Using STAMP to Understand Recent Increases in Malicious Software Activity

by

David S. Zipkin

B.A. Computer Science
Dartmouth College, 1997

Submitted to the Engineering Systems Division
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Technology and Policy

at the

Massachusetts Institute of Technology

June 2005

Signature of Author.....
Technology and Policy Program, Engineering Systems Division
May 13, 2005

Certified by.....
Nancy G. Leveson
Professor of Aeronautics and Astronautics
Professor of Engineering Systems
Thesis Supervisor

Accepted by.....
Dava J. Newman
Professor of Aeronautics and Astronautics and Engineering Systems
Director, Technology and Policy Program

Using STAMP to Understand Recent Increases in Malicious Software Activity
by
David S. Zipkin

Submitted to the Engineering Systems Division on May 13, 2005
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Technology and Policy

Abstract

My advisor, Professor Leveson has developed an accident modeling framework called STAMP (Systems Theoretic Accident Modeling and Processes.) Traditional accident models typically focus on component failure; in contrast, STAMP includes interactions between components as well as social, economic, and legal factors.

My research extends Leveson's STAMP accident model and applies it to a security problem. I have chosen to investigate the threat posed by malicious computer software such as computer viruses. The problem is especially interesting because surrounding the technical aspects of malicious software is a rich socio-technical system.

The first part of the thesis investigates two recent computer worm outbreaks and identifies the numerous ways in which the security system failed. For both outbreaks, there were multiple points of failure including the existence of un-patched workstations, software organizations that distributed insecure software, the lack of sufficient legal disincentives to dissuade hackers, as well as many others.

The thesis goes on to examine why the system was operating in such an insecure manner. As is generally the case when modeling an accident, the explanation goes beyond any single factor. I argue that that lack of Internet security can be largely attributed to the fact that those providing critical parts of Internet security do not have sufficient incentives to make good security decisions; instead they often make decisions at odds with Internet security.

The thesis concludes with a discussion of policy and technical recommendations for addressing computer security.

Thesis Supervisor: Dr. Nancy G. Leveson
Professor of Aeronautics and Astronautics
Professor of Engineering Systems

To Abby

for making my final week of thesis writing forever memorable

Table of contents

Chapter 1—Introduction and Background

This thesis has a dual purpose. Its principal purpose is to test the hypothesis that Leveson's STAMP (Systems Theoretic Accident Modeling and Processes) framework can be successfully applied to a problem in the security realm. To test this hypothesis, I used STAMP to analyze the increasingly critical problem of malicious software (malware) that is threatening the utility of the Internet.

The secondary purpose of the thesis is a complete analysis of the systemic causes of malicious software using STAMP. I believe this thesis contains an analysis that yields significant understanding about the causes of the malicious software problem, which is a strong indication that STAMP can be used to analyze security problems as well as safety accidents.

The thesis contains seven chapters. The remainder of the first chapter is devoted to giving the reader an understanding of computer security and an appreciation of the magnitude of the problem at hand. Chapter 2 provides background about STAMP and discusses why I believe STAMP can be used in a security context.

Chapters 3, 4, and 5 constitute the STAMP analysis. Chapter three defines the Internet Security System (ISS) in detail, which is the collection of components whose interactions determine the ability of the Internet to withstand attacks. Chapter four investigates how the ISS has repeatedly failed. It dissects two examples of security failures and details which parts of the ISS failed to protect against the malicious software. Chapter 5 discusses the changes to the ISS that occurred over the past decade and explains how these changes weakened the ISS. It also provides a discussion of the forces that drove those changes.

Chapter 6 is devoted to a discussion of technical and policy-oriented approaches intended to improve the Internet's resilience to malicious software and chapter 7 is a brief summary of the report.

I believe that STAMP served as a very useful tool for conducting the security analysis. Using STAMP, I was able to take a diverse and complex security system and develop a strong understanding of how the system responds to the challenges it faces. By conducting the STAMP analysis, I arrived at the central conclusion presented in chapter

5: those providing critical parts of Internet security do not have sufficient incentives to make good security decisions; instead they often make decisions at odds with Internet security.

Defining the problem

Attacks on the Internet can be divided roughly into two groups: targeted attacks and broad attacks. This thesis focuses on broad attacks in which an attacker uses an attack method, such as a type of computer virus known as a worm, to reach as wide an audience as possible.[1]

Broad attacks have evolved beyond their early incarnations, when the author's main purpose was frequently to draw attention to his programming prowess by creating as wide an impact as possible. In recent years, attackers have found creative ways to earn money from broad attacks and the popularity of broad attacks are increasing. Attackers use these attacks to compromise large numbers of computers, installing a piece of malicious software on those computers without the computer owner's knowledge or acquiescence that gives the attacker use of the victim's computer.

The author of this malicious software can make money by charging others for use of the computers he has compromised. Others are willing to pay for the use of these compromised computers and put them to a variety of unsavory uses, including:

- Sending spam
- Extortion and Blackmail via Denial of Service Attacks
- Stealing personal or financial information
- Running "Phishing" Servers
- Advertising click fraud
- Terrorism

The mechanics of malicious software

In order to understand the scope and threat posed by malicious software, it is useful to know more about its inner workings. As varied as the purposes of broad attacks

[1] An example of a targeted attack is when a hacker attempts to gain access to or disable a computer resource belonging to a specific person or organization.

can be, all rely on infecting as many computers as possible with malicious software. Malicious software spreads in two primary ways:

- a user unintentionally, but voluntarily runs a piece of malicious software. The malicious software can be delivered in various ways; it may be an email attachment or perhaps a link included in an instant messenger (IM) message
- the malicious software can install itself without any human intervention by exploiting a vulnerability in software already running on the computer

Once a computer has been infected with the malicious software, two things happen: propagation and payload installation. First, the newly infected computer assists the spread of the malicious software by attempting to infect other computers. See Figure 1 and Figure 2 below.

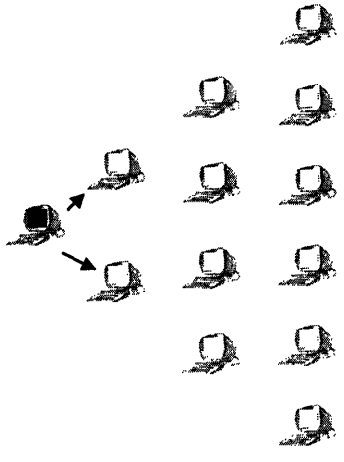


Figure 1—A single computer is infected

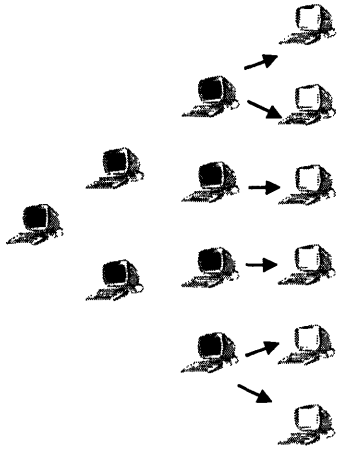


Figure 2—Infected computers work to find other vulnerable computers

The software installed during the propagation phase contains instructions to spread the malicious software. It also contains a payload. Depending on the malicious software's intentions, the purpose of the payload can vary. The three common payload types are:

- *Remote control*—this type of payload allows the compromised computer to be used for one of the financial ventures described above. Such payloads might

make a computer into an email relay or instruct it to participate in a denial of service attack

- *Backdoor*—this type of payload allows additional software (usually remote control) to be installed at a later time by the malicious software author
- *Destructive*—destructive payloads cause damage to data and computers. For instance, a destructive payload may erase data from a hard drive.

Regardless of the type of payload installed, the result is a compromised computer. Perhaps surprisingly, destructive payloads are uncommon. Rather, computers compromised with remote control payloads are frequently instructed to join networks of compromised computers called *botnets.*

A *botnet* (short for ro*bot net*work) is a collection of compromised computers that take instructions from a single leader. Some botnets consist of upwards of 50,000 computers[2] and researchers estimate that more than a million computers are members of at least one botnet.

The owners of these botnets rent out the processor time of the computers they control. The going rate for a compromised computer ranges from approximately $0.04 to $0.08[3] per week of use. (Appendix 1 contains online advertisements offering the use of a botnet.) A small botnet of 1000 computers can supplement the owner's income by a few thousand dollars a year. Larger botnets can provide significant income to malicious software programmers. (A 50,000-computer botnet, fully utilized, can yield close to $200,000 a year.)

Uses of Botnets

As outlined above, there are a number of creative uses for such a distributed network of computing power. It enables bad actors to more effectively engage in multiple kinds of nefarious behavior:

---

[2] Know your Enemy: Tracking Bots. 13 Mar. 2005. The Honeynet Project & Research Alliance. 2 Apr. 2005 <http://http://www.honeynet.org/papers/bots/>.
<http://http://www.honeynet.org/papers/bots/>.

[3] LaMacchia, Brian A. Security Attacks and Defenses. Working Group on Dependable Computing and Fault Tolerance. Information Systems and Organizations. Puerto Rico, US. 29 Jan. 2005. 1 Apr. 2005 <http://www.laas.fr/IFIPWG/Workshops&Meetings/47/WS/08-LaMacchia.pdf>.

Sending spam

Botnets are well suited for sending spam. By using hijacked home and office computers, spammers can circumvent common anti-spam techniques used by ISPs and firms, such as blacklisting, to stop spam. Additionally, they save on bandwidth costs because they use bandwidth stolen from the hijacked computer. Researchers estimate that hijacked computers send 66% of the spam on the Internet today.[4]

Extortion and Blackmail via Denial of Service Attack

The owner of a botnet can attempt to disable a service (such as a website) that is running on another server. By concentrating the resources of the botnet on this service, the attacker can overwhelm it and render it inaccessible to others.[5] This is known as a distributed denial of service attack (DDoS). In June 2004, a botnet DDoS attack on Akamai Technologies brought down the websites of Google, Microsoft, and Yahoo.[6]

Consider a botnet of 5000 home computers on average broadband connections. If each computer devoted 200 kbps (kilobits per second) to the DDoS attack, the resulting 1000 mbps (megabits per second) would be sufficient to occupy nearly all of the DDoS victim's bandwidth.

With their ability to disable websites proven, botnet owners have been able to extract tens of thousands of dollars in "protection fees" from high-revenue websites in return for not launching denial of service attacks[7].

Stealing personal or financial information

The owner of a botnet can also instruct the compromised computers under his command to search their hard drives for sensitive information such as social security numbers or financial information. Some types of malicious software can be instructed to

---

[4] LaMacchia.

[5] A useful analogy for understanding denial of service attacks is a fast food restaurant. These restaurants can comfortably serve 20 or so customers. A denial of service attack on a fast food restaurant would be like thousands of people waiting in line to buy food, but not actually purchasing anything. Any legitimate customers would be forced to wait in extraordinarily long lines and effectively denied service.

[6] Biever, Celeste . "How zombie networks fuel cybercrime." New Scientist 3 Nov. 2004. 5 Mar. 2005 <http://www.newscientist.com/channel/info-tech/electronic-threats/dn6616>.

[7] Biever.

search computers for the license information of well-known software.[89] Later, this information can be sold.

Running "Phishing" Servers

In a "phishing" attack, a user receives a forged email designed to appear as if it came from a legitimate organization. The email requests that the user updates his personal information, such as a credit card number or bank account password. If the user follows the link given in the email, he is taken to a website that looks like the legitimate organization's site but, of course, is not. Compromised computers are often employed to host the counterfeit websites and capture the sensitive information.

Advertising click fraud[10]

Botnets have also been used to tamper with online advertising. Google and Yahoo developed a popular advertising business model where advertisers pay each time their ad is clicked. Advertisers bid for popular keywords; the cost of each click is typically around $1 but very popular keywords can exceed $30. (Appendix 4 contains pricing information for select keywords.)

An extension of this business model allows web publishers to devote a portion of their web pages to hosting advertisements. Here, the web publishers receive a portion of the revenue generated.

Commanding the computers in a botnet to execute searches and click on specific advertisements will fraudulently inflate the number of times an advertisement is clicked and will result in increased costs for the advertiser as well as increased revenues for the web publisher who hosts the advertisements. Botnets have been used to both drive up a competitor's advertising costs and to inflate the revenues of the web publisher.[11]

How big a problem is this?

8

[9] Phatbot Trojan Analysis. 15 Mar. 2004. Lurhq Corporation. 2 May 2005 <http://www.lurhq.com/phatbot.html>.

[10] Vise, David A. "Clicking To Steal." Washington Post 17 Apr. 2005. 2 May 2005 <http://www.washingtonpost.com/wp-dyn/articles/A58268-2005Apr16.html>.

[11] Ives, Nat. "Web Marketers Fearful of Fraud In Pay-Per-Click." New York Times 3 Mar. 2005, sec. C:1. Lexis Nexis. 2 May 2005

While we do not know the total amount of economic damage caused by malicious software, we can gauge the intensity with some statistics:

- Over 1,000,000 computers on the Internet are compromised and controlled by malicious attackers[12]
- Botnets have been known to constitute upwards of 50,000 computers[13]
- 2/3 of spam on the Internet is sent by bots[14]
- There were approximately 200 denial of service attacks per day at the end of 2004[15]

If the Internet becomes increasingly hazardous and lawless, I expect that people will begin to go offline, retreating to intensely maintained private corporate networks or simply choosing not to use the Internet at all.

However, the potential for future damage is significantly worse. While there have been surprisingly few outbreaks of malicious software with destructive payloads, there have been a large number of successful worms with remote access payloads. A successfully propagating worm could just as easily have a destructive payload.

Researchers at Berkeley's International Computer Science Institute found that a well-designed worm could, theoretically, tear through the Internet and infect 95% of 1 million vulnerable computers in slightly more than ½ of a second.[16] Termed a *flash worm*, it could potentially deliver a truly malicious virus to a large amount of computers at a virtually unstoppable rate.

Such a worm would be appealing to terrorists or enemy states. Rather than simply leaving behind back doors that might result in more zombie computers, such a worm could have truly malicious intent, perhaps deleting data or rendering computers

---

[12] Know your Enemy: Tracking Bots.
[13] Know your Enemy: Tracking Bots.
[14] LaMacchia.
[15] Symantec Internet Threat Report: Trends for July 04 - December 04. Vol. VII.: Symantec Corporation, 2005. 1-96.
[16] Moore, David, et al. "The top speed of flash worms." Proceedings of the 2004 ACM workshop on Rapid malcode (2004): 33-42. 2 May 2005 <http://portal.acm.org/ft_gateway.cfm?id=1029624&type=pdf&coll=GUIDE&dl=GUIDE&CFID=42889884&CFTOKEN=47157339>.

unusable. It could also be used to launch an immense denial of service attack on critical infrastructure.

In order to achieve phenomenal spread promised by a flash worm, the writer of the worm would need to detailed information about many computers. *Botnets are uniquely suited to gather this type of information.*

The remainder of this thesis

The remainder of this thesis uses STAMP to address the question of how the Internet arrived in this dangerous, vulnerable state and discusses potential remedies and mediations.

Chapter 2—STAMP Background and Applying STAMP to Security

This chapter introduces STAMP (Systems Theoretic Accident Modeling and Processes), a technique developed by Prof. Nancy Leveson, to model accidents. As discussed in the previous chapter, the principle goal of this thesis is to understand if STAMP can be effectively applied to security. This chapter offers a brief overview of STAMP and discusses how I apply STAMP to a security problem.

Limitations of Previous Safety Models

Historically, accident investigators have used a number of different techniques to investigate and analyze accidents. These techniques have significant flaws that may result in incomplete understanding of the accident being modeled[17]. According to Leveson, most accident models view accidents in a linear fashion, assuming that a chain of events causes the accident, usually beginning with a component failure. If we trace the chain-of-events, these models say, we will come to find the cause of the accident.

However, systems have been growing more complex and accidents do not always fit easily into chain-of-event model. Leveson writes that a number of changes have occurred to the systems we are trying to safeguard, which make the old accident models less effective: since World War II, the pace of technological change has increased, resulting in systems that rely on lesser-understood technologies; software of dizzying complexity is now routinely a part of such systems and, unlike physical components, whose failure properties can be well understood, software can contain failure modes unknown even to its developers. The relationship between humans and automations is changing too, with humans being asked to oversee the operation of ever more complicated automation.

In this new world, the chain-of-event model is not sufficient, Leveson writes. The chain-of-event model places a focus on failure events. This focus fails to sufficiently account for four types of factors that should be considered by an accident model.[18]

---

[17] Nancy, Leveson G. Safeware: System Safety and Computers. Addison-Wesley, 1995. Chapter 2.
[18] Leveson, Nancy G. "A New Accident Model for Engineering Safer Systems." Safety Science 42.2 (2004): 1-30.

System accidents

Chain-of-event models primarily focus on component failure as a cause for accidents. *System accidents* are accidents caused by the way two components interact. In such cases, both components work as designed, but their interaction causes a failure. [19]

Human Error

Human error is a "catch-all" frequently used in chain-of-event models to assign blame. In reality, what is error is much murkier. Workers rarely work exactly as they are told, instead procedures evolve over time. Therefore, at the time of the accident it is usually easy to find someone who did not follow instructions exactly and assign blame to him or her. To be complete, an accident model must include the idea that workers are going to attempt to change their work patterns to optimize their work to some local goal. The model must consider why workers were able to modify their patterns and why they made the decision to modify their work patterns as they did.[20]

Social and organizational factors

In traditional event-based models, it is difficult to accurately represent social and organizational factors. Management structure, culture, and the reasoning for making decisions can contribute to an accident, and it is therefore important to include them in the model in order to be complete.[21]

Adaptation

Older chain-of-event models of physical systems did not need to change much, as the physical systems they modeled did not change frequently. However, as the accident's context is broadened to include very dynamic factors, such as socio-technical factors, the model must contain the ability to adapt. There are a multitude of changes that can occur, including personnel changes and process changes. Furthermore, when a change occurs in one component, other affected components must be made aware and adapt in kind.[22] This information updating is complex and needs to be considered when seeking to understand an accident.

---

[19] Leveson, Safety Science.
[20] Leveson, Safety Science.
[21] Leveson, Safety Science.
[22] Leveson, Safety Science.

Using STAMP to model accidents

To address these environmental changes and the deficiencies of old accident models, Leveson developed STAMP (Systems-Theoretic Accident Model and Processes). STAMP views accidents in a much broader context than the previous chain-of-event models and includes the factors discussed above that are not handled well in event-chain models.

STAMP views safety as a control problem.[23] A system is a set of interrelated components that must be kept in equilibrium by a feedback and controllers. When the feedback loops and controllers are designed correctly and work as planned the system will be resilient to "sparks" such as external disturbances and component failures. But if the control loops are incorrectly designed or have degraded, the system will not be successful at regaining equilibrium after such a "spark" and an accident may occur.

Leveson writes:

> In this conception of safety, accidents occur when external disturbances, component failures, and/or dysfunctional interactions among system components are not adequately controlled...[24]

The "sparks" a system faces such as external disturbances and component failure are inevitabilities of operation. If the system is truly safe and a "spark" occurs, it will be resilient enough to continue safe operation. An accident occurs when the system cannot regain a safe equilibrium after such a "spark". The accident should be attributed to a safety system that was unable handle the anomalous event, not to the anomalous event.

Understanding an accident is not a matter of understanding the disturbance or component failure, rather it is a question of understanding why the system was not able to regain equilibrium after the event.

Designing a safe system is a matter of designing appropriate control structures. The designer must fully expect external disturbances and component failures and create a system that is able to effectively respond to such events. Furthermore, such a system must evolve and adapt over time.

---

[23] Leveson, Nancy G. A New Approach to System Safety Engineering. 2 May 2005 <http://sunnyday.mit.edu/book2.pdf>, Chapter 4.

[24] Leveson, Nancy G. A New Approach to System Safety Engineering, Introduction to part II

The STAMP Process

Conducting a STAMP analysis consists of three sub-analyses[25]:

1) *Static Safety Control Structure*

2) *Structural Dynamics*

3) *Behavioral Dynamics*

Together, these three analyses can give a complete picture of an accident. Briefly, the first analysis, the *Static Safety Control Structure* contains a definition of the safety system. It defines the components included in the system, the hazards threatening the system, and the required behavior of the system (e.g., the constraints it must adhere to). The second analysis, the *Structural Dynamics,* shows how the control system changed over time, focusing on what state it was in at the time of the accident. Finally, the third analysis, *Behavioral Dynamics*, addresses what forces caused the system to migrate from the original state as shown in the *Static Safety Control Structure* to the unstable state identified in the *Structural Dynamics* model.

