jonsson and T. Olovsson, "A quantitative model of the security intrusion process based on attacker behavior," *IEEE Transactions on Software Engineering*, vol. 23, no. 4, April 1997.

[Lar03a] R. LaRose and M. S. Eastin, "A social cognitive explanation of Internet uses and gratifications: toward a new theory of media attendance," presented at International Communication Association, Communication and Technology Division. San Diego, May 2003.

[Lar03b] R. LaRose and N. Rifon, "Your privacy is assured --- of being invaded: Web sites with and without privacy seals," presented at IADIS International Conference. Lisbon, Portugal, June 3—6, 2003.

[Mar03] K. Marais and N. Leveson, "Archetypes for organizational safety," presented at IRIA '03. Williamsburg, VA, 2003, Available at: http://sunnyday.mit.edu/papers/iria-marais.pdf

[Nav06] Naval Postgraduate School and Rivermind, Inc., *Cyberciege*, version 1.5b, Feb. 2006, [Win2000/XP] http://cisr.nps.navy.mil/cyberciege/index.htm

[Ort99] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, Sept.-Oct. 1999, pp. 633—650.

[Pan05] S. Panjwani et al, "An experimental evaluation to determine if port scans are precursors to an attack," in *Proc. International Conference on Dependable Systems and Networks (DSN05)*, Yokohama, Japan, June 28—July 1, 2005.

[Pou04a] F. Pouget and M. Dacier, "Honeypot-based forensics," presented at AusCERT Information Technology Security Conference 2004 (AusCERT '04). Ashmore, Australia, May 23—27.

[Pou04b] F. Pouget, M. Dacier, and V. H. Pham, "Understanding threats: a prerequisite to enhance survivability of computing systems." Presented at International Infrastructure Survivability Workshop 2004 (IISW04), in conjunction with 25th International Real-Time Systems Symposium (RTSS04). Lisbon, December 5—8, 2004, Available at: http://www.honeynet.org/papers/individual/IISW04.pdf

[Pou05] F. Pouget, M. Dacier, and V. H. Pham, "Leurre.com: On the advantages of deploying a large scale distributed honeypot platform," In *Proceedings E-Crime and Computer Conference 2005 (ECCE '05)*, March 2005.

[Ros06a] S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling and simulation of the escalation archetype in computer security," presented at *2006 Symposium on Simulation and Software Security (SSSS '06)*. Huntsville, AL, April 2006.

[Ros06b] S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling the symptomatic fixes archetype in enterprise computer security," submitted to *30th Annual International Computer Software and Applications Conference (COMPSAC06)*. Chicago, Sept. 2006.

[Ros06c] S. N. Rosenfeld, I. Rus, and M. Cukier, "Archetypal behavior in computer security," submitted to *Sixth European Dependable Computing Conference (EDCC-6)*. Coimbra, Portugal, Oct. 2006.

[Sen90] P. M. Senge, *The Fifth Discipline: The Art and Practice of the Learning Organization*, New York: Doubleday Currency, 1990.

[Sen94] P. Senge, A. Kleiner, et al, *The Fifth Discipline Fieldbook*, New York: Doubleday, 1994.

[Ste04] F. Stevens, T. Courtney, et al, "Model-based validation of an intrusion-tolerant information system," In *Proc. of the 23rd Symposium on Reliable Distributed Systems (SRDS '04)*, 2004, pp. 184—194.

[Wol03] E. F. Wolstenholme, "Toward the definition and use of a core set of archetypal structures in system dynamics," *System Dynamics Review*, vol. 19, no. 1, Spring 2003, pp. 7—26.

Publications and Submissions

- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling and simulation of the escalation archetype in computer security," 2006 Symposium on Simulation and Software Security (SSSS '06). Huntsville, AL, April 2006. (50% acceptance rate.)
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modeling the symptomatic fixes archetype in enterprise computer security," submitted to *30th Annual International Computer Software and Applications Conference (COMPSAC06)*. Chicago, Sept. 2006.
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Archetypal behavior in computer security," submitted to *Sixth European Dependable Computing Conference (EDCC-6)*. Coimbra, Portugal, Oct. 2006.
- S. N. Rosenfeld, I. Rus, and M. Cukier, "Modelling the tragedy of the commons archetype in enterprise computer security," soon to be submitted to *IEE Proceedings Information Security*.