Static Safety Control Structure

The Static Safety Control Structure shows the system as designed to address hazards. As discussed above, the safety system is divided into components, representing the key parts of the socio-technical system. The components may be physical (e.g., factories, computers) or they may be organizational (e.g., government organizations, firms). Each component in the system has a set of constraints for safe operation. In STAMP, a component's constraints are the set of rules that, if enforced, increases the resilience of the safety system. Each component has one or more controllers associated with it that, relying on feedback from the component, works to ensure the constraints are maintained.

The following diagram (Figure 3) is a sample Safety Control Structure. The lines detail the feedback channels and control mechanisms that connect components.

[25] Leveson, Nancy G. A New Approach to System Safety Engineering, Chapter 8

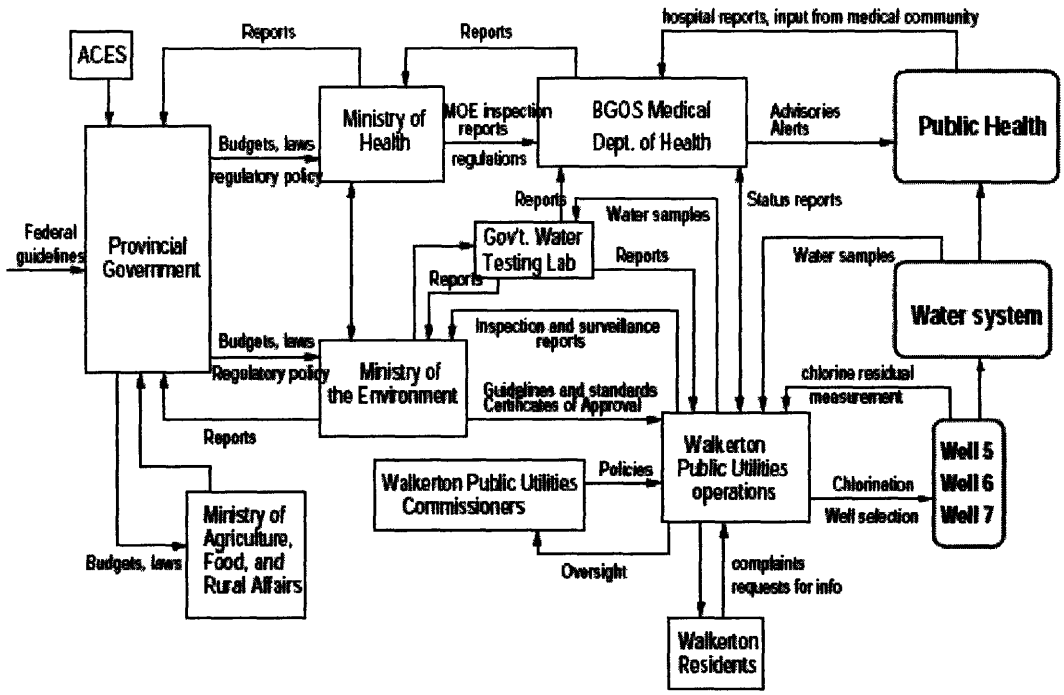


Figure 3—A sample Safety Control Structure from Leveson[26]

Structural Dynamics

The second part of a STAMP analysis is the *Structural Dynamics* analysis. This part of the analysis focuses on how the safety structure changed between its instantiation and the time of the accident. In this part of the analysis, we identify which constraints were not enforced at the time of the accident and, more importantly, identify why they were not enforced.

In order for a constraint to be effectively enforced, four conditions must be met.[27] There must be a:

- **Valid Goal Condition**—the controller must be attempting to achieve a goal
- **Adequate Control Actions**—the controller must be able to exert sufficient control onto the system component it is controlling
- **Correct Mental Model**—the controller must have a accurate understanding of the system it is controlling

[26] Leveson, Nancy G. A New Approach to System Safety Engineering, Chapter 8
[27] Leveson, Nancy G. A New Approach to System Safety Engineering, Chapter 4

- **Sufficient Feedback**—the controller must receive accurate and sufficient feedback about the state of the system

Behavioral Dynamics

The purpose of the final part of STAMP analysis is to understand the "Behavioral Dynamics" of the system. This section seeks to understand what forces drove the system from its resilient state to the state where it could not longer enforce constraints and the accident occurred.

Applying STAMP to Security

Similarities between Safety and Security

I believe that the STAMP framework can also be applied to security problems because security problems have many of the characteristics that Leveson developed the STAMP safety model to address. Both security and safety exist within socio-technical systems: systems with a complex technical problem at their core, but heavily affected by economics, regulations, and social factors. In security systems, as we saw in safety systems, the relationship between cause and effect can be non-linear, delayed, and hard to distinguish.

Security, like safety, is an emergent property. Security arises from the interactions among all the components in the system. It is not possible to determine if a system is secure (or safe) by looking at a single system component. Finally, both types of problems are highly dynamic and rely heavily on feedback.

Just as the safety system is viewed as a control system designed to keep a system in equilibrium in the face of "sparks" such as external disturbances, I argue we can similarly view the security control system as a control system designed to keep the system in equilibrium in the face of sparks. However, in the security case these sparks are motivated attacks on the security of the system rather than external disturbances or component failures.

As is the case for safety, the cause of an accident was not the external disturbance or component failure; instead it was the inability of the system to respond to that

disturbance. The same is true in this conception of security. The cause of the security incident is not the attempted attack; rather it is the inability of the system to effectively respond to the attack attempt.

We can take Leveson's statement about using STAMP for safety from above:

> In this conception of **safety**, accidents occur when external disturbances, component failures, and/or dysfunctional interactions among system components are not adequately controlled...

and modify it to pertain to security problems:

> In this conception of **security**, incidents occur when **attacks** are not adequately handled by the control system...

Differences between Safety and Security

There are important differences between safety and security; this section discusses these differences and argues that it is still reasonable to use STAMP as a means for understanding a security incident. The differences are related to the location and frequency of the disturbances the system faces.

In the case of security, the disturbances the system must respond to originate with a motivated attacker, rather than a random anomaly such as a component failure. Assuming an intelligent attacker, the "disturbance" they create will be designed to exploit the part of the system they perceive to be most likely to yield to an attack, thus the perceived weakest point of the system will be under greater pressure than a part that is perceived to be more resilient.

In addition to having targeted disturbances, the frequency of the disturbances will likely be different in the case of security. If a system is viewed as weak it will attract a greater frequency of attacks because those attacks will have a higher yield.

Although these differences are present, they still generalize to the same problem as safety: can a system respond to disturbances, regardless of their source or frequency. The STAMP model does not make any assumptions about the source of the "disturbances"; all disturbances are treated the same; the control system simply attempts to regain equilibrium after a disturbance.

In summary, I believe STAMP can be used for security as well as safety because it is concerned with a system's ability to respond to a disturbance, not the source of the disturbance.

Multiple Hierarchies in the STAMP framework

STAMP relies on the idea that components are arranged in a hierarchy; superior components are responsible for controlling the actions of subordinate components. Components rely on their superiors to provide the control actions that maintain their constraints. For example, if the superior component is a regulatory authority, it can control its subordinate with regulations or laws. If the superior component is a work supervisor, it may use the manager-employee relationship to achieve compliance with the constraint by the subordinate component.

When modeling the security of the Internet, a loosely coupled system, I observed two things. First, there are multiple control hierarchies and second, each of the multiple hierarchies has a top-level component whose behavior is not controlled by another. For instance, home computer users are not required by any other entity to use their computers in a specific way. At this point, there is not a law mandating that they use their computer in a secure manner nor do that have any relationship with their Internet Service Providers requiring them to take certain security precautions. They make decisions about their security actions and live with the consequences.

Top-level components are their own controllers, determining what constraints they should enforce on their own. Although they do not have a strict controller, they incorporate information from many other sources to determine what constraints they should enforce and how they should enforce them. I assume these top-level components act to maximize their welfare, unless otherwise regulated.

Chapter 3—STAMP Static Control Structure

Chapters 3, 4, and 5 are devoted to conducting a STAMP analysis (as defined in the last chapter) of the Internet Security System. This chapter contains the first of those three parts: the *Static Security Control Structure* (SSCS). In it, I describe the many components of the Internet's security structure, identify their relationships to one another. I identify the system hazard that the Internet Security System faces and enumerate the constraints that the components must adhere to for the system to operate securely and avoid the hazard.

The next chapter investigates the *Structural Dynamics* of the system and discusses how the security system has failed in multiple instances. For context, I use two recent significant virus outbreaks as case studies. By investigating the two Internet worm outbreaks, we can begin to understand which constraints were not able to be maintained, leading to a weakened system and ultimately to the outbreaks.

Chapter 5 contains the final step of the security STAMP analysis, the *Behavioral Dynamics*. In this chapter I discuss how and why the security system evolved in a manner that made it very difficult to maintain the necessary constraints on behavior.

The Internet Security System

The system that has developed to protect users from threats posed by malicious software is complex and broad. The distributed nature of the Internet necessitates that security is not achieved by central control; the scope, geography and underlying design of the Internet do not allow for it. Instead, the system includes many components exerting force upon one another to keep the system in dynamic equilibrium. Control may be exerted directly or by regulation and market forces.

I have identified 15 entities that interact to produce today's Internet Security System. They are organized into multiple control hierarchies, reflecting the distributed nature of the Internet. In addition to the 15 entities working to secure the system, there are two entities that attempt to disrupt Internet Security by funding or writing malicious software.

Figure 4 identifies these 17 components and organizes them according to their STAMP control hierarchies.

Each component in the diagram has a set of constraints that are to be enforced by the control actions of the components located above them. For instance, component *1. Computers* has a set of constraints which are enforced by both *3. Users* and *6. Software / Operating Systems*. In turn, there is a set of constraints imposed on *6. Software / Operating Systems* that are enforced by *7. Software / Operating System Vendors.* The constraints are derived by considering what must be done in order to avoid the overarching system hazard, loss of utility of the Internet due to the spread of malicious software. Top-level components, such as *ISPs* or *Computer Users* enforce constraints upon themselves.

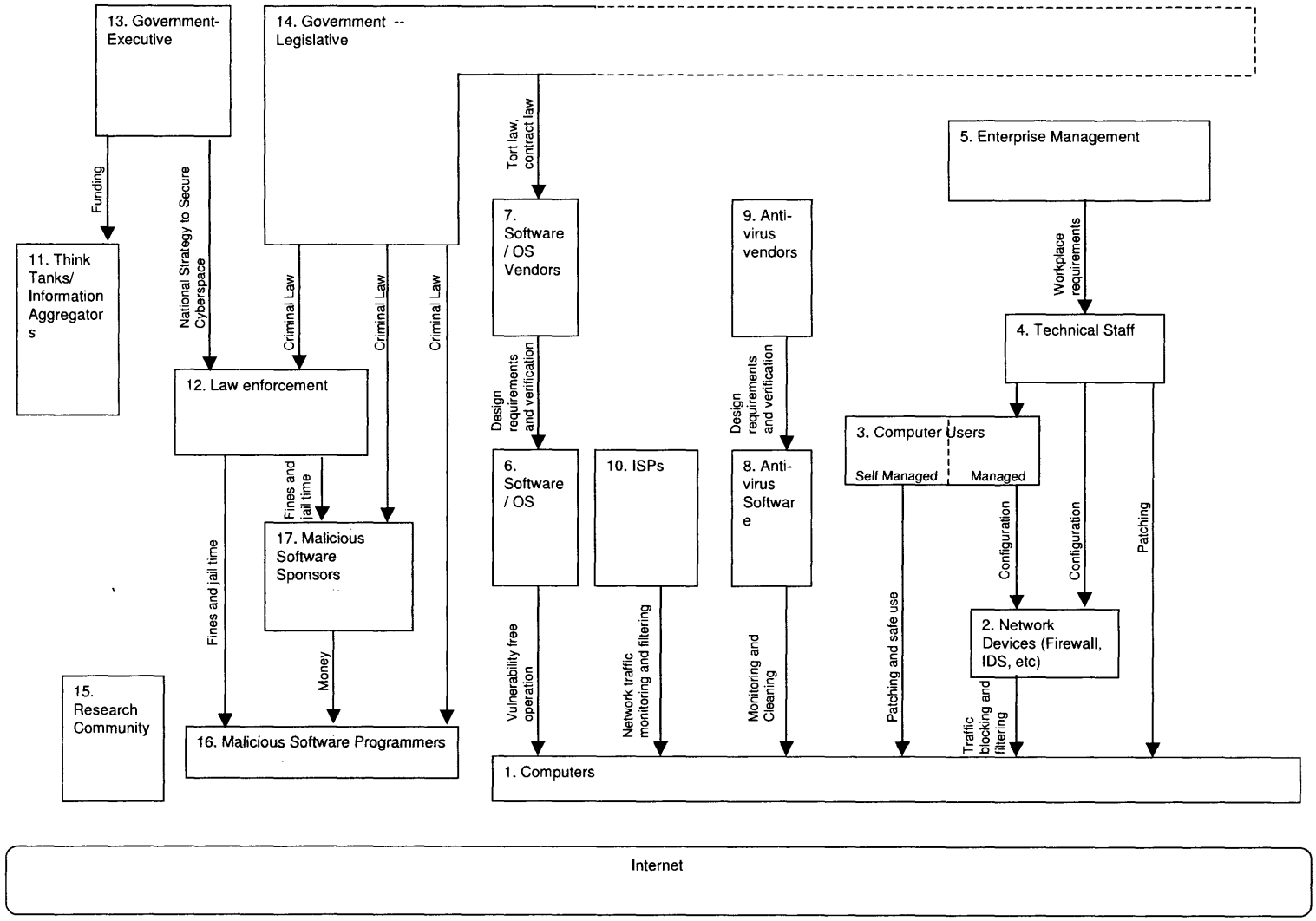


Figure 4—The Internet Security System

The remainder of this chapter discusses the entities in the figure and explains the roles they play in securing the Internet. In keeping with the STAMP framework, each section concludes with a set of constraints that the parent components must enforce to ensure secure operations and enumerates the components responsible for ensuring those constraints are enforced. The next chapter looks at two virus outbreaks that occurred and discusses which constraints were violated when the outbreak occurred.

1. Computers

There are hundreds of millions of computers connected to the Internet,[28] over 90% of which run versions of Microsoft Windows.[29] (See Figure 5) Individuals, firms, schools, and other organizations own these computers but regardless of their purpose, they are targets for malicious software.

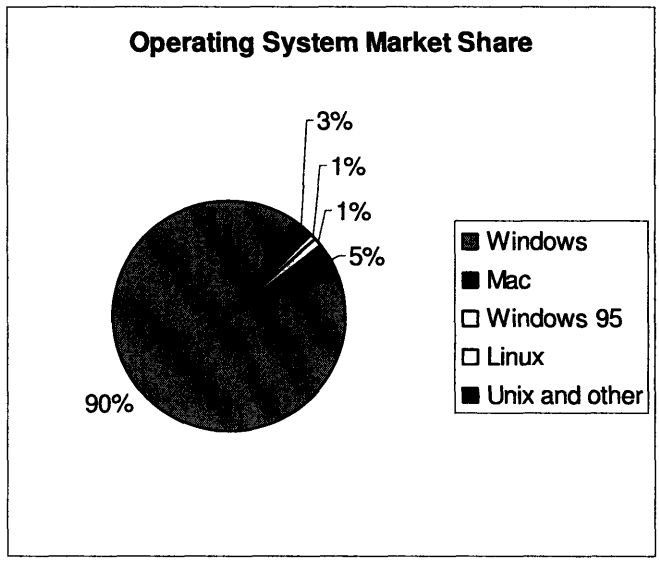


Figure 5--Operating System Marketshare (June 2004)

Computers are the last line of protection against the malicious software that threatens the Internet because the goal of malware is to compromise these computers. In order to maintain the security of the Internet, computers must not run software that is malicious nor may they help to spread malicious software once infected.

[28] CIA World Factbook. 21 Apr. 2005. Central Intelligence Agency. <http://www.cia.gov/cia/publications/factbook/rankorder/2153rank.html>.
[29] Google Zeitgeist. <http://www.google.com/press/zeitgeist/jun04_pie.gif >.

This is a difficult task, as most users do not fully understand what the software they are running does. Even a knowledgeable user cannot know with certainty what the software he executes is going to do. There are many other ISS components that must try to ensure computers do not run malicious software and do not help spread malicious software. These enforcing components include the users, operating systems and other software they run, technical support staff, anti-virus software and the ISPs that connect the computers to the Internet that determines if the following constraints remain enforced.

Constraint	**Enforced by**
1-1: Computers must not execute malicious software programs	• Users • Technical Staff • Software and Operating Systems • Anti-virus software
1-2: If compromised, computers must not spread virus	• Network Devices • Users • Technical Staff • Software and Operating Systems • Anti-virus software

2. Network Devices

Some of the hundreds of millions of computers connected to the Internet connect are accessible directly from the Internet. But many computers, both on corporate networks and in homes, are partially protected from Internet-based attacks by network devices such as firewalls. A firewall is only part of a defensive strategy, but it provides two critical services that can protect a computer from malicious software. First, a firewall can stop certain types of network traffic that may carry malicious software, if properly configured. Second, many firewalls change a computer's network address (IP address) so it cannot be seen by other computers unless it initiates contact first. This is known as network address translation (NAT) and can help insulate a computer from self-propagating malware.

Constraint	**Enforced by**
2-1: Devices must be properly configured and maintained to prevent malicious network traffic	• Users • Technical Staff

from accessing computers	

3. Computer Users

It is useful to break computer users into two types: those that manage their own computer and those that rely on someone else such as office technical support to manage their computer for them.

As described in chapter 1, there are two ways that a computer can become infected with malicious software. The first is when the user executes a malicious piece of software such as an email attachment. The second type of infection occurs when a computer is running software with a vulnerability that can be exploited by the malicious software. In this second case, the malicious software takes advantage of a security flaw and installs itself on the computer.

Generally, the vendor that produced that insecure software releases a security update known as a patch shortly after they become aware of the vulnerability. Installing the patch can protect the computer from strains of malware that exploit that vulnerability. It is the responsibility of the computer user or the organization managing the computer to install the security patch in a timely manner.

The diagram shows that self managed users are not bound by any laws or regulations to install security patches on their computers. The user needs to enforce constraints 1-1 and 1-2 (e.g., by applying security patches, by not opening unknown attachments), but no other component works to ensure that self managed users fulfill this responsibility. In cases such as this, it is up to the individual user to decide how much energy and expense to expend in order to maintain these security constraints.

Constraint	**Enforced by**
3-1: Users must invest time and effort in securing their computers	• Users (Self enforced for unmanaged users)

4. Technical Staff

As discussed above, there is a significant portion of computer users who rely on a technical staff to administer their computers, generally in a corporate environment. This

frequently means the burden of applying software patches is shifted from the users to a technical staff overseeing many computers.

The technical staff has a difficult job enforcing constraints 1-1 and 1-2 on the computers and needs to balance proactive work (such as developing response plans and upgrading users to new versions of software) with a significant amount of reactive work.

Constraint	**Enforced by**
4-1: Have sufficient time and training to effectively complete work tasks	• Technical Management

5. Technical Management

The layers of management overseeing the Technical Staff are responsible for ensuring that the Technical Staff component maintains their constraints in addition to making strategic security decisions. They need to find a level of security that is sufficient but will minimize costs and allow them to complete their business mission. Because security is a non-event, finding this balance is difficult.

Constraint	**Enforced by**
5-1: Must correctly balance security risk with performance pressure	• Technical Management (Self Enforced)

6. Software / Operating Systems

As discussed above, malicious software can spread in multiple ways, including email and self-propagation. For malware to propagate without any user intervention, it must exploit vulnerabilities in either a computer's operating system or in an application running on the operating system.

If an author is writing malicious software for financial gain (by creating botnets of compromised computers), the money he or she can make is proportional to how many computers he infects. A rational virus writer will target operating systems and software with a high market share.

Constraint	**Enforced by**
6-1: Software must not contain security	• Software and Operating

vulnerabilities	System Vendors

7. Software Vendors

As I discuss in chapter 5, creating software is a hard problem, and vulnerabilities exist in most, if not all, software packages. For example, SecurityFocus's Bugtraq database, known in the industry as the de facto clearinghouse for security vulnerabilities, recorded an average of 55 new vulnerabilities per week in 2004.[30]

Most major software companies seem to be working hard to reduce vulnerabilities, but this focus was not always present. In 2002, Microsoft recognized that the connected nature of the Internet was making security even more important and launched their *Trustworthy Computing Initiative,* revisiting their development tools and processes.

Even with increased attention given to writing secure software, software will have vulnerabilities and software companies must have a process for notifying customers about the vulnerabilities and delivering updated software to them. Companies face a challenge: customers demand notification of vulnerabilities in a timely manner, but notifying also alerts malicious software authors about potential exploits.

Constraint	**Enforced by**
7-1: Vendors must successfully balance pressure to run a profitable business and create secure software	• Software and Operating System Vendors (Self Enforced)

8. Anti-virus and other Prophylactic Software

The market for antivirus software is large and growing. According to IDC, the size of the market was about $4.2 in 2004,[31] up from $2.2 billion in 2002.[32] (Other prophylactic software such as anti-spyware and software firewalls are not included in these numbers.)

[30] Symantec Internet Threat Report: Trends for July 04 - December 04, p34.
[31] IDC Study referenced in Symantec Press Announcement <http://www.symantec.com/press/2004/n040921a.html>.
[32] IDC Research announcement <http://www.idc.com/austria/downloads/about/featured_research_03_1.pdf>.

This kind of software fights with malicious software, attempting to prevent its spread as well as remove infections that have already occurred. At the same time, many forms of malicious software attempt to disable prophylactic software.

Most anti-virus software works by searching for identifying characteristics of known malicious software. Once it identifies a piece of malware by these characteristics (also known as a virus signature) it attempts to remove it. As new malware is discovered, it is added to the set of known signatures and distributed to customers, generally via some automatic updating process.

Constraint	**Enforced by**
8-1: Antivirus Software must not be disabled by malicious software	• Anti-virus and other Prophylactic Software Vendors
8-2: Antivirus Software must detect and remove malicious software as well as prevent malware from installing itself on an otherwise vulnerable computer	• Anti-virus and other Prophylactic Software Vendors

9. Anti-virus and other Prophylactic Software (PS) Vendors

It is the role of the anti-virus and other Prophylactic Vendors to create software that can find and remove malicious software. It must be robust enough to withstand the malicious software's creative attempts to disable it.

In addition creating this software, the PS vendors must also identify new malicious software threats as soon as possible. Once identified, they must capture and dissect the threat, create a new signature for it, test it, and release it to the public. Any delay could result in more widespread infections.

Constraint	**Enforced by**
9-1: Antivirus Software Vendor must provide up to date virus definitions as quickly as possible and automatically deliver them to customers or notify customers about their availability	• Anti-virus vendors (self-enfoced)

10. ISPs[33]

The ISPs play an interesting role in the security system—they offer connectivity to the computers connected to the Internet. ISPs have customers of all sizes. Large enterprises and institutions rely on them to connect their networks to the Internet just as home users and small businesses use ISPs to connect to the Internet. This puts the ISPs in an important position as gatekeeper to the Internet. They have the ability to monitor and control the traffic originating from and destined for their clients.

Monitoring the network can be costly to the ISP on a number of dimensions. If an ISP chooses to block certain types of Internet traffic, they risk interrupting some legitimate services of their customers. Some smaller, consumer focused ISPs chose to filter their customer's traffic and attempt to filter out malicious network traffic, but larger ISPs like AT&T[34] and Verizon do not, believing it is the customer's responsibility to protect against malicious network traffic.[35, ,36]

Constraint	**Enforced by**
10-1: ISPs must exert some degree of control over the computers that they connect to the Internet while keeping customers satisfied	• Internet Service Providers (Self enforced)

11. Think Tanks/Information Aggregators

These organizations, which include the National Cyber Security Partnership, the Internet Storm Center, and Carnegie Mellon's CERT/CC, provide a number of services to the Internet community. Some act as clearinghouses for member organizations to share threat and attack data in the hopes of viewing the "bigger picture". Others offer prevention and response best practices and others, like the National Cyber Security Partnership focus on user community awareness.[37]

---

[33] Schneier, Bruce. Schneier on Security. 2 Dec. 2004. 2 May 2005 <http://www.schneier.com/blog/archives/2004/12/striking_back_a.html>.
[34] However, AT&T does do some spam filtering
[35] Telephone interview with Verizon Sales. 18 Apr. 2005.
[36] Telephone interview with AT&T Sales, 19 Apr. 2005.
[37] CERT was created in 1988 with funding from DARPA as a response to the Morris worm

Constraint	Enforced by
11-1: Information aggregators must have access to information about Internet threat activity	• Government-Executive • Think Tanks / Information Aggregators (Self Enforced)
11-2: Must recognize and communicate dangerous trends	• Government-Executive • Think Tanks / Information Aggregators (Self Enforced)

12. Law Enforcement

Law enforcement agencies, including the FBI and local police departments, must be skillful enough to apprehend the authors of malicious software that leads to outbreaks. Their task is compounded by the international nature of the Internet. Law enforcement is bound by jurisdiction, and, without international agreement, their ability to catch those involved with malicious software is limited.

In addition to apprehending hackers, law enforcement can play a preventative role as well. If law enforcement agencies are widely known to be effective at apprehending malware creators, some would-be malicious software authors will be deterred.

Constraint	Enforced by
12-1: Must have a impressive and well known record for arresting those responsible for virus outbreaks	• Law Enforcement • Government—Legislative

13. Government—Executive

In 2002, the president released the *National Strategy to Secure Cyberspace.* In it, the executive lays out the US government's cyber security objectives: the government must work to prevent attacks and if attacks can't be prevented, then it must work to minimize the damage of the attacks.

The strategy states that the most of the responsibility for securing cyberspace rests on the private sector, but the government will play a role in ways the private sector cannot or will not.[38]

The National Cyber Security Division (NCSD) of the Department of Homeland Security was created in June of 2003 to address some of these challenges. The purpose

[38] United States. The White House. The National Strategy to Secure Cyberspace. Feb. 2003. <http://www.whitehouse.gov/pcipb/cyberspace_strategy.pdf>.

of the NCSD is to conduct cybersecurity analyses, issue alerts and warnings about threats, improve information sharing, respond to major incidents, and aid in national-level recovery efforts.[39] The NCSD works closely with the information aggregation community, providing the funding to US-CERT. NCSD also works with law enforcement.

Constraint	Enforced by
13-1: Must prevent cyber attacks	• Government—Executive (Self enforcing)
13-2: Must minimize the damage of cyber attacks	• Government—Executive (Self enforcing)

14. Government—Legislative

Statutes created by the Federal and State legislative branches of US governments play an important part in the Internet Security System. There are a set of criminal laws that are meant to govern those who would create the malicious software or use botnets for financial gain. Additionally, there are contract and tort laws that proscribe rules designed to guide the behavior of computer software vendors. The US does not have laws that bind computer users or firms to maintain their computers in certain ways.

Criminal laws aimed at malicious software authors

The dual purpose of the US and foreign criminal laws are to both punish those responsible for violating someone's computer and to deter others from doing the same. There are federal and state laws that criminalize hacking as well as distributing malicious software such as viruses on computers within state or federal jurisdiction.[40] In 2001, the Patriot Act amended to the statutes to clarify that the US could also prosecute criminal conduct that initiated outside of the US.

The punishment varies by the law that was violated, but can be significant, including fines, prison, or both. The statute allows for prison terms for up to 20 years in some cases. In extreme cases involving government computers, the prison term can be for life.

---

[39] United States. Department of Homeland Security. Ridge Creates New Division to Combat Cyber Threats . 6 June 2003. <http://www.dhs.gov/dhspublic/display?content=916>.

[40] US Cybercrime Law: Defining Offenses Susan Brenner (Information Systems Frontiers) 6:2

Laws affecting software producers—Tort Law and Contract Law

In the US, there are two legal pathways that are designed to hold software producers accountable for harm related to insecure software they have unintentionally created.[41] These are tort law and contracts law. Tort law has a dual purpose. It is designed to compensate a damaged party for harm incurred and also to deter companies from creating products that can cause harm.

Using torts in the software security context has an added layer of complexity to it. In the case of malicious software, the party most responsible for the damage is the hacker that wrote and released the software. While these crimes are covered under the criminal code, law enforcement has not been successful at apprehending those responsible for viruses and worms.[42] In the case where the optimal target is difficult to obtain judgment against, it is efficient to go after the second best solution, write Landes and Posner in The Economic Structure of Tort Law. In this case, the next best target to apply tort liability to is the software vendor.[43]

Within tort law there are two areas that most frequently apply to computer software: strict product liability and negligence. Strict product liability states that a company is responsible for personal injury or property damage caused by its product, regardless of contractual disclaimers or warranties. In order for strict product liability to apply, the injured party must show that they suffered personal injury or property damage and that the product was defective.[44]

Strict product liability does not include liability for economic damage, which is a limitation that is especially relevant to computer software. At this time, computer data is

---

[41] Other criminal and tort pathways exist for prosecuting cases where software producers intentionally release insecure products.

[42] In 2002, the Justice Department convicted just 75 people for all computer crimes, not only virus related crimes. (Fryer, Alex. "Microsoft offers bounty for arrest, conviction of cybercriminals." Seattle Times 6 Nov 2003.)

[43] Pinkney, Kevin. "Putting Blame Where Blame is Due: Software Manufacturer and Customer Liability for Security Related Software Failure." Albany Law Journal of Science and Technology 13 Alb. L.J. Sci. & Tech. 43 (2002).

[44] Armour, Jody, and Watts Humphrey. "Software Product Liability." Software Engineering Institute, Carnegie Mellon Institute (1993).

not considered property so any malware that destroys a company's data by exploiting a software flaw would not qualify the software vendor for strict product liability.[45]

Negligence is another legal avenue that provides an incentive for software companies to improve the security quality of their product. A consumer can charge negligence if he feels that the damage he incurred came as the result of a lack of "reasonable" quality control processes during the design of the software.[46] In a software context, this could mean releasing software that is known to have defects or employing quality assurance processes insufficient for finding defects. In my research, I did not find any examples of lawsuits where a vendor was successfully sued for negligence because of security flaws in their software.

While tort law holds software producers responsible for only injury or property damage, producers may be responsible for economic damage caused by their software under contract law. This contract between the vendor and consumer is codified in the end-user licensing agreement ("EULA") that the user must agree to before installing the software.

A EULA makes very limited warrantees and generally forces the consumer to agree that the vendor is not liable for any types of economic damage the results from the software's use. Practically, only the largest software purchasers are able to negotiate terms in the EULA. Most consumers are bound to the standard EULA, which is typically very favorable to the software vendor.[47]

Constraint	**Enforced by**
14-1: Must craft enforceable laws that have a positive effect on the security of the Internet	• Government—Legislatve (Self enforcing)

[45] Barnes, Douglas A. "Deworming the Internet." Texas Law Review 83 (279-329). <http://ssrn.com/abstract=622364>.
[46] Miyaki, Patrick. "Computer Software Defects: Should Computer Software Manufacturers Be Held Strictly Liable for Computer Software Defects?" Santa Clara Computer and High Technology Law Journal May 1992: 121-144.
[47] Wildstom, Stephen. "Want to sue over buggy code? Forget it, Microsoft and other software makers shield themselves with the End User Agreement." BusinessWeek 22 Sept 2003.

15. Computer Science Security Research Community

The research community is closely tied to the Internet security community. Worthwhile ideas and discoveries made by the research communities should be transferred to the private sector.

Constraint	**Enforced by**
15-1: Must develop new technologies that can be used to control malicious software	• Computer Science Security Research Community (Self enforcing)

The Internet Security ecosystem also includes the actors that attempt to take advantage of weaknesses in the system to profit or to cause damage. In this analysis, these actors include those that write and release the malicious software as well as those that use computers compromised by the malicious software to earn money. In Figure 3, they are represented as two separate components: the Malicious Software Writers and Malicious Software Sponsors.

16. Malicious Software Writers[48]

The motivations of malicious software programmers were discussed in chapter 1. Their motivations have varied, but they are increasingly financially motivated.

Constraint	**Enforced by**
16-1: Malicious Software Writers must not create and release malicious software	• Law Enforcement • Government—Legislative

[48] Malicious software authors are often referred to as hackers. The term hacker has a rich history and a very broad meaning. Most generally, the term hacker is used to describe one with a curiosity for learning how something works. ("The Hacker Ethic", Ethics in the Computer Age Proceedings, ACM Press, New York, New York, November 1994.) For many, the term hacker is a compliment. However, in recent years the media and others have co-opted the term and used it to refer to one who uses computers to break the law. While not historically correct, this is the definition of hacker most widely understood and so, for the sake of simplicity, I use the term hacker to refer to refer to the authors of malicious software.

17. Malicious software sponsors

The Internet Security ecosystem also includes the actors that take advantage of weaknesses in the system to make a profit or to cause damage. As discussed earlier, these bad actors can use compromised computers to accomplish their goals. They are typically distinct from the writers of the malicious software; instead they either contract with the programmers to write the virus or later "lease" computing power for a hacker's stable of hijacked computers.

Chapter 1 describes the some of the ways sponsors could use malicious software to make money. They include:

- Sending SPAM through compromised computers
- Blackmailing others with the threat of a Distributed Denial of Service attack using compromised computers
- Stealing sensitive financial and personal information from compromised computers
- Using compromised computers to send emails and host websites tricking recipients into sharing financial
- Fraudulently clicking on online advertisements to raise revenue or deplete competitor's advertising budgets
- Launching DDoS or other types of attacks to commit terrorism

Constraint	**Enforced by**
17-1: Sponsors must not pay hackers to write malware or to gain the use of their compromised computers (i.e., botnets)	• Law Enforcement • Government—Legislative

Chapter 4—STAMP Structural Dynamics

The purpose of the previous chapter was to explain the components of the Internet Security System (ISS) and describe how they interact. When functioning properly, the ISS can make the Internet more resilient to an outbreak of malicious software.

This chapter is devoted to understanding in what ways the ISS fails, allowing an outbreak of malicious software. First, I discuss two well-known outbreaks of malicious software, the Blaster and Sobig worms. Then, I parse the details of the outbreaks and discuss which constraints were not maintained and which components were not able to maintain them. A complete description of constraint failures can be found in the table in Appendix 2.

Blaster Background

On August 11, 2003 the first variant of the Blaster worm (Blaster.A) appeared. This piece of malicious software spread vigorously and effectively, ultimately infecting at least 8 million computers and as many as 9.5 million.[49] Blaster spread without requiring any interaction on the part of users and thus infected computers at a much faster rate than malware that requires a user to open an attachment in their email. There were six variants of Blaster; each compromised computers that would later be used for the purpose of launching distributed denial of service (DDoS) attacks, however some variants installed additional software that would further cede control of the computer to the attacker.[50]

The apparent purpose of Blaster.A was to launch a large-scale DDoS attack on Microsoft's WindowsUpdate website[51] on August 6, 2003. Microsoft was able to defuse the DDoS attack with some clever website renaming, however much more damage was caused by the worm's side effects.

A computer infected with Blaster or its variants will frequently restart. It also sends out a large number of messages onto the Internet, looking for other computers to

---

[49] Lemos, Robert. "Alarm growing over bot software." CNET News.com 30 Apr. 2004. <http://news.com.com/Alarm+growing+over+bot+software/2100-7349_3-5202236.html>.
[50] Symantec Security Response W32.Blaster.C.Worm. 27 July 2004. Symantec Corporation. <http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.c.worm.html>.
[51] http://www.windowsupdate.com

infect. When the number of Blaster infected computers was at its peak, a computer would receive a network message containing an infection within one second of connecting to the Internet.[52]

It is difficult to estimate the total damage caused by the variants of Blaster, but it is believed to have resulted in productivity losses and lost revenue surpassing at least $500 million.[53] It caused a large enough disturbance to initially be blamed for the East Coast power outage on August 14, although it was later shown that Blaster was not responsible for the outage but did hamper efforts to restore power.

Within days of Blaster's discovery, a number of variants of the initial worm were also released and discovered.[54] There were 8 variants of Blaster released,[55] the first and second only two days after the initial Blaster worm was discovered. All variants of Blaster.A use the same exploit, a vulnerability in Microsoft Windows for which a patch had been offered nearly a month before worm was released. The most notable difference among the variants of Blaster was the date and target of the DDoS attack.

The original author of Blaster has not been found, but authorities did arrest Jeffrey Lee Parson, a 18-year old from Hopkins Minnesota, for modifying and releasing a variant of Blaster.A called Blaster.B, which infected approximately 7,000 computers, just a small fraction of the 9.5 million total infections.

The mechanism by which Blaster spread was a software error in the code for Microsoft Windows's RPC implementation. Computers use a protocol called RPC (Remote Procedure Call) to allow one computer to run a subroutine on another. For example, a computer running an email client might send an RPC message to an email server requesting any new email messages. The use of RPC is very common and occurs not only in computers running the Windows operating systems, but others such as Unix and Linux computers.

Microsoft announced this vulnerability and its corresponding patch on July 16, 2003 in Security Bulletin MS03-026. According to the bulletin an attacker could send a

---

[52] Lemos, Robert. "MSBlast epidemic far larger than believed." CNET news.com 2 Apr. 2004. <http://news.com.com/2100-7349_3-5184439.html>.
[53] Morrison, Jim. "Blaster Revisited." ACM Queue 2 (2004): 34-44.
[54] A variant of a worm is a new strain of the malware, built by modifying the original worm's code.
[55] Symantec Security Response. Symantec Corporation. <http://securityresponse.symantec.com/avcenter/venc/auto/index/indexW.html>.

specially crafted RPC message and gain control of a user's computer. Users were urged to install the patch as soon as possible.

As is often the case, developers of malicious software used the information provided by Microsoft to create an exploit for the vulnerability outlined in the Security Bulletin. The exploit allowed a hacker to send a message to a computer and—if the receiving computer is still unpatched and therefore vulnerable—the hacker is rewarded with control of the attacked computer.

As the infection spread, so did the problems Blaster.A caused. Aside from the DDoS attack the virus was programmed to execute on August 16, 2003, it was not an otherwise malicious worm. That is, it did not contain instructions to delete or steal personal data nor did it allow a computer to be remotely controlled for purposes other than the planned DDoS.[56] It was, however, a poorly coded program and frequently caused computers to slow or frequently restart.[57]

Microsoft was able to avoid the DDoS attack that was encoded in the original version of Blaster. The variants did not target major websites for their DDoS attacks and there was not much coverage in the media detailing the outcome of these DDoS, making it likely that a sponsor paid the variant writers to attack websites. However, the unintended consequences of Blaster still extracted a very high cost. In the months after its release, various think tanks estimated that Blaster infected between 300,000 and 1,000,000 computers. However, more recently Microsoft revise their estimate to 9.5 million infections, based on the number of Blaster removal kits that were downloaded from their website.[58,59]

---

[56] Other variants of Blaster, however, included a "backdoor", allowing compromised computers to be fully hijacked and made part of botnets.

[57] It is not known for sure, but these do not seem like intentional elements of the worm because they serve to alert users of the infection. Users who have been alerted are more likely to remove the worm from their computer thus reducing the number of computers that will take part in the DDoS attack.

[58] Lemos, Robert. "MSBlast epidemic far larger than believed." CNET news.com 2 Apr. 2004. <http://news.com.com/2100-7349_3-5184439.html>.

[59] Lemos, Robert. "Worm worries grow with release of Windows hacks." CNET News.com 28 Apr. 2004. <http://news.com.com/Worm+worries+grow+with+release+of+Windows+hacks/2100-1002_3-5201807.html>.

Sobig Background

Another interesting case to examine is the spread of the Sobig family of worms. Similar to Blaster in its magnitude, Sobig was quite different across a number of operational dimensions. Sobig spread via a different mechanism and was motivated by a different purpose. At a systems level, however, Sobig and Blaster have many similarities, sharing some of the failures that allowed them to spread so widely.

The first of the six Sobig viruses, *Sobig.A,* was discovered January 9, 2003.[60] Like its five successors, it was a mass-mailing worm that is widely assumed to have been created by spammers to facilitate sending spam.[61] Versions of Sobig progressed in sophistication until the final version, *Sobig.F,* was released on August 18, 2003, just two weeks after Blaster.[62]

The succeeding 5 versions of Sobig, *Sobig.B* though *Sobig.F* were similar to the first version but included revisions that fixed earlier bugs, making the Sobig virus more effective.

As opposed to Blaster, which spread from computer to computer without human intervention by exploiting a flaw in Microsoft's RPC implementation, Sobig is a mass-mailing worm, that spreads by tricking users to open an email attachment. The Sobig lifecycle can be divided into two phases. The first phase, **propagation** begins when a user opens an infected email attachment. Upon opening it, his or her computer becomes infected and then sets about infecting other computers as well. In order to infect other computers, the newly infected computer sends out emails, each containing an attachment that, if executed, will infect the recipient's computer. Sobig scours the newly infected computer's hard drive looking for new addresses to send the emails to. The reader has no doubt received many Sobig messages, which frequently contain the subject "Re: Details". A sample email is below:

---

[60] Symantec Security Response W32.Sobig.A@mm. 9 Jan 2003. Symantec Corporation. <http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.a@mm.html>.
[61] Lemos, Robert. "Sobig spawns a recipe for secret spam." CNET news.com 25 June 2003. <http://news.com.com/Sobig+spawns+a+recipe+for+secret+spam/2100-1002_3-1020963.html>.
[62] Symantec Security Response W32.Sobig.f@mm. 18 Aug 2003. Symantec Corporation. <http://securityresponse.symantec.com/avcenter/venc/data/w32.sobig.f@mm.html>.

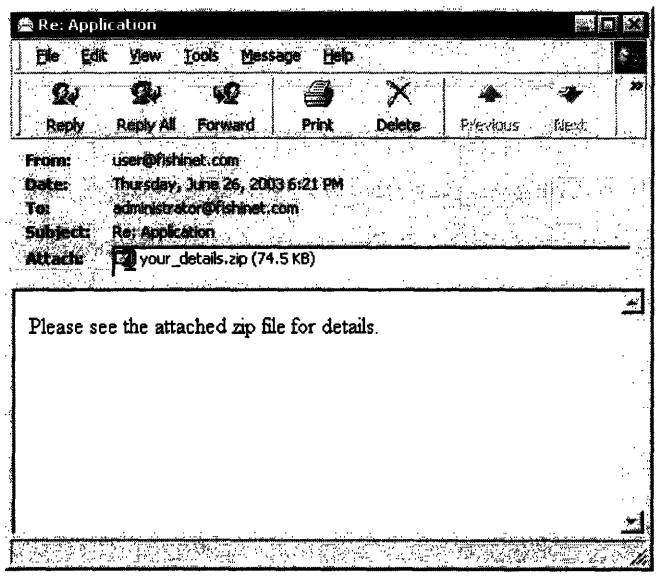


Figure 6—A sample Sobig Email[63]

After propagating, the Sobig worm enters its second phase. It downloads and installs a software package from an Internet server.

The first five versions of Sobig instructed the compromised computer to download a file known as a trojan, which in turn downloaded and installed a proxy server. This proxy server allows someone, presumably the author of Sobig or his sponsor, to send email anonymously via the compromised computers rather than directly from his own computer. This is especially useful for sending spam emails.

The final version of Sobig, *Sobig.F,* was highly virulent. During the propagation phase, millions of computers were infected; at its peak 1 in 17 email messages on the Internet were carrying the Sobig virus.[64] In the case of Sobig.F, however, authorities were able to shut down the servers containing the payloads. It is assumed that Sobig.F would have also downloaded a trojan and then used the trojan to install the proxy server, but that is not clear. Perhaps the trojan would have been used for other insidious purposes such as launching DDoS attacks or click fraud.

The extent of the damage caused by Blaster and Sobig is hard to determine; Symantec estimates that together they caused a loss of upwards of $2 billion in lost

[63] Trend Micro Website. <http://www.trendmicro.com/vinfo/images/worm_sobig_e_img1.gif>.
[64] Becker, David. "MyDoom virus declared worst ever." CNET News.com 29 Jan. 2004. <http://news.com.com/2100-7349_3-5149764.html>.

productivity.[65] Microsoft has offered $250,000 reward for any information leading to the Sobig or Blaster authors, but neither has been caught.

Learning from Blaster and Sobig

In the previous section about the damage caused by Sobig and Blaster, I showed that although they spread in different manners, they have similarities at higher levels. For instance, both allowed the malware developer to install arbitrary software on a compromised computer.

The next two sections use STAMP to understand *why* Sobig and Blaster spread as successfully as they did. First, a brief discussion of how the failed component interactions that led to the spread of the malware are classified in STAMP. The following section discusses which constraints I found were violated in my investigation of Blaster and Sobig.

Understanding why Constraints Were Not Enforced

In a STAMP security environment such as the one described by Figure 4, the constraints placed on a component are expected to be enforced by components that are higher in the hierarchy. In the diagram, a box placed above another component and connected by a downward arrow represents a controlling component. For example, the constraints on component 8, anti-virus software, are enforced by component 9, the anti-virus vendors. In other words, it is the responsibility of component 9 to ensure that the anti-virus software they create adheres to constraints 8-2: Antivirus Software detect and remove malicious software and 8-1: Antivirus Software must not be disabled by malicious software.

In STAMP, there are four reasons why a component might not be effective at exerting control on its subordinate component, thus allowing its constraints to become invalid.[66]

1. **Invalid goal condition**—the controlling component is attempting to enforce the constraint by targeting a goal

---

[65] Symantec Internet Threat Report: Trends for July 04 - December 04, p34.
[66] Leveson, Nancy G. A New Approach to System Safety Engineering, Chapter 4.

2. **Invalid action condition**—the actions of the controlling component are not effective at changing the state of the component they are controlling
3. **Invalid model condition**—the controlling component's mental model of the component being controlled is inaccurate and results in ineffective control
4. **Invalid feedback condition**—the controlling component does not receive the feedback it needs to effectively control the subordinate component.

My STAMP analysis attributes each violated constraint to one or more of the above conditions.

Additionally, there are a number of cases where a component has constraints on its behavior but it does not have a superior component that controls its behavior. For example, component 3, the user component, has a number of constraints on its behavior but it does not have any components responsible for controlling its behavior. At this point, there is not a law requiring that users take certain actions to secure their computers, nor are there governing bodies who require that computers are well maintained before they are allowed to connect to the Internet.

Each user, however, has his or her own welfare at heart. In cases such as these, the security system relies on the users' calculations of their best interest. If they feel that it is in their interest to maintain their computers, they will invest the time, effort, and money required to ensure their computers are secure.

Where the Internet Security System Failed

The Internet security system was operating with many constraints violated. Appendix 2 details which constraints were violated and contributed to Blaster and Sobig's success. Some of these constraints had never been enforced while others had slowly stopped being enforced. Because the constraints had been violated, the system was unable to repel multiple types of worm attacks—a mass mailing worm and a network propagating worm.

The full analysis in Appendix 2 identifies which constraints were violated and hindered the Internet security system's effective response. The analysis also addresses which components failed in their responsibility to maintain the constraints put upon a subordinate component.

Of the violated constraints in the analysis, I believe the following three played the most direct roles in the Blaster and Sobig outbreaks.

- 1-1. Computers must not execute malicious software
- 6-1. Software must not contain security vulnerabilities
- 16-1. Virus programmers must not create and release malicious software

Constraint 1-1: Computers must not execute malicious software

This constraint was clearly violated at the time of the Blaster and Sobig outbreaks. To learn from it, we look at why the constraint was not enforced. The Internet Security System relies on four other components to enforce this constraint: Users, Technical Staff, Software / Operating Systems, & Anti-virus software.

First, users did not effectively enforce this constraint on their computers. Sobig spread rapidly because so many naïve users double-clicked on an attachment in their email. This constitutes a failure in the **model condition**. The users did not sufficiently understand the working of the system they were using.

The Blaster worm spread by exploiting a security vulnerability in Microsoft Windows. Microsoft had published a security patch for this vulnerability almost 2 months earlier[67] but the millions of computers infected by Blaster had not installed it. This was a failure on the part of the users who maintain their own computers and the Technical Staff who maintain corporate networks. In this case, the unenforced constraint is due to an invalid **goal condition** or **model condition**. Users either did not intend to keep their computers updated or they did not realize that their computer needed to be patched in order to remove the vulnerability.

In both cases the users did not invest the time to secure their computers or obtain the knowledge necessary to operate the computers securely. For computers that were managed by a company's technical staff, the staff is responsible for not applying patches as they are released, but they did not. Different firms have different reasons for the lapse in patching, but it could be due to invalid **feedback conditions** or invalid **action conditions.**

[67] Microsoft Corporation. <http://www.microsoft.com/technet/security/bulletin/MS03-026.mspx>.

The operating system running on the infected computer was also at fault for not enforcing this constraint in the case of Blaster. Although there was a security patch available, the existence of the vulnerability allowed computers running Microsoft Windows to execute the malicious Blaster software. This is an invalid **action condition.**

Anti-virus software was able to help enforce this constraint in many cases, but not in enough to stem the outbreaks. Major anti-virus vendors responded within 1 day[68, 69] with updated virus definitions. Those computers running anti-virus software with updated definitions were immune from Sobig and Blaster. However, those without virus protection or without updated virus definitions were not.

The next chapter delves deeper into the forces that prevent computer users and firms from sufficiently protecting their computers.

Constraint 6-1: Software must not contain security vulnerabilities

The violation of this constraint contributed to the spread of Blaster, but not to Sobig. Within the Internet Security System, it is the responsibility of the Software Vendor component to ensure that the software adheres to this constraint. In the case of Blaster, Microsoft failed to ensure that its software was vulnerability free.

This is both a failed **action condition** and failed **feedback condition.** Although their goal seems to be to make secure software, especially in light of the 2002 *Trustworthy Computing Initiative* and the ensuing focus on computer security, Microsoft failed in two ways. First, the process for creating the software resulted in defects in the software and secondly, there was insufficient feedback evidenced by the fact that the vulnerability was only found after the software was released.

The next chapter discusses the inherent difficulties of developing large, complex software.

Constraint 16-1. Virus programmers must not create and release malicious software

[68] McAfee Virus Information. McAfee. <http://us.mcafee.com/virusInfo/>.
[69] Symantec Security Response. Symantec Corporation. <http://securityresponse.symantec.com>.

The number of viruses released each year is increasing dramatically (Figure 7), meanwhile there have only been a few significant arrests. Notably, the authors of Blaster and Sobig, as well as a number of other well-known viruses are still yet to be caught.

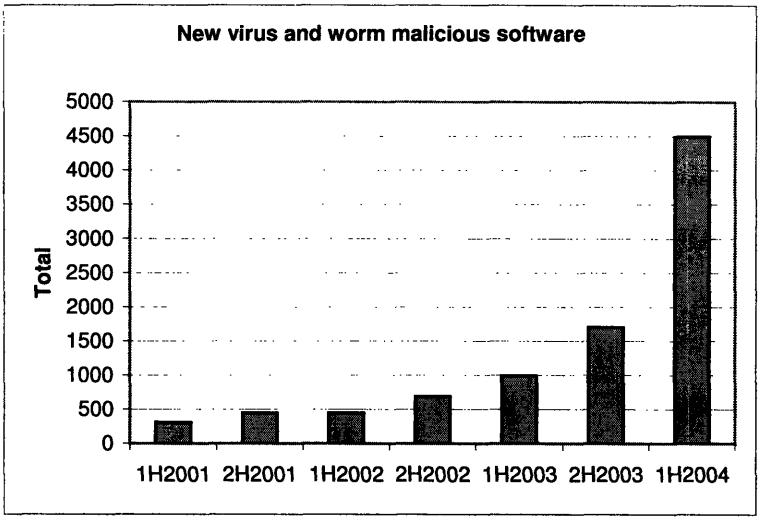


Figure 7—The amount of malicious software is increasing rapidly[70]

Law enforcement and the legislative branch of governments attempt to enforce this constraint by creating an environment where the expected cost of being caught for a virus writer is greater than the expected benefit. While the penalties are high, the perceived likelihood of being caught is low,[71] keeping the expected cost of apprehension low. At the same time, other Internet Security System components entice virus programmers with financial rewards as well as status for creating viruses, thus raising the perceived benefit of writing a virus.

In the cases of Blaster and Sobig, as well as other recent malicious software, the penalties calculated by the authors of Blaster and Sobig were not high enough to prevent them from writing the viruses. However, a recent high profile virus writer arrest along with a $5 million bounty fund from Microsoft may deter some future malware authors.

The next chapter discusses why laws and other deterrents have not been effective at stemming the flow of new malicious software.

Other violated constraints

[70] Symantec Internet Threat Report: Trends for July 04 - December 04, p10.
[71] Krebs, Brian. "Hackers to Face Tougher Sentences." Washington Post 2 Oct. 2003. <http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&contentId=A35261-2003Oct2>.

A number of other constraints were violated, but I do not believe these contributed as directly to the failure of the security system as the constraints that were discussed above. They still do play an important role in explaining the non-resilience of the Internet Security System and are included in Appendix 2. Some of these violated constraints are:

- **1-2: If compromised, computers must not spread malicious software**—once compromised, the computers continued to spread malicious software. They were not sufficiently contained by users, network devices, ISPs, or anti-virus software
- **2-1: Devices must be properly configured and maintained to prevent malicious network traffic from accessing computers**—many network devices were helpful in the fight against the Blaster worm, but many others were not correctly configured by home users or by an enterprise's technical staff and allowed the Blaster worm to enter the network
- **11-1: Information aggregators must have access to information about Internet threat activity**—In order to gather a complete picture of the malicious software threat, information aggregators such as think tanks collect attack information from various sites on the Internet. Some firms are reluctant to share their attack statistics because they feel it can reveal real or perceived weaknesses
- **11-2: Think tanks and information aggregators must recognize and communicate dangerous trends**—computers connected to the Internet have become increasingly homogeneous. Some think tanks have identified this as a threat that allows malware to spread broadly and quickly, but it has not been communicated vocally enough to cause any changes in behavior
- **15-1: The research community must support security activities with basic research**— The President's Information Technology Advisory Committee published a report in February 2005 saying that the current rate of technology transfer from research to commercial use was not rapid enough.[72]
- **13-1: The government must take action to prevent cyber attacks**—In addition to other measures, the government attempts to prevent cyber attacks by funding

[72] Benioff, Marc, and Edward Lazowska. National Coordination Office for Information Technology Research and Development. Report to the President: Cyber Security: A crisis of prioritization . 28 Feb. 2005. <http://www.itrd.gov/pitac/reports/20050301_cybersecurity/cybersecurity.pdf>.

security research. However, the President's Information Technology Advisory Committee published a report in February 2005 saying that the research community was under funded and should have their funding increased by $90 million annually.[73] Perhaps in response to that report, the NSF recently created a cyber security center that will receive $3.8 million per year.[74] Additionally, a harsh 2004 report from the Office of Inspector General finds that the Department of Homeland Security's Cyber Security Division (NCSD) is under performing.[75] The report found that the division failed to provide sufficient leadership to the private sector. Compounding the problem, the NCSD has seen significant leadership changes, with 3 chiefs in less than 2 years.[76]

---

[73] Benioff, Marc, and Edward Lazowska.
[74] Weiss, Todd R. "NSF establishes cybersecurity center." Computer World 12 Apr. 2005. <http://www.computerworld.com/securitytopics/security/story/0,10801,101024,00.html>.
[75] Lemos, Robert. "Report: Federal cybersecurity effort needs improvement." CNET News.com 23 July 2004. <http://news.com.com/Report:+Federal+cybersecurity+effort+needs+improvement/2100-1009_3-5281898.html>.
[76] Lemos, Robert. "Yet another cybersecurity chief steps down." CNET News.com 12 Jan. 2005. <http://news.com.com/Yet+another+cybersecurity+chief+steps+down/2100-7348_3-5534064.html>.

Chapter 5—STAMP Behavioral Dynamics

In the last chapter I attributed the limited resilience of the Internet Security System to the fact that numerous constraints were inadequately enforced. This chapter offers explanations for why these key constraints were not enforced.

The components that make up the Internet Security System (ISS) experienced significant change over the past decade. This chapter contains the third part of the STAMP analysis and attempts to understand what forces drove those changes as well as how the components of the system responded to the changes. Explaining these responses helps us understand why the constraints could not be enforced. This understanding can be used to develop and to value new approaches to Internet Security.

In order to understand the forces driving those changes, I use a combination of System Dynamics and economic concepts. System Dynamics is a method developed by MIT professor Jay Forrester in 1956 for understanding complex systems that exhibit nonlinear behavior.

In this chapter, I discuss four interesting examples of constraints that were not sufficiently enforced. I argue that these constraints were not enforced because of how the system responded to changes that had occurred over the past decade. Interestingly, the evolutions that occurred in the system were generally advancements, welcomed by users of the Internet. Unfortunately, many of these advances, while rational and beneficial for the parties that entered into them, had harmful effects on the Internet Security System as a whole.

The examples of inadequate control I revisit in this chapter are:

- Users did not invest sufficient resources to secure their machines or help to stem the rapid infection of other machines (led to the violation of constraint 1-1)
- Enterprises did not invest sufficient resources to secure computers under their control (led to the violation of constraint 1-1)
- Companies created and released software with vulnerabilities (led to the violation of constraint 6-1)

- Laws and law enforcement were ineffective at preventing developers from writing malicious software (led to the violation of constraints 16-1 and 17-1)

The last decade

By most accounts, the previous decade was full of exciting advances on the Internet. The Internet changed from a network used by a small number of academics and early adopters to a network rich with information and communication possibilities, which attracted the technically savvy as well as technically unsophisticated. In this 10-year time period, the number of Internet users climbed from 15% of American adults to over 60%.[77] Ever faster, powerful, and cheaper computers coupled with the widespread availability of faster and cheaper, always-on high-speed Internet connections spurred these increases. As technical improvements brought more information and more users online, the Internet became an even more useful and entertaining destination and others were persuaded to come online as well.

According to Berdnt, the price of computers (adjusted for performance) decreased by approximately 40% per year in the period 1995-2000. Even when not adjusted for performance, the cost of a computer sufficient for using the Internet continued to fall.[78]

Home broadband was not available a decade ago; today almost 60 million American households have broadband.[79] Numbers describing European connectivity rose in a similar manner.

Nothing comes without a price and the capabilities and options brought by the Internet invited exploitation. Participants in the Internet Security System had responsibilities that would have reduced the risk, but as I showed in the previous chapter many were remiss in their duties. Key failures are discussed in the upcoming sections where I explain why participants' responses to the changes in technical landscape were predictable and rational.

---

[77] Internet Evolution. Comp. Susannah Fox, and Lee Rainie. 25 Jan. 2005. Pew Internet & American Life Project. <http://www.pewinternet.org/PPF/r/148/report_display.asp>.

[78] Bendt, E R. "Price and Quality of Desktop and Mobile Personal Computers: A Quarter-Century Historical Overview." American Economic Review (2001).

[79] US Broadband Penetration Grows to 57% in March-April. 18 Apr. 2005. With data from Nielsen//NetRatings. <http://www.websiteoptimization.com/bw/0504/>.

Users did not invest sufficient resources to secure their machines or help to stem the rapid infection of other machines

The analysis in the previous chapter showed that personal computers are frequently left unpatched by their owners. We can use System Dynamics and economic concepts to understand why these computers remain perpetually unpatched.

When confronted with improvement in usability, speed, and price in the home computer and ISP market, more consumers purchased computers and moved online, initiating a decline in the security abilities and knowledge of the average user.

These new, inexperienced users are less likely to maintain their computers for a variety of related reasons. Less savvy users were likely to not appreciate the fact that their computers are vulnerable or even to know if they have been compromised.

Even among the more savvy users that implicitly consider the cost compared to benefit of addressing their vulnerable computers, many choose inaction due to the lack of measurable return from preventive security actions.

Compounding the problem, lock-in effects and network externalities give incentives to individual users to select and remain with the market leader of specific software applications. As discussed earlier, this creates a homogenous computing environment, which creators of malicious software also benefit from.

I now visit the individual aspects of this argument in greater depth.

As the prices of computers dropped and broadband became more accessible, a different type of user came to the Internet. The usability of computers is improving and they require less knowledgeable users. Users include young children, senior citizens, and those generally disinterested in the inner workings of their computers. As usability improves, users need less skill to operate their computers. However, users still do need operational knowledge and computer skills to make intelligent decisions about personal investments in computer security.[80]

Figure 8 is a System Dynamics model that shows the forces that attracted new users to the Internet. The model shows how the average security ability of users declined as new users came online. In System Dynamics, this behavior can be explained via

[80] CyberInsecurity: The cost of monopoly. Rebecca Bace, et al. <http://www.ccianet.org/papers/cyberinsecurity.pdf>.

reinforcing loops (denoted with **R**s in Figure 8). The model contains three reinforcing loops: affordable computers and broadband as well as the possibility of communicating online with a growing number of friends and associates lured new users to the Internet. The increase in users fueled the decline in computing cost and broadband cost. The increase in Internet users also resulted in increased communication potential for potential new users, which continued to drive down cost and increase communication possibilities.

The model also contains a *balancing loop* (denoted by a **B**), which slows the behavior of the system. The balancing loop in the model shows that when the level of Internet crime increases, the attractiveness of the Internet decreases. In turn, the rate at which new users come online slows. As the rate of new users coming online slows, so does the decline in the average Internet security ability and the level of Internet crime. However, this balancing loop is not yet as strong as the other reinforcing loops, which drive the system's behavior.

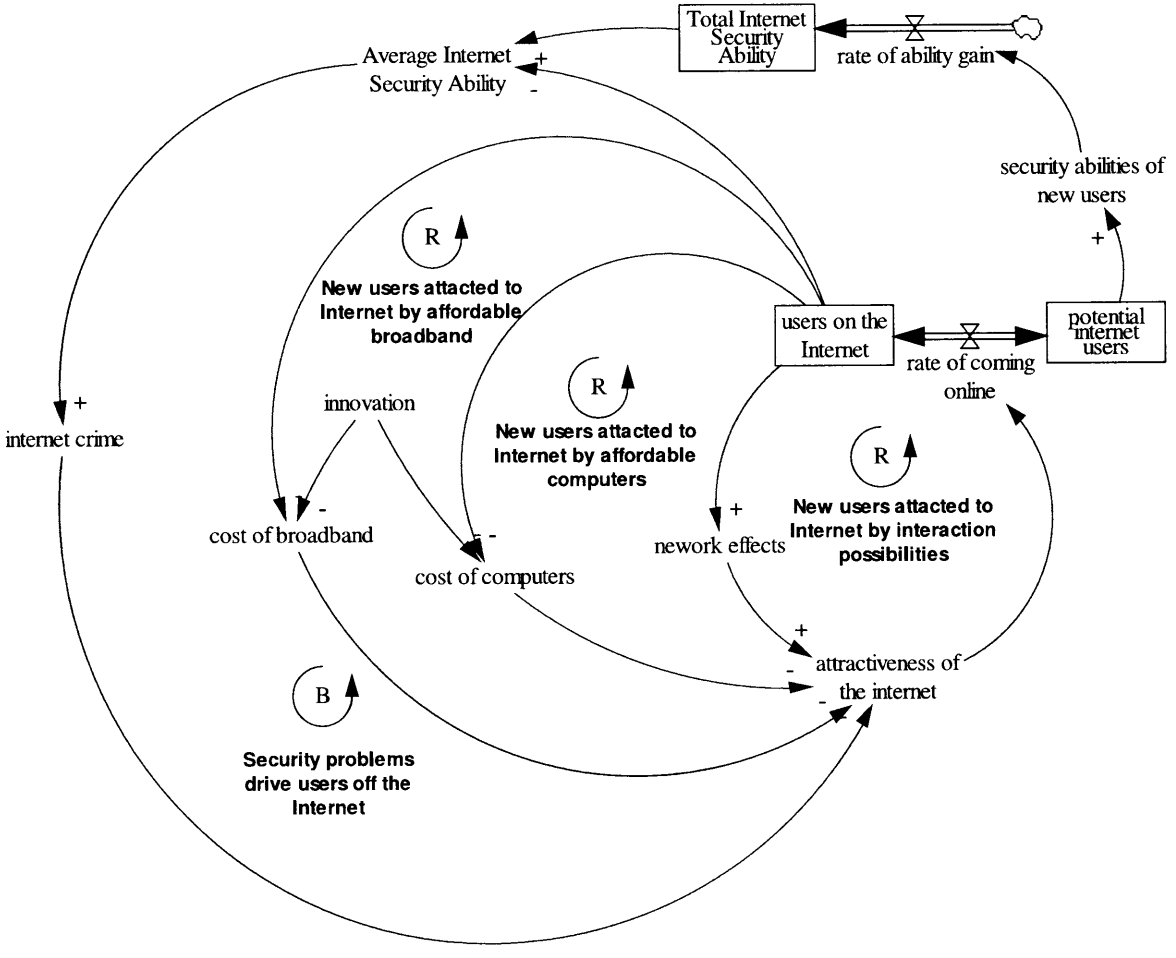

Total Internet Security Ability
rate of ability gain
Average Internet Security Ability
security abilities of new users
R
New users attacted to Internet by affordable broadband
users on the Internet
potential internet users
rate of coming online
innovation
R
New users attacted to Internet by affordable computers
internet crime
R
New users attacted to Internet by interaction possibilities
cost of broadband
nework effects
cost of computers
attractiveness of the internet
B
Security problems drive users off the Internet

Figure 8

This limited security knowledge creates problems because these users are ill-equipped to determine the amount of money, effort and time they wish to invest in securing their personal computers.

Anecdotally, many computers go unpatched because their owners simply do not understand that they are insecure. Recent studies from AOL, AOL UK, and the National Cyber Security Alliance (a public-private partnership "focused on promoting cyber security") found that 80% of survey respondents' computers had been infected with malware, in most cases unbeknownst to them. The survey also found that over 70% of respondents believed their computers were outfitted with anti-virus software configured to update daily, when in fact, only 33% actually have anti-virus software installed and configured to update at least once a week.[81,82] 20% of computer users reported not knowing what to do to protect themselves online.[83]

Even users who have operational knowledge sufficient to make informed decisions about personal computer security will still tend to make suboptimal decisions, albeit for different reasons. This is because computer security can be viewed as a negative externality, meaning it has consequences that are not reflected in the cost of the good.[84] In the Internet security context this means "lack of security on one machine can cause adverse effects on another".[85]

To understand this, we need to consider the two types of security breaches to which computer users are vulnerable. The first type—a targeted attack—consists of a hacker compromising a computer with the intention of harvesting some piece of personal or financial information or destroying personal data (Figure 9). The second type—a broad attack—occurs when a computer is compromised with the intention of making it

---

[81] Lemos, Robert. "Plague carriers: Most users unaware of PC infections." CNET News.com 25 Oct. 2004. <http://news.com.com/Plague+carriers+Most+users+unaware+of+PC+infections/2100-1029_3-5423306.html>.

[82] "AOL/NCSA Online Safety Study.": AOL and National Cyber Security Alliance, 2004. <http://www.staysafeonline.info/news/safety_study_v04.pdf>

[83] " 'Geek speak' confuses net users." BBC News World Edition 6 Apr. 2005. <http://news.bbc.co.uk/2/hi/technology/4413155.stm>.

[84] Pindyck, Robert, and Daniel Rubinfeld. Microeconomics. 5th ed.: Prentice Hall, 2001. p 47. S. J. Liebowitz and Stephen E. Margolis Journal of Economic Perspectives, Volume 8, Number 2, Spring 1994.

[85] Camp, Jean. "Pricing Security." Economics of Information Security. Comp. Jean Camp, and Stephen Lewis: Springer, 2004.

participate in some sort of distributed scheme as part of a botnet, such as a distributed denial-of-service attack (DDoS) or a spam server. (See Figure 10)

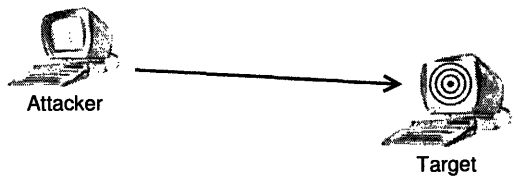


Figure 9

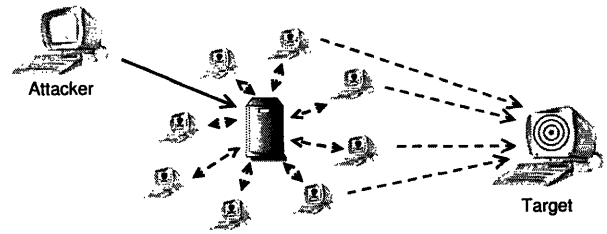


Figure 10

An investment in security by a computer user will help prevent damages resulting from the first type of attack. (Figure 9) For instance, protecting your computer reduces the chance that a targeted attacker will be able to gain access to your financial information. But interestingly, if a computer is compromised by a broad attack with the intention of using it to send SPAM or launch a DDoS attack, its owner is not severely affected. (Figure 10) They may experience some degree of performance degradation, but the real target of the attack is another computer. Similarly, investing in security by installing the latest patches or running appropriate anti-virus software will not protect a user from DDoS attacks or help them to receive less spam. Their lack of investment results in a cost to someone else, not to themselves.

The following two System Dynamics diagrams (Figure 11 and Figure 12) illustrate that when users are individually targeted, they are more likely to take actions that improve the overall security of the Internet.

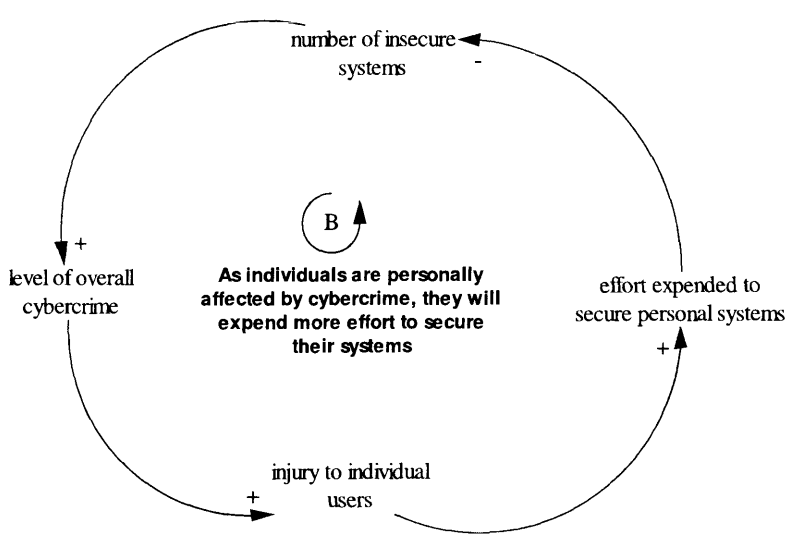


Figure 11—A targeted attack

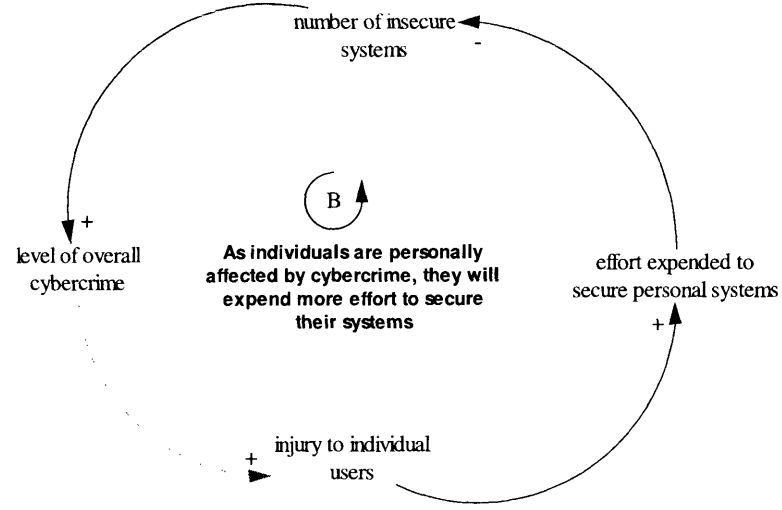


Figure 12—A broad Attack

Nobel Prize winner, Mancur Olson termed this a class of problem a collective action problem.[86] Those in a position to act to secure the Internet by patching their computers have little incentive because the benefits of their actions are spread among all the users of the Internet. They receive the benefits of other party's security investments whether or not they invest in their own computer security. Without a way to limit the benefits of proactive computer security to users that exhibit such behavior, there is limited incentive to invest in personal computer security.

Another economic reality, network effects, also contributes to the difficulties users have maintaining the security constraints imposed on their computers. A network effect increases the benefit of a good as more people use it. There are two types of network effects: direct and indirect. Direct network effects[87] are the benefits users of a good receive from interacting with other users of the good. The telephone is a commonly cited example of a direct network effect.

The other type of network effect, an indirect network effect, is the benefit one receives from the ancillary services that are available because there is a strong user community. For instance, there is an indirect network effect in having a popular make of car. Because the car is popular, there are many auto mechanics able to perform maintenance on it.

Network effects also inform computer user's choices. As new users come online, they must choose what type of computer and software to purchase. There are strong indirect network effects that drive a computer user's choices. Selecting an operating system or software package with a large user base means there will be a greater selection of software, more easily accessible knowledge and training, and potentially even lower prices. (Direct network effects are less significant because most Internet services interoperate across operating systems.)

The indirect network effects will result in less diversity in decision making by consumers, resulting in large installed bases of software or operating systems, such as Microsoft Windows. Homogenous platforms enable malicious software to spread faster

---

[86] Olson, Mancur. The Rise and Decline of Nations: Economic Growth, Stagflation, and Social Rigidities. Chapter 2, New Haven: Yale UP, 1984.
[87] Pindyck.

and also attract more persistent attention from malicious software developers searching for exploitable vulnerabilities.

Enterprises did not invest sufficient resources to secure computers under their control

Similar economic forces also drive those in charge of managing networks at large corporations and educational institutions. As was true for individual users, enterprises are frequently reluctant to act because of security externalities and the collective action dilemma. Direct as well as indirect network effects affect their decisions. Finally, lock-in plays a factor in their decision-making as well. The remainder of this section discusses how these factors play a role in security decision-making for enterprises.

Enterprises encounter security externalities in two ways. The first is similar to what individual users experience. Like individual users, enterprises can also be the targets of two types of attacks. The first, as above, targets the enterprise, hoping to acquire sensitive information or to damage data. The second type attempts to compromise workstations that will later be used to launch a DDoS attack or send spam. Patching an insecure workstation will help prevent both types of attacks, but much greater economic harm could come to the firm from the first type of attack. For enterprises, the likelihood of the first type of attack is greater, so the incentives for proactive security would be stronger.

The second type of externality involves sharing information about attempted and successful attacks. As discussed in chapter 3, information integration organizations like US-CERT and the Internet Storm Center collect attack information to try to form comprehensive pictures of emerging threats. Gal-Or and Ghose show that there are benefits to sharing security information.[88] However, companies may be reluctant to share such information, fearing that it may be used to launch better attacks or may hurt the perception of their company and lead to decreased market share or stock price, if made public. These fears are partially confirmed in research by Campbell and Gordon[89]. They

[88] Gal-Or, Esther. "The Economic Incentives for Sharing Security Information." Economics of Information Security. Ed. Jean Camp, and Stephen Lewis. City: Publisher, 2004.

[89] "The economic cost of publicly announced information security breaches: empirical evidence from the stock market" Campbell, Gordon, Loeb, Zhou; Journal of computer security 11(2003) 431-448.

found that breaches resulting in lost confidential information result in "significant negative market reaction". However, no such correlation was found when the breaches did not result in the loss of such information.

As was true for personal users, network effects also affect enterprises' decision making, but in additional ways. Companies find benefits in having homogenous infrastructure; they save money on training and support and can additionally enable certain types of collaboration requiring similar platforms. Once an enterprise has invested heavily in a technology, they are less likely to change due to lock-in. These forces also lead to homogenous environments, which, as I discussed earlier, allow malicious code to spread farther and faster.

Companies release software with vulnerabilities

Technically speaking, creating large-scale commercial software is an extremely difficult process. The complexity of the problem makes software defects a near certainty. Compounding this already difficult problem, the realities of software economics create a perverse incentive structure that results in insecure code. More specifically, software vendors have incentives to race to market, releasing software before fully vetting it. Once a user base has been captured, a software company can invest in a lower level of feature and security improvements because their customers are "locked-in" to their current product. Finally, the market for software did not place a high value on security features until recently.

In 1988 Frederick Brooks wrote, "Software entities are more complex for their size than perhaps any other human construct..."[90] That was 17 years ago and software has only gotten more complicated, with many major software applications encompassing millions lines of code. Brooks argues that the "hard part" of building software is not the construction of the code, rather it is the specification, design, and testing. The software engineering community has developed techniques for working through this complexity, but it remains impossible to create perfect software. Complex software has far too many

[90] Brooks, Frederick P., "No Silver Bullet: Essence and Accidents of Software Engineering," Computer, Vol. 20, No. 4 (April 1987) p. 10-19.

different states to exhaustively check them all.[91] Leveson writes that exhaustive testing is impossible for most software; testing can find defects, but it cannot prove the absence of defects.[92]

The complexity of software engineering creates a hurdle that is only exacerbated by the economics of the software business. Creating software has a very high upfront cost, but creating additional copies of software costs almost nothing. In other words, software has a high fixed cost and very low marginal cost. Following the substantial initial investment, software companies try to recoup their investment by selling as much of their software as they can.

This fact, when coupled with the network effects and lock-in discussed earlier, creates an environment with a strong incentive to be the first to market. By taking market share early, software vendors capture users and become an early market standard. If the software vendors can capture a critical core of users, they can resist attacks from competitors because their clients are benefiting from the network effects and constrained by the cost of switching to new software. This creates an incentive for firms to shorten their development cycle to get to market first.[93]

In order to further explore these software economic facts, I created an executable System Dynamics model. (The full model structure is available in Appendix 3.) The model embeds some of the economic forces that affect the software marketplace.

It models a simplified competitive landscape of just two companies (Company A and Company B), each with a product (Product A and Product B). The two products have identical feature sets. The companies compete only on their quality. The model is based on the following assumptions:

- Users are either customers of only one product at a time, but can switch if they choose
    - Users will begin to switch products if the quality of their product is lower than the quality of the other product. The greater the quality differential, the more users will switch.

---

[91] An analysis by Leveson of a TCAS II, an aircraft collision avoidance system shows that it has 10^{40} different states. (Leveson. A New Approach to System Safety Engineering, Chapter 3)

[92] Leveson, Safeware, Chapter 18

[93] Raman, Jari. "Network Effects and Software Development – Implications for Security." Proceedings of the 37th Hawaii International Conference on System Sciences (2004).

- Users will also switch if the network effects of the competing technology are great enough to incite them to switch.

- The software companies will try to maximize their profit by retaining customers while minimizing investment in their product.

Executing the model shows interesting behavior in the sample market that is useful for understanding actual software markets:

1) A strong network effect makes the marketplace "tippy" and likely to be dominated by one competitor
2) Network effects allow the market leader to maintain their position even if their product contains more defects than their competitor
3) The stronger the network effect, the less a company needs to invest in product quality to maintain market leadership
4) Increasing investment in product quality leads to a greater degree of market ownership

Before delving into these observations from the model, it is helpful to understand a baseline run. The graph below shows a case where Product A began with slightly more of the market than Product B. A and B used the same investment strategy; how much effort to put into improving their products is a function of their market share. They invest heavily if they are far behind and lightly if they are in the lead.

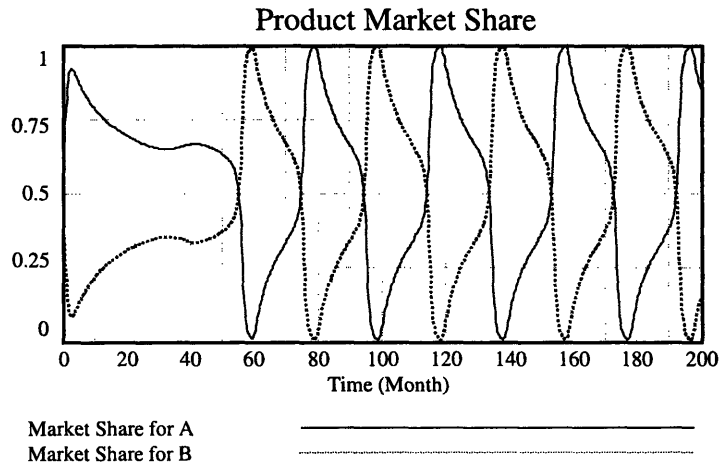


Figure 13

The above graph (Figure 13) shows that the two companies continually trade market leadership. Clearly, their investment strategy is only partially working. When holding market leadership, each company under invests and is overtaken by its competitor. The graph below (Figure 14) shows how decreases in Company A's product quality is met with increased development efforts. Because Company B follows the same strategy; a similar graph can be drawn for Company B.

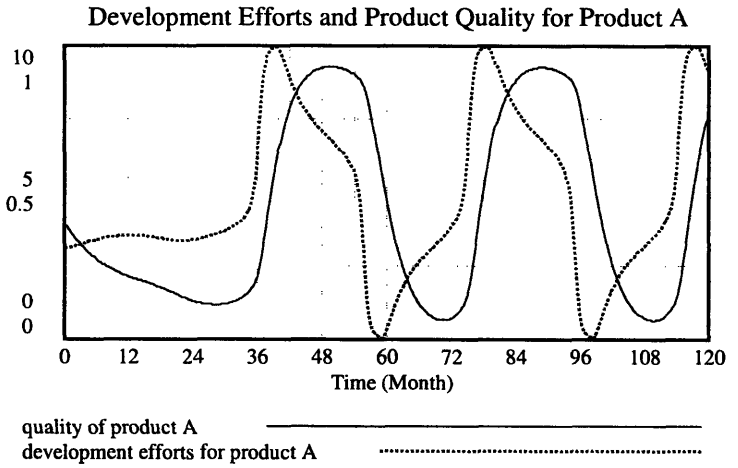


Figure 14

With a basic understanding of the model, we now turn to the insights I draw from the model.

A strong network effect makes the marketplace "tippy" and likely to be dominated by one competitor

The baseline graphs above were drawn with only a very small network effect present, but the following graphs show the change in behavior when the strength of the network effects is increased. The figure on the left is the same as Figure 15 above, but the graph on the right is the output when I strength the network effects. Using the same strategy, the market "tips" and allows Company A to maintain market leadership because of the "stickiness" brought upon by network effects.

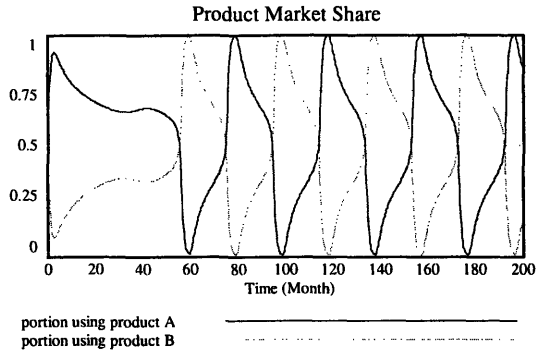


Figure 16—Slight network effects

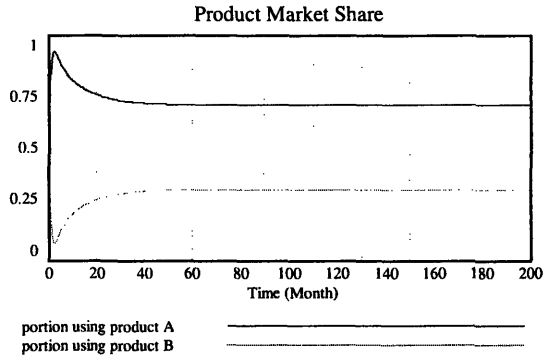


Figure 17—Strong network effects

Network effects allow the market leader to maintain their position even if their product contains more defects

The two graphs below show the results of one simulation run with strong network effects. The system equalizes with Company A owning approximately 70% of the market (Figure 18). The second graph (
Figure 19) shows that at equilibrium, the market leader has more defects than its competitor. This is because the benefits from network effects and costs of switching more than compensate for problems associated with quality and customers choose not to switch.

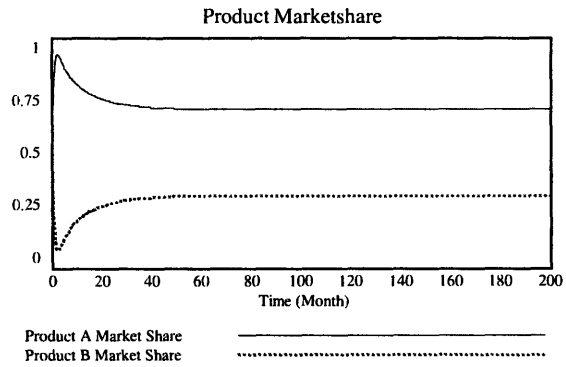


Figure 18

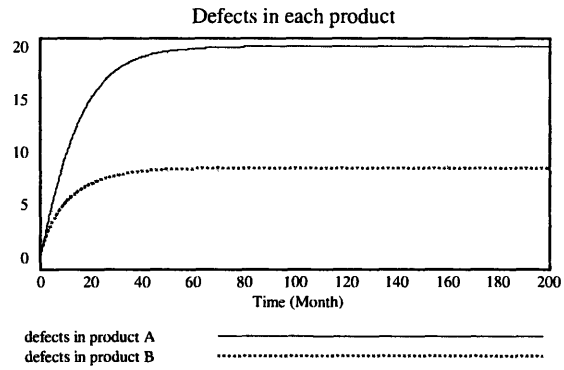


Figure 19

Companies need to invest fewer resources in product quality to maintain market leadership when network effects are high

Incumbent companies in industries with strong network effects can invest less than their competitors in software quality and still maintain market leadership. The stronger the network effects, the less the incumbent needs to invest, the model shows. To find this, I expanded the model to incorporate variation in Company A's strategy. The extension allows Company A to vary their investment in software. By trying different levels of investment, I was able to find the level of investment where the market behavior shifts from oscillation to Company A's market dominance.

Experimentation with the model shows that the greater the strength of the network effects, the less investment in software quality is needed to end the oscillation and secure market leadership. Figure 20 shows that inverse relationship.

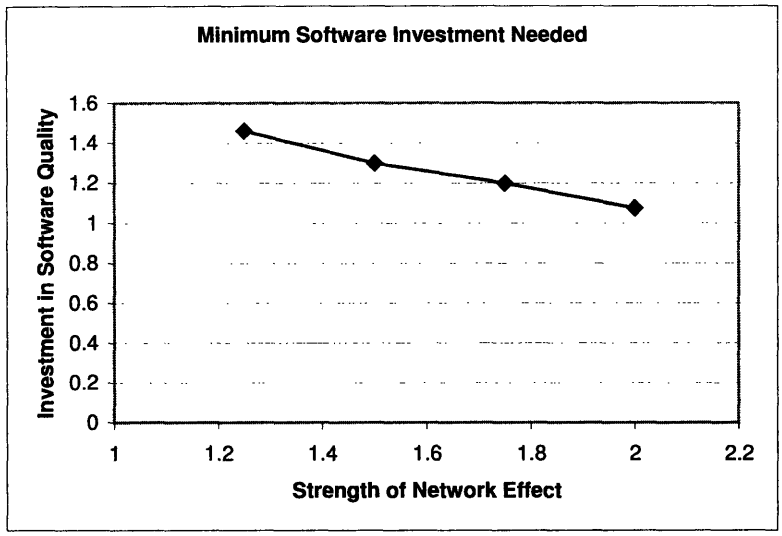


Figure 20

Increasing investment in product quality leads to a greater degree of market ownership

Not all incentives in the software marketplace are at odds with security, however. The model also shows that increasing investment in software quality initially change the market behavior from oscillation to market dominance. Additional investment will then lead to a larger portion of the market being acquired, as seen in Figure 21.

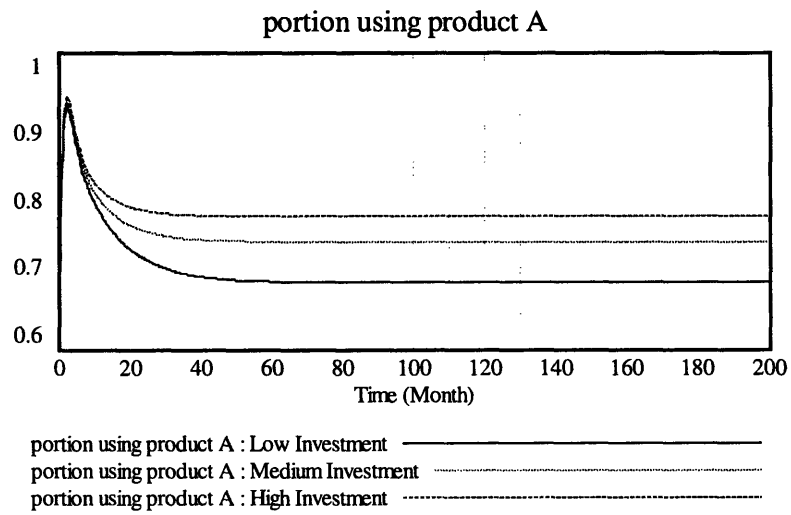


Figure 21

The above insights from the analysis of the model are interesting, but are by no means without caveats. This is a very simple model; an abstraction designed to show some key relationships. Most of the core assumptions are not replicated in the real world. For instance, marketplaces generally have more than two competitors who sell products that differ in feature set as well as quality. The competitors also employ more advanced strategies pricing and investment than this model contains.

Laws and law enforcement were ineffective at preventing developers from writing malicious software

Most of the older literature on "hacker" motivations paints a picture of a hacker as a person primarily bent on earning intangible rewards. Beveren describes hackers as people seeking to satisfy their curiosity, exert power or control over something that is not theirs, or simply be recognized and embraced by their peers.[94] This omits the fact that *some* hackers have always been motivated by financial rewards, but in recent years, especially the past 1 ½ to 2 years, *typical* hacker behavior has progressed from malicious behavior to monetizing attacks.[95,96]

Why did this progression occur? The broad changes of the last decade created more avenues for programmers to reap illicit financial rewards by writing malicious code. Conditions were right to create a supply of compromised computers. The increase in personal computers and the simultaneous availability of broadband created a deep supply of vulnerable computers that were ripe for compromising. Relatively easily, and without their owners knowing, these computers could be compromised and added to botnets. On the demand side, a market emerged of people willing to lease the computing power of these botnets to launch moneymaking schemes such as sending spam or launch denial of service attacks. The presence of an easy supply of computers to compromise and the promise of payment for these computers fuels the ongoing trade in computing power for nefarious purposes.

---

[94] Beveren, John V. "A Conceptual Model of Hacker Development and Motivations." Journal of E-Business 1 (2001).
[95] LaMacchia.pdf
[96] "Hackers to Face Tougher Sentences." Washington Post 2 Oct. 2003. <http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&contentId=A35261-2003Oct2>.

Chapter 6—Addressing Internet Security

The previous chapter showed that Internet security is often at odds with the other interests of the involved parties. Actors are often forced to choose between improving their own welfare and improving the overall security of the Internet. Not surprisingly, they frequently make choices that impair the security of the Internet. This is a case of misaligned incentives that can be traced back to the nature of software and the Internet, as I showed in chapter 5.

In this chapter, I discuss multiple approaches to improving Internet security. These approaches fall into two principle categories. The first category contains solutions that attempt to realign peoples' incentives so that they engage in behavior that improves their welfare as well as improves the security of the Internet. This can be done using law, regulations or market forces to provide incentives for improved behavior (or penalties for unhelpful behavior).

Alternatively, solutions in the second category signal an acceptance that we cannot satisfactorily realign incentives without an unacceptable level of institutional policy. Instead, solutions in this category rely on technology as a "safety net", working to secure the Internet even as human behavior is pushing it in the opposite direction.

This chapter contains three subsections. The first two are devoted to realigning incentives to improve peoples' security behavior. They focus on changing laws and using insurance or market principles to improve security behavior. The third subsection discusses different technical approaches.

Changes to laws

As discussed in chapter 4, there are two legal pathways that are designed to hold software producers accountable for harm related to insecure software they have unintentionally created.[97] They were designed before Internet security became a significant problem and were not intended to address problems created by recent threats like malicious software and botnets. As such, they were not especially effective. In my

[97] Other criminal and tort pathways exist for prosecuting cases where software producers intentionally release insecure products.

research I did not find any cases where product liability has been used to hold a company liable for damage relating to a security flaw.

In this section, I consider if variations to the current product liability regime would result in more secure software. A 2002 National Academy of Science report already commented on this issue, suggesting "policy makers should consider legislative responses to the failure of existing incentives to cause the market to respond adequately to the security challenge."[98]

This is a contentious question; public policy says that the burden should be put on the least avoider, the party that can most efficiently remedy the situation. Some believe that software vendors are the least cost avoiders,[99] others feel that consumers or network operators could most efficiently remedy the situation. The following subsections discuss three alternative liability schemes:

- Expand the extent of liability on the software producer
- Assign liability to parties other than the software producer
- Strengthen criminal penalties

Expand the extent of liability on the software producer

Some people propose engendering securer software by increasing the liability of software vendors. Practically, there are two ways to do this. One way of doing this is to change the law so that economic damage caused by software defects is considered to be sufficient to invoke strict product liability rather than limiting strict product liability to just physical damage or personal harm as the law says now. A second approach would be to limit the level of indemnity that companies can attain within their End-User Licensing Agreement (EULA).

There is a large debate regarding whether this would be effective in improving software security. Proponents believe that making it easier to hold a producer responsible would result in an increase in how they determine the amount of time and resources to

[98] Lohr, Steve. "Product Liability Lawsuits Are New Threat to Microsoft." New York Times 6 Oct 2003.
[99] Pinkney, Kevin. "Putting Blame Where Blame is Due: Software Manufacturer and Customer Liability for Security Related Software Failure." Albany Law Journal of Science and Technology 13 Alb. L.J. Sci. & Tech. 43 (2002).

spend on development and testing. If the penalties for releasing insecure software increase, they will spend more time ensuring the software is secure, they say.

Not surprisingly, there are many critics of this idea. There are two frequently heard arguments against increasing liability. The first criticism is that innovation could be stifled. Increasing liability would result in companies paralyzed by the risk of releasing insecure software. In this scenario, companies would spend dramatically more time testing each release and the innovation cycle would be slowed. They also might simply choose to not release worthwhile products because the cost of liability is too great.[100, 101]

Supporters of increased liability retort that stopping some innovation is a necessary cost of security. The FDA, for example, frequently stops innovation to ensure that we have safe drugs on the market.[102]

The second criticism is that it holds software to an unrealistic standard[103]. It is impossible (or at least entirely unfeasible) to make perfectly secure software. Advocates of the idea believe otherwise and say that the techniques do exist to efficiently make near-zero defect software.[104]

Assign liability to parties other than the software producers

The second option is to assign liability to parties other than the software producers such as the consumers responsible for administering software products, who can also be at fault for outbreaks of malicious software. When software companies learn of security flaws, they release a "patch" that consumers can install to fix the vulnerability. Virus outbreaks frequently happen shortly after a patch has been released because the patch alerts hackers to the presence of a new vulnerability. The malicious software authors rely on the fact that many consumers do not install patches in a timely manner.

---

[100] McLaughlin, Laurrianne. "Buggy Software: Can New Liability Rules Help Quality?" IEEE Software (2003): 104-108.
[101] Heckman, Carey. "Two Views on Security Software Liability." IEEE Security and Privacy (2003): 73-75.
[102] "Fighting the Worms of Mass Deception." The Economist Nov 27 2003.
[103] Ryan, Daniel. "Two Views on Security Software Liability." IEEE Security and Privacy (2003): 70-73.
[104] Junnarkar, Sandeep. "UCITA: Why software users will lose." CNET News.com 17 Oct 2002. 22 Nov 2004 <http://news.com.com/2008-1082-962353.html>.

Rather than placing the entirety of the liability for damage done by a virus on the software vendor, this variation places some of the liability on the consumer who left their personal computers or their firm's systems unprotected.

If we were to begin considering consumers partially liable for their inaction, then we may see them approach their patching duties even more diligently. This increased concern might result in additional pressure from consumers for the software companies to invest more heavily in secure software at the risk of losing their customers.

This type of liability is difficult to apply because it is so diffuse. Additionally, it would be difficult to determine what is a reasonable amount of time to require that a security patch to be installed. Time is critical because a hacker can often create a new virus within days of when a patch is released. While the company may understand this urgency, they generally cannot act immediately. They face the risk of installing a new piece of software on nearly all their computers. Additionally, patches often come at unscheduled times; installing the patch means diverting IT workers from their scheduled activities.

Strengthen criminal statues

An alternate approach to broadening or shifting liability is to more aggressively target the source of the malicious software: the writers of that software and their sponsors. By strengthening the penalties for computer crimes, lawmakers could potentially deter more malicious software writers. However, I believe this is unlikely to work as penalties were significantly decreased in November 2003. Based on the number of compromised computers and the rate new malicious software is released, this has not had an appreciable effect. According to Kevin Mitnick, a hacker who spent six years in prison for a computer crime, most "hackers" do not weigh the consequences of their actions in this manner; they do not expect to get caught.[105]

Using regulation to realign incentives

[105] "Hackers to Face Tougher Sentences." Washington Post 2 Oct. 2003. <http://www.washingtonpost.com/ac2/wp-dyn?pagename=article&contentId=A35261-2003Oct2>.

Using regulations to realign interests is also an avenue worth exploring. Clever regulation that creates a structure where system participants augment system security while pursuing their own goals would likely have a beneficial effect on the security of the Internet.

This section discusses three ways regulation can accomplish this alignment of interests:

- Require selling risk to insurance companies
- Trading vulnerability credits
- Allow an emerging cyber-crime pattern to continue developing and ensure the public is aware

Require selling risk to insurance companies

Requiring companies to buy "network insurance" might have beneficial effects for the Internet's resilience. Insurance companies are in the business of buying different types of risk from companies and individuals and have the capability to determine the appropriate amount to charge for premiums. For businesses, they provide the benefit of taking an unlikely, but potentially large or catastrophic expense and translating it into a predictable annual cost.

As they do for fire and theft insurance, Internet insurance companies would drive better behavior with the promise of lower premiums. Just as a bank can lower their premiums with a metal detector,[106] a company could lower their premiums by taking prudent security steps or having long periods without security problems.[107] This in turn would drive innovation, as the insured demand security features and innovations in the software they buy and software and technology producers strive to create products that are endorsed by the insurance companies.

Software producers could also turn to insurance companies. They could reduce their premiums by investing in new development and testing methods and technologies.

---

[106] Schneier, Bruce. Lecture: Security, Liberties, and Trade-Offs in the War on Terrorism. The Fifth Hope Conference. Hotel Pennsylvania, New York City. 9 July 2004.

[107] Schneier, Bruce, "Hacking the Business Climate for Network Security," IEEE Computer, p86-88 (April 2004)

Trading vulnerability credits

The general approach to addressing externalities such as Internet security is to "internalize" the externality. To internalize an externality means the costs that are normally projected on parties other than the decision-maker are somehow included in the decision-maker's calculations. Prior to 1960 it was thought that only governments—through regulations, taxes, and subsidies—could enact this internalization. In 1960, Nobel prize winner R.H. Coase showed that a more economically efficient internalization could happen through private negotiation.[108] For instance, a commonly given example assumes a factory located next to a farm. The factory pollutes the air, damaging the quality of the farm's crops. Whether the factory has the legal right to pollute or the farm has the legal right to fresh air, letting the party without the legal standing (whichever one that may be) pay off the party with the rights will result in an economically efficient outcome, says Coase.

A modern day example of this theory is seen in emissions trading. Involved companies buy and sell credits that give them the right to put a certain amount of pollution into the environment. Companies that have invested in decreasing their emissions rate need fewer credits and can sell them on an open market. Correspondingly, companies using unclean technology will need to buy more credits. This controls the amount of pollution while providing incentives to companies to make their operations more efficient.

In her paper, *Pricing Security,* Jean Camp proposes a similar system where computer owners are "charged" for vulnerabilities in software running on their system. Participants would receive a certain number of "vulnerability permits" per device.[109] In addition to their standard prices, software packages would have vulnerability price as well. Running a piece of software would require that the machine's owner has procured the appropriate vulnerability permits. Software vendors with high vulnerability prices

---

[108] Biography of Ronald Coase. New School University. <http://cepa.newschool.edu/het/profiles/coase.htm>.

[109] Another option briefly explored by Camp considers putting the onus on software producers, requiring them to buy permits to pay for the vulnerabilities they create. Camp argues that this structure would place a disproportionate burden on free software producers and significantly chill open source development. Interestingly, Coase wrote that it does not matter to which party the responsibility is given—an economically efficient outcome will result either way.

would presumably find that their effective prices are higher than their competitors and work to improve their code quality.[110]

To better understand how the system would work, consider this scenario. Imagine the system is at equilibrium; all computer users have purchased sufficient vulnerability permits for their current configurations. A new vulnerability is discovered in a popular web browser. Users of that browser would have some small period of time (perhaps two days) to fix the vulnerability (either by patching the software or removing it). After the allowable time period elapses, the number of vulnerability permits required to run the software would increase. People who wish to continue running the browser will need to procure more permits on the open market.

This is an interesting system and has worked well in pollution-creating industries. I believe the ideas behind Camp's system are sound, but initiating and administering the system would not be feasible.

Working as designed, the permit system does a fine job of internalizing the cost of network security. When new vulnerabilities are discovered, users of that software must invest their time in fixing the vulnerability or invest their money in purchasing new credits. In the longer term, these users will either move to software with a reputation for being more secure or will pressure the software vendor to create more secure code.

The elegance of the system is appealing, but the details of administering such a system are unwieldy. There is not an entity with the authority to create such a system at this point. If it were to be created by the US government, it would only bind US computers. International organizations like the UN or ICANN (Internet Corporation for Assigned Names and Numbers) would understandably become mired in creating and administering a program affecting hundreds of millions of computers.

Another contentious issue is determining the number of permits users would need to purchase in order to run a particular piece of software. Presumably some governing board would create a methodology for determining the cost of vulnerabilities. This would inevitably become highly political, with companies lobbying the board for lower price assignments.

---

[110] Camp, Jean. "Pricing Security." Economics of Information Security. Comp. Jean Camp, and Stephen Lewis: Springer, 2004.

Finally, monitoring and enforcement would be extremely difficult. Camp proposes a "citizen's militia" responsible for checking for compliance. Another option is a trustworthy mechanism for computers to report to the authorities when computers are running software without sufficient vulnerability credits. Both options present significant privacy concerns.

Additionally, if a piece of malicious software was able to exploit a sanctioned vulnerability and install itself, this vulnerability trading structure would not provide incentive for users to remove it from their machines. Furthermore, it also would not prevent users from unwittingly executing malicious software, as is the case with email attachments.

Allow an emerging cyber-crime pattern to continue developing and ensure the public is aware

As discussed in chapter 5, most botnet attacks use compromised computers to launch DDoS attacks or send spam to other targets (Figure 9 and Figure 10). However, a recent study by Symantec shows malicious software writers and sponsors are more frequently scouring infected computers hard drives, looking for personal or financial information that can be exploited.[111] As this practice increases, so does the potential cost to individual computer users.

If this practice increases and is well publicized, it may result in users taking more action to protect their computers.

Technological Solutions

Another approach to Internet security is to conclude humans are a weak link in the security chain. The previous chapter shows that users are either incapable or unwilling to expend the time and effort to accumulate the knowledge needed to successfully protect their systems. Simultaneously, the time elapsing between the discovery of a vulnerability and the release of a piece of malicious software has been steadily decreasing. Over time, it is questionable if human intervention will be able to intervene in time.

---

[111] Symantec Internet Threat Report: Trends for July 04 - December 04. Vol. VII.: Symantec Corporation, 2005. p 1-96.

If human action is a weakness then one approach to take is to use technology to remove reliance on human action as much as possible. For instance, rather than relying on users to patch their computers perhaps it should be done for them automatically as the newest version of Microsoft Windows XP can do.

There are a number of new technologies and proposed design changes intended to plug some of the holes created by the lack of human intervention. Below, I discuss a few of the ones that I think have potential to help. They fall into the following three categories:

- Technologies that prevent vulnerable computers from doing harm to others
- Technologies that help decrease vulnerabilities in new software
- Internet design changes that build security concepts into the Internet

Technologies that prevent vulnerable computers from doing harm to others

An unpatched computer is a danger to the computer's owner and to the Internet at large. Researchers and companies are addressing this threat with different approaches. Rather than rely on human generated virus signatures that detail exactly how a piece of malicious software looks and behaves, numerous companies including Mazu Networks, offer software that spots abnormal behavior without a signature.

Microsoft's new Active Protection Technology (APT) takes a similar approach, but observes the behavior of individual computers rather than the entire network. If the APT program notices something out of the ordinary, such as a high volume of email being sent, it can take action.

Another approach is to prevent unsecured (and potentially compromised) computers from joining a network. An aspect of Cisco's "Self Defending Network" offering interrogates computers when they plug in to an enterprise's network. If the computer is not running the most recent security patches or other security critical software it is placed in a quarantine zone until it can be fixed. When coupled with Trusted Computing,[112] a way to ensure that computers honestly report their status, the idea is even more powerful.

[112] Home Page. Trusted Computing Group. <https://www.trustedcomputinggroup.org/home>.

These technical approaches can have significant effects. Microsoft released Windows XP Service Pack 2 in August 2004. It improved the security configuration and added new security technology. In the period shortly after the release of Service Pack 2, the number of compromised computers joining botnets observed by anti-virus company Symantec decreased by 83%.[113]

Technologies that help decrease vulnerabilities in new software

New tools are being developed to help programmers write more secure software. A set of tools called *static analyzers* help developers by inspecting code after they write it. Some security conscious companies require that every piece of code a developer writes is checked before it is added to the product.

Another approach is to insert some degree of variability into programs as they are running. Doing so will not eliminate exploits, but will make it so that an exploit that works on one person's computer will not work on any other computer.[114] This would make it much more difficult for malicious software writers to spread their programs. This has not been widely deployed because it results in a small drop in performance.[115]

Internet design changes that build security concepts into Internet

Much like our behavior in the physical world is governed by certain physical laws (e.g., I can't walk through a wall), the behavior of Internet users is limited by a different set of rules. These rules are codified in the "code" of the Internet.[116] For instance, the early designers of the Internet made the design decision to instill the Internet with a strong sense of anonymity and transparency and encoded this design decision in the code of the routers and computers that comprise the Internet. These design decisions led to great freedom on the Internet; freedom that was frequently used for innovation and open communication. Of course, this is the same freedom that has allowed malicious software developers to able to take advantage of others.

---

[113] Symantec Internet Threat Report: Trends for July 04 - December 04. Vol. VII.: Symantec Corporation, 2005. p 22.
[114] Bray, Brandon . Compiler Security Checks in Depth. Feb. 2002. Microsoft Corporation. <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dv_vstechart/html/vctchCompilerSecurityChecksInDepth.asp>.
[115] Talk by Microsoft C++ Compiler Program Manager. MIT, Cambridge, MA. Nov. 2004.
[116] Lessig, Lawrence. Code and Other Laws of Cyberspace: Basic Books, 2000.

The "physical laws" that govern our behavior on the Internet are different than those that control our behavior in real space because they are only the product of a group of engineer's design decisions. These Internet laws can be changed.

One possible change comes from David Clark, a MIT professor and an architect of some of the Internet's core protocols. In a recent paper, he argues that the design of the Internet should be modified to include the concept of *trustworthiness*[117]. In an Internet that has the concept of trustworthiness, one node on the Internet would relate to another according to how trustworthy it is known to be. For instance, Clark proposes that routers will provide access to other routers that are known to be trustworthy while blocking access to unknown routers.

[117] Clark, David , et al. Addressing Reality: An Architectural Response to Real-World Demands on the Evolving Internet, Proceedings of the ACM SIGCOMM 2003 Workshops. <http://delivery.acm.org/10.1145/950000/944761/p247-clark.pdf?key1=944761&key2=0268135111&coll=GUIDE&dl=ACM&CFID=42319975&CFTOKEN=14008375>.

Chapter 7—Summary

As stated in chapter 1, this thesis has a dual purpose. I intended to test if STAMP could be effectively applied to a security problem. The applicability of STAMP for security could be judged in the efficacy of the security analysis.

I believe that STAMP served as a very useful tool for conducting the security analysis. Using STAMP, I was able to take a diverse and complex security system and develop a strong understanding of how the system responds to the challenges it faces.

The STAMP analysis began in chapter 3 by defining the Internet Security System (ISS) as a collection of 17 components whose interactions determine the security of the Internet. This part of the STAMP analysis discussed the 17 components and enumerated security constraints that, if maintained, would reduce the chance of an outbreak of malicious software.

The Internet Security System is constantly under attack, which provided interesting cases to study. In second part of the STAMP analysis I looked at two successful attacks, Blaster and Sobig, through the lens of constraints. The analysis stepped through the constraints that were violated and allowed the outbreaks to occur. Among the many constraints that were violated, I selected three as playing especially large roles in the outbreak. They were:

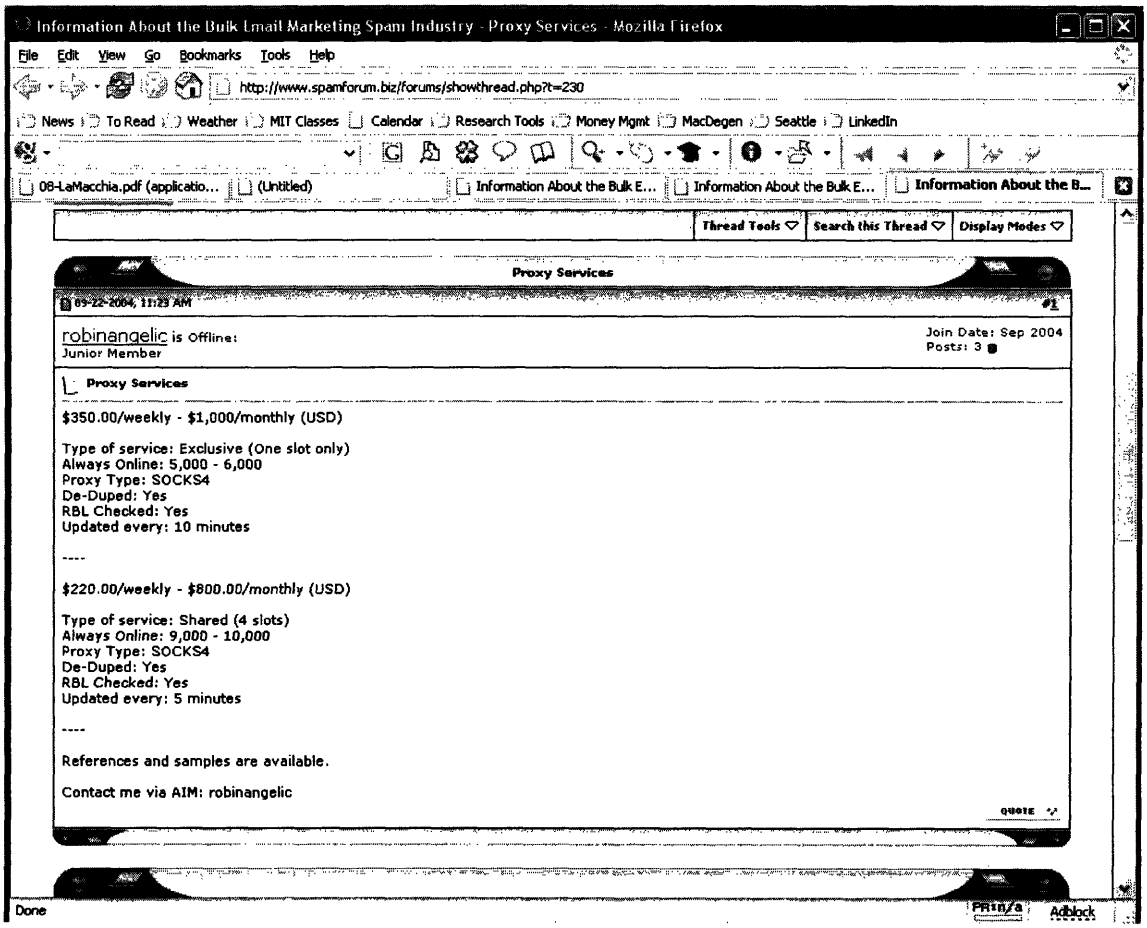
- Constraint 1-1: Computers must not execute malicious software
- Constraint 6-1: Software must not contain security vulnerabilities
- Constraint 16-1: Virus programmers must not create and release malicious software

Chapter 4 identifies the constraints that were violated; chapter 5 explains which components allowed the constraint to become violated and, more importantly, why those components were not able to keep the constraints valid. In chapter 5, I show that both individual users and enterprises were responsible for the insufficient enforcement of constraint 1-1. Software vendors did not enforce constraint 6-1 and lawmakers and law enforcement inadequately enforced constraint 16-1.

The remainder of chapter 5 discusses why these components either were not able to or chose not to enforce these critical constraints. Using System Dynamics models and

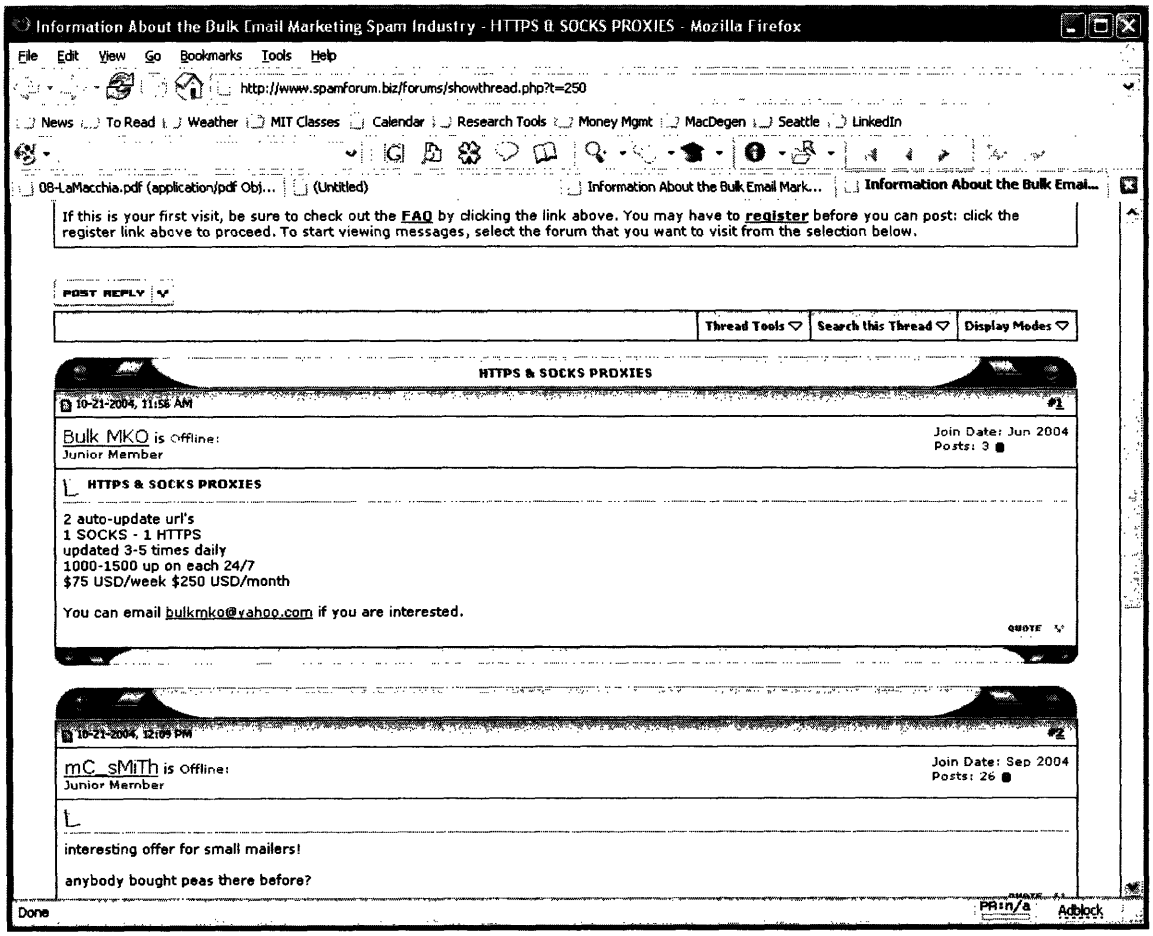
economics concepts, I arrived at the central conclusion presented in chapter 5: those responsible for providing critical parts of Internet security do not have sufficient incentives to make good security decisions; instead they often make decisions at odds with Internet security. These misaligned incentives contribute to the lack of enforcement of key security constraints, leaving the Internet less resilient to attack.

Appendix 1—Sample online advertisements offering compromised computers


Information About the Bulk Email Marketing Spam Industry - Proxy Services - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.spamforum.biz/forums/showthread.php?t=230
News To Read Weather MIT Classes Calendar Research Tools Money Mgmt MacDegen Seattle LinkedIn
08-LaMacchia.pdf (applicatio... (Untitled) Information About the Bulk E... Information About the Bulk E... Information About the B...
Thread Tools Search this Thread Display Modes
Proxy Services
03-22-2004, 11:23 AM #1
robinangelic is Offline
Junior Member
Join Date: Sep 2004
Posts: 3
Proxy Services
$350.00/weekly - $1,000/monthly (USD)
Type of service: Exclusive (One slot only)
Always Online: 5,000 - 6,000
Proxy Type: SOCKS4
De-Duped: Yes
RBL Checked: Yes
Updated every: 10 minutes

$220.00/weekly - $800.00/monthly (USD)
Type of service: Shared (4 slots)
Always Online: 9,000 - 10,000
Proxy Type: SOCKS4
De-Duped: Yes
RBL Checked: Yes
Updated every: 5 minutes

References and samples are available.
Contact me via AIM: robinangelic
QUOTE
Done
Adblock


Information About the Bulk Email Marketing Spam Industry - HTTPS & SOCKS PROXIES - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.spamforum.biz/forums/showthread.php?t=250
News To Read Weather MIT Classes Calendar Research Tools Money Mgmt MacDegen Seattle LinkedIn
08-LaMacchia.pdf (application/pdf Obj... (Untitled) Information About the Bulk Email Mark... Information About the Bulk Emai...
If this is your first visit, be sure to check out the FAQ by clicking the link above. You may have to register before you can post: click the register link above to proceed. To start viewing messages, select the forum that you want to visit from the selection below.
POST REPLY
Thread Tools
Search this Thread
Display Modes
HTTPS & SOCKS PROXIES
10-21-2004, 11:58 AM
#1
Bulk MKO is Offline:
Junior Member
Join Date: Jun 2004
Posts: 3
HTTPS & SOCKS PROXIES
2 auto-update url's
1 SOCKS - 1 HTTPS
updated 3-5 times daily
1000-1500 up on each 24/7
$75 USD/week $250 USD/month
You can email bulkmko@yahoo.com if you are interested.
QUOTE
10-21-2004, 12:09 PM
#2
mC_sMiTh is Offline:
Junior Member
Join Date: Sep 2004
Posts: 26
interesting offer for small mailers!
anybody bought peas there before?
Done
PR:n/a
Adblock

Appendix 2—Full STAMP Analysis

1. Computers

Security Constraint	**1-1: Computers must not execute malicious software programs**
Constraint to be Enforced by:	a) Users b) Technical Staff c) Software / Operating System d) Anti-virus software

	Blaster Worm	Sobig Worm
a) Users		
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition • Model Condition	• Model Condition
Degree it was attempted to be enforced - (Goal Condition)	Self managing users did not strive to patch their computers	
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	Self managing users did not understand their computers were vulnerable	Users did not recognize that attachment in Sobig was malicious
Missing Feedback - (Feedback Condition)		
	Blaster Worm	Sobig Worm
b) Technical Staff		
Was the constraint un-enforced and did it contribute to the incident?	Yes	No
Primary reason for lack of enforcement	Action Condition Model Condition	

Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	Technical staff had not installed latest Windows patch, leaving computer vulnerable	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)	Technical staff at some firms may not have recognized that their computers were vulnerable	
	Blaster Worm	Sobig Worm
c) Software / Operating System		
Was the constraint un-enforced and did it contribute to the incident?	Yes	No
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	Security vulnerability in Operating System allowed Blaster to propagate	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		
	Blaster Worm	Sobig Worm
d) Anti-virus software		
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Model Condition	Model Condition
Degree it was attempted to be enforced - (Goal Condition)		

Inadequate Control Actions *- (Action Condition)*		
Model Flaws *- (Model Condition)*	Anti virus software could not protect from Blaster in cases where virus definition was not updated	Anti virus software could not protect from Sobig in cases where virus definition was not updated
Missing Feedback *- (Feedback Condition)*		

Security Constraint	**1-2: If compromised If compromised, computers must not spread virus (firewall, uniformity)**	
Constraint to be Enforced by:	a) Network Devices b) Users c) Technical Staff d) Software / Operating System e) Anti-virus software f) ISPs	
a) Network Devices	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Yes
Primary reason for lack of enforcement	• Goal condition	• Goal condition
Degree it was attempted to be enforced *- (Goal Condition)*	Some firewalls were configured to block tcp port 135, the channel that Blaster traveled on. Others were not.	Most firewalls do not inspect email traffic
Inadequate Control Actions *- (Action Condition)*		
Model Flaws *- (Model Condition)*		

Missing Feedback - (Feedback Condition)		
b) Users	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Various	• Various
Degree it was attempted to be enforced - (Goal Condition)	Spread was facilitated by homogenous computer environment. Most users do not seek a heterogeneous computing environment	Spread was facilitated by homogenous computer environment. Most users do not seek a heterogeneous computing environment
Inadequate Control Actions - (Action Condition)	Once infected, many users had a difficult time removing Blaster from their computer during which time it propagated	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		Most Sobig infected users did not know they were infected
c) Technical Staff	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition	• Goal Condition • Feedback Condition
Degree it was attempted to be enforced - (Goal Condition)	Spread was facilitated by homogenous computer environment. Due to the benefits of homogeneity most do not seek a heterogeneous computing environment	Spread was facilitated by homogenous computer environment. Due to the benefits of homogeneity most do not seek a heterogeneous computing environment

Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		Most administrators of Sobig infected computers did not know they were infected
d) Software / Operating System	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition • Model Condition	• Goal Condition • Model Condition
Degree it was attempted to be enforced - (Goal Condition)	Upon infection the OS did not try to limit the spread of Blaster	Upon infection the OS did not try to limit the spread of Sobig
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	The operating system did not "understand" the propagation routing it was executing came from malicious software	The operating system did not "understand" the propagation routing it was executing came from malicious software
Missing Feedback - (Feedback Condition)		
e) Anti-virus software	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Partially
Primary reason for lack of enforcement	• Model Condition	• Model Condition
Degree it was attempted to be enforced - (Goal Condition)		

Inadequate Control Actions *- (Action Condition)*		
Model Flaws *- (Model Condition)*	In cases where virus signatures were not updated, the AV software did not alert the user and attempt to remove Blaster	In cases where virus signatures were not updated, the AV software did not alert the user and attempt to remove Blaster
Missing Feedback *- (Feedback Condition)*		
f) ISPs	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Yes
Primary reason for lack of enforcement	• Goal Condition	• Model Condition
Degree it was attempted to be enforced *- (Goal Condition)*	Only some ISPs chose to block traffic that seemed to be Blaster related	
Inadequate Control Actions *- (Action Condition)*		
Model Flaws *- (Model Condition)*		ISP's control model did not include the contents of email attachments
Missing Feedback *- (Feedback Condition)*		

2. Network Devices

Security Constraint	**2-1: Devices must be properly configured and maintained to prevent malicious network traffic from accessing computers**	
Constraint to be enforced by:	a) Users b) Technical Staff	
a) Users	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	No
Primary reason for lack of enforcement	• Model Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	Many users were not aware that their devices were improperly configured to handle the Blaster threat	
Missing Feedback - (Feedback Condition)		
b) Technical Staff	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	No
Primary reason for lack of enforcement	• Feedback Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		

Model Flaws *- (Model Condition)*		
Missing Feedback *- (Feedback Condition)*	Many technical staffers were not aware that their devices were improperly configured to handle the Blaster threat	

3. Users

Security Constraint	**3-1: Users must invest time and effort in securing their computers**	
Constraint to be enforced by:	Self enforced constraint	
a) Users (Self Enforced)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition • Model Condition • Feedback Condition	
Degree it was attempted to be enforced - (Goal Condition)	A significant number of users choose not to invest the time and effort to secure their computers	
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	Those that do choose to secure their computers frequently lack the understanding to know how to approach the problem	
Missing Feedback - (Feedback Condition)	Users do not know when their computers are sufficiently protected	

4. Technical Staff

Security Constraint	**4-1: Have sufficient time and training to effectively complete work tasks**	
Constraint to be enforced by:	a) Technical & Company Management	
a) Technical & Company Management	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Partially
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	Corporate technical staffs are generally pulled in many directions and staffed on proactive as well as reactive tasks.	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

5. Technical Management

Security Constraint	**5-1: Must correctly balance security risk with performance pressure**	
Constraint to be enforced by:	Self Enforced	
	Blaster Worm	Sobig Worm
a) Technical & Company Management		
Was the constraint un-enforced and did it contribute to the incident?	Partially	
Primary reason for lack of enforcement	• Model Condition • Feedback Condition	
Degree it was attempted to be enforced - (Goal Condition)	*This is a goal that companies generally attempt to enforce*	
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	It is very difficult to successfully gauge the amount of security risk a company faces.	
Missing Feedback - (Feedback Condition)	It is very difficult to understand the effectiveness of security measures already in place.	

6. Software and Operating Systems

Security Constraint	**6-1: Software must not contain security vulnerabilities**	
Constraint to be enforced by:	Software and Operating System Vendors	
a) Software and Operating System Vendors	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	No
Primary reason for lack of enforcement	Action Condition Feedback Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The controls in place for creating vulnerability free software were not effective	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)	The vendor was not aware of the security vulnerability that was exploited by Blaster.	

7. Software and Operating System Vendors

Security Constraint	**7-1: Vendors must successfully balance pressure to run a profitable business and create secure software**
Constraint to be enforced by:	Software and Operating System Vendors

a) Technical & Company Management (Self Enforced)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	No
Primary reason for lack of enforcement	• Action Condition • Feedback Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	Companies use available information to determine how much to invest in security their products	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)	It is impossible for a vendor to know if any security vulnerabilities remain	

Security Constraint	**7-2: Vendors must release security patches in a timely and prudent manner**
Constraint to be enforced by:	Self Enforced

	Blaster Worm	Sobig Worm

a) Technical & Company Management (Self Enforced)		
Was the constraint un-enforced and did it contribute to the incident?	No	No
Primary reason for lack of enforcement		
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

8. Anti-virus software

Security Constraint	**8-1: Antivirus Software must not be disabled by malicious software**	
Constraint to be enforced by:	Anti-virus and other Prophylactic Software Vendors	
a) Software and Operating System Vendors	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	No	No
Primary reason for lack of enforcement		
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

Security Constraint	**8-2: Antivirus Software must detect and remove malicious software as well as prevent malware from installing itself on an otherwise vulnerable computer**	
Constraint to be enforced by:	Anti-virus and other Prophylactic Software Vendors	
a) Software and Operating System Vendors	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	No	No

Primary reason for lack of enforcement		
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

9. Anti-virus software vendors

Security Constraint	**9-1: Antivirus Software Vendor must provide up to date virus definitions as quickly as possible and automatically deliver them to customers or notify customers about their availability**	
Constraint to be enforced by:	Anti-virus software vendors (Self Enforced)	
a) Anti-virus software vendors (Self Enforced)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Partially
Primary reason for lack of enforcement	• Action Condition	• Action Condition
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	Updated definitions that addressed Blaster were provided very quickly (the same day the worm was discovered in Symantec's case), but any delay in creating and distributing the new signature allowed more computers to be compromised	Updated definitions that addressed Sobig were provided very quickly (one day after the worm was discovered in Symantec's case), but any delay in creating and distributing the new signature allowed more computers to be compromised
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

10. Internet Service Providers (ISPs)

Security Constraint	**10-1: ISPs must exert some degree of control over the computers that they connect to the Internet while keeping customers satisfied**	
Constraint to be enforced by:	Internet Service Providers (Self Enforced)	
a) Internet Service Providers (Self Enforced)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Partially	Partially
Primary reason for lack of enforcement	• Goal Condition	
Degree it was attempted to be enforced - (Goal Condition)	Most ISPs are very hesitant to actively filter packets as it would result in additional costs and perhaps migration of customers to competitive ISPs that are not filtering.	
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

11. Think Tanks / Information Aggregators

Security Constraint	**11-1: Information aggregators must have access to information about Internet threat activity**	
Constraint to be enforced by:	Government—Executive	
a) Government—Executive	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	No	No
Primary reason for lack of enforcement		
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

Security Constraint	**11-2: Must recognize and communicate dangerous trends**	
Constraint to be enforced by:	Government—Executive	
a) Government—Executive	Blaster Worm	Sobig Worm

Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	While some think tanks may have identified that a very homogenous computing environment could be dangers, no action was taken to reverse this.	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

12. Law Enforcement

Security Constraint	**12-1: Must have a impressive and well known record for arresting those responsible for virus outbreaks**
Constraint to be enforced by:	Government—Executive Government—Legislative

	Blaster Worm	Sobig Worm
a) Government—Executive b) Government—Legislative		
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Model Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)	With a few exceptions, law enforcement has had significant troubles apprehending those responsible for virus outbreaks.	
Missing Feedback - (Feedback Condition)		

13. Government--Executive

Security Constraint	**13-1: Must prevent cyber attacks**	
Constraint to be enforced by:	Government – Executive (Self Enforcing)	
a) Government – Executive (Self Enforcing)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The Department of Homeland Security's National Cyber Security Division (NCSD) was faulted by the Inspector General for not providing sufficient leadership for the private sector.[118] The President's Information Technology Advisory Committee published a report in February 2005 saying that the research community was under funded and should have their funding increased by $90 million annually.[119]	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

[118] Lemos, Robert. "Report: Federal cybersecurity effort needs improvement." CNET News.com 23 July 2004.
[119] Cyber Security: A crisis of prioritization (President's Information Technology Advisory Committee) http://www.itrd.gov/pitac/reports/20050301_cybersecurity/cybersecurity.pdf

Security Constraint	**13-2: Must minimize the damage of cyber attacks**
Constraint to be enforced by:	Government – Executive (Self Enforcing)

	Blaster Worm	Sobig Worm
a) Government – Executive (Self Enforcing)		
Was the constraint un-enforced and did it contribute to the incident?		
Primary reason for lack of enforcement		
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

14. Government—Legislative

Security Constraint	**14-1: Must craft laws that have a positive effect on the security of the Internet**	
Constraint to be enforced by:	Government—Legislative (Self Enforcing)	
a) Law Enforcement (Self Enforcing)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition • Action Condition	
Degree it was attempted to be enforced - (Goal Condition)	The US congress has chosen not to enacted law that place responsibility for cyber security on software vendors and on users	
Inadequate Control Actions - (Action Condition)	Even with the current hacking statutes in the US, many viruses are written each year	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

15. Security Researchers

Security Constraint	**15-1: Must develop new technologies that can be used to control malicious software**	
Constraint to be enforced by:	Security Researchers (Self Enforcing)	
a) Security Researchers (Self Enforcing)	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	• Goal Condition	
Degree it was attempted to be enforced - (Goal Condition)	The President's Information Technology Advisory Committee published a report in February 2005 saying that the current rate of technology transfer from research to commercial use was not rapid enough.[120]	
Inadequate Control Actions - (Action Condition)		
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		

[120] Cyber Security: A crisis of prioritization (President's Information Technology Advisory Committee) http://www.itrd.gov/pitac/reports/20050301_cybersecurity/cybersecurity.pdf

16. Malicious software writers

Security Constraint	**16-1: Malicious software programmers must not create and release malicious software**	
Constraint to be enforced by:	Law Enforcement Government—Legislative	
a) Law Enforcement	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The threat of arrest and likelihood of arrest did not sufficiently dissuade virus programmers.	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		
b) Government—Legislative	Blaster Worm	Sobig Worm
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The laws enacted and the penalties expected by the virus writers did not sufficiently dissuade the virus programmers.	

Model Flaws *- (Model Condition)*		
Missing Feedback *- (Feedback Condition)*		

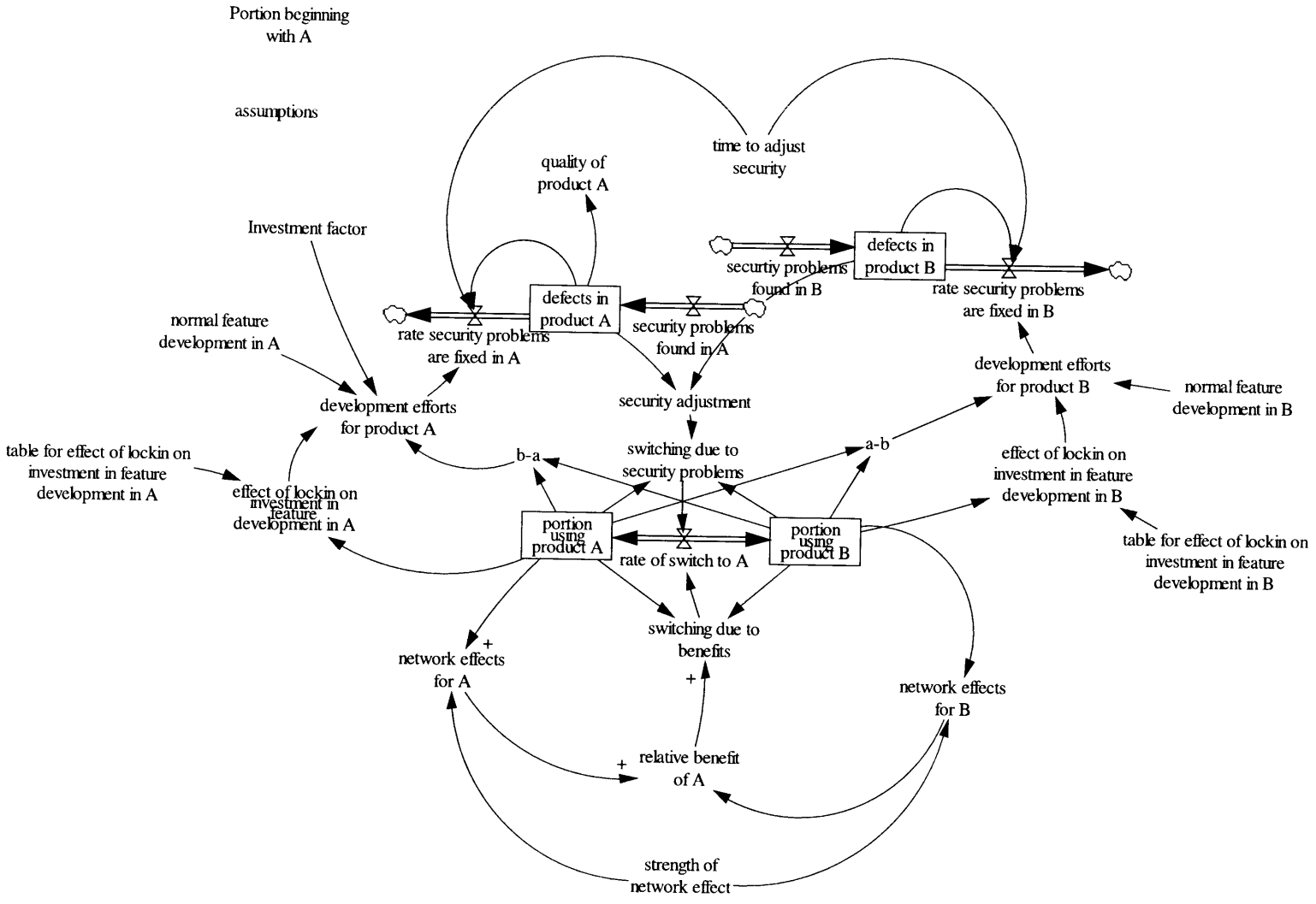
17. Malicious Software Sponsors

Security Constraint	**17-1: Sponsors must not pay hackers to write malware or to gain the use of their compromised computers (i.e., botnets)**
Constraint to be enforced by:	Law Enforcement Government—Legislative

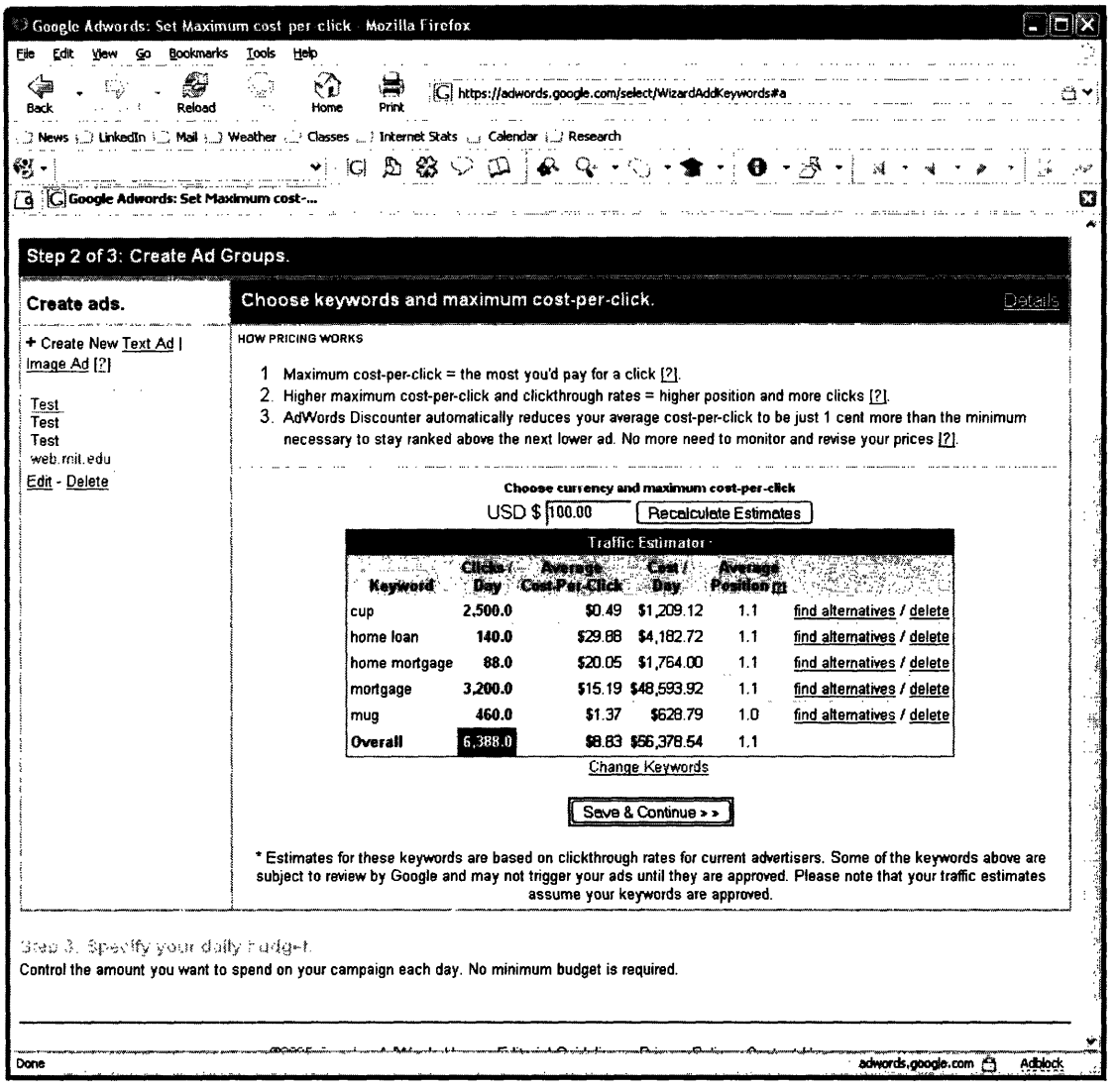
	Blaster Worm	Sobig Worm
a) Law Enforcement		
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The threat of arrest and likelihood of arrest did not sufficiently dissuade virus programmers.	
Model Flaws - (Model Condition)		
Missing Feedback - (Feedback Condition)		
	Blaster Worm	Sobig Worm
b) Government— Legislative		
Was the constraint un-enforced and did it contribute to the incident?	Yes	Yes
Primary reason for lack of enforcement	Action Condition	
Degree it was attempted to be enforced - (Goal Condition)		
Inadequate Control Actions - (Action Condition)	The laws enacted and the penalties expected by the virus writers did not sufficiently dissuade the virus programmers.	

Model Flaws *- (Model Condition)*		
Missing Feedback *- (Feedback Condition)*		

Appendix 3—System Dynamics Model of a Simple Software Industry


Portion beginning with A
assumptions
time to adjust security
quality of product A
Investment factor
securtiy problems found in B
defects in product B
rate security problems are fixed in B
defects in product A
security problems found in A
normal feature development in A
rate security problems are fixed in A
development efforts for product B
normal feature development in B
security adjustment
development efforts for product A
table for effect of lockin on investment in feature development in A
b-a
switching due to security problems
a-b
effect of lockin on investment in feature development in B
effect of lockin on investment in feature development in A
portion using product A
portion using product B
rate of switch to A
table for effect of lockin on investment in feature development in B
switching due to benefits
network effects for A
network effects for B
relative benefit of A
strength of network effect

Appendix 4—Google Keyword Prices


Google Adwords: Set Maximum cost per click - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Back Reload Home Print
https://adwords.google.com/select/WizardAddKeywords#a
News LinkedIn Mail Weather Classes Internet Stats Calendar Research
Google Adwords: Set Maximum cost-...
Step 2 of 3: Create Ad Groups.
Create ads.
Choose keywords and maximum cost-per-click.
Details
+ Create New Text Ad | Image Ad [?]
Test
Test
Test
web.mit.edu
Edit - Delete
HOW PRICING WORKS
1 Maximum cost-per-click = the most you'd pay for a click [?].
2. Higher maximum cost-per-click and clickthrough rates = higher position and more clicks [?].
3. AdWords Discounter automatically reduces your average cost-per-click to be just 1 cent more than the minimum necessary to stay ranked above the next lower ad. No more need to monitor and revise your prices [?].
Choose currency and maximum cost-per-click
USD $ 100.00 Recalculate Estimates
Traffic Estimator
Keyword | Clicks / Day | Average Cost-Per-Click | Cost / Day | Average Position [?]
cup 2,500.0 $0.49 $1,209.12 1.1 find alternatives / delete
home loan 140.0 $29.88 $4,182.72 1.1 find alternatives / delete
home mortgage 88.0 $20.05 $1,764.00 1.1 find alternatives / delete
mortgage 3,200.0 $15.19 $48,593.92 1.1 find alternatives / delete
mug 460.0 $1.37 $628.79 1.0 find alternatives / delete
Overall 6,388.0 $8.83 $56,378.54 1.1
Change Keywords
Save & Continue >>
* Estimates for these keywords are based on clickthrough rates for current advertisers. Some of the keywords above are subject to review by Google and may not trigger your ads until they are approved. Please note that your traffic estimates assume your keywords are approved.
Step 3: Specify your daily budget.
Control the amount you want to spend on your campaign each day. No minimum budget is required.
Done
adwords.google.com
Adblock