Threat Effects Analysis: 
Applying FMEA to Model Computer System Threats

John B. Bowles, University of South Carolina
William Hanczaryk, Winthrop University

Key Words:  FMEA, software FMEA, threat, data flow diagram, malware, computer security

ABSTRACT

As the 21st century progresses, computer systems have become a target for a new type of criminal who attacks software with malicious intent. Failure Modes and Effects Analysis, which is normally used to improve system reliability by identifying and mitigating the effects of potential system failures, provides a basic framework that can be applied to counter the threats a computer system will encounter in its operational environment. The process consists of: 1) becoming familiar with the system and system components; 2) developing a threat model by identifying external dependencies and security assumptions; 3) identifying and classifying the types of threats to the system; 4) determining the effects of the threat; and 5) making changes to counter the potential threats. This approach ensures that the assessment of the threat will be done in a systematic and meticulous manner that is more likely to result in a secure and reliable system.

1. INTRODUCTION

Acronyms

FMEA  Failure Modes and Effects Analysis
TEA  Threat Effects Analysis
CERT  Crisis Emergency Response Team
DFD  Data Flow Diagram
CIA  Confidentiality, Integrity, Availability

Reliability requires that a system “perform adequately under the operating conditions encountered”. For modern computer systems the “conditions encountered” increasingly are a hostile operating environment in which criminals attempt to gain unlawful access to, or modify the information stored on the system, usurp the computer for their own purposes, or in some way make the system unusable for its intended purpose. The threats include a bewildering array of different types of malicious software (“malware”) such as: computer viruses, Internet worms, Trojan horses, logic-bombs, time-bombs, adware, spyware, trackware, and rootkits, as well as methods of attack such as the creation and use of zombies, denial of service attacks, phishing, and spoofing to name only a few.

As shown in Figures 1 and 2 the number of computer viruses and incidents has increased exponentially since the first viruses were introduced in the 1970s [1]. In 2003 the Crisis Emergency Response Team (CERT) at Carnegie-Mellon University recorded 137,529 security incidents [2]. This was almost 37 times the 3734 incidents recorded five years earlier in 1998 and 100 times the number of incidents reported in 1993. An “incident” could involve a single site or thousands of sites; it could also be of short duration or involve ongoing activity for some time. CERT stopped tracking the number of incidents in 2004 because the use of automated tools had made attacks on Internet-connected sites so widespread and commonplace that counting the number of incidents reported was no longer of much value in assessing their impact. To operate reliably in this environment a computer system must be able to thwart these attacks.

CERT has also tracked the number of vulnerabilities reported in software since 1995. Unfortunately, the number of
vulnerabilities cataloged has been growing each year as shown in Figure 3. By 2007, more than 30,780 vulnerabilities had been cataloged [2].

Clearly, the developers of computer software continue to introduce vulnerabilities into the systems they build and the makers of malware have become very adept at taking advantage of those vulnerabilities to achieve their objectives. As software becomes more wide spread and PC-based software is adapted to embedded and mobile systems, problems and failures due to malware can only increase. For example, a computer virus was recently identified in automobile navigation system [3] and cell phones have also become vulnerable.

Failure Modes and Effects Analysis (FMEA) is a formal process that analyzes each item in a system, considers every possible way in which that item can fail, determines how each failure will affect system operation, and ultimately utilizes the results of the analysis to nullify the root causes of the failures or at least alleviate their effects. The basic FMEA methodology, shown in Figure 4, has been successfully applied to electrical, mechanical, and software systems; to products, maintenance procedures, and manufacturing processes; and at the functional, interface, and detailed levels of design. The process, as adapted by Ozarin [4] for software is shown in Table 1. Here we extend the technique to analyze and manage the threats to a computer system’s operation caused by malware rather than explicit component failure modes.

Figure 3 - Number of Vulnerabilities Reported, 1995 – 2006.

The difference between a failure mode analysis and a threat analysis is that a failure mode analysis examines the way a system can fail due to design and operational errors and a threat analysis examines the way an attacker can intentionally make a system fail. To illustrate the difference, consider a commercial airplane. The primary system goal is to transport people safely from one airport to another. A Failure Modes and Effects Analysis of a commercial airplane would investigate the possible ways that the components of the airplane (engine, controls, fuel injector, landing gear, etc.) can fail and develop ways to prevent or mitigate the effects of those failures. A Threat Effects Analysis (TEA) of the airplane would investigate possible ways that a terrorist or other attacker could prevent the airplane from successfully transporting people from one airport to another or divert the airplane to another purpose and develop ways to prevent or mitigate the effects of those attacks.

Here we are most interested in computer systems and the types of threats posed by computer viruses, denial of service attacks, and other malware that may be encountered in a given operational environment, but, as the airplane example illustrates, the same approach can be used to analyze threats to other types of systems.

2. THREAT EFFECTS ANALYSIS

A threat is an expression of intention to inflict evil, injury, or damage. With respect to computer systems, malicious software intentionally attacks software systems to: 1) cause an unintended effect; 2) prevent an intended effect; 3) or do both. Contemporary software designers use threat modeling to better understand and repulse threats to their systems. Threat modeling provides a framework for understanding potential security threats to a system, determining risk, and establishing appropriate mitigations [4]. An abundance of threat models exist today because of the great variety of threats and because there is no universal standard method for modeling threats to software systems.

Following Ozarin and Siracusa (Table 1) the basic steps of a threat analysis to model the malware threats to computer systems and develop techniques to counter these effects are:

- software and system familiarization;
- develop rules and assumptions;
- describe threats;
- classify threats;
- identify threats;
- determine system threat effects; and
- apply changes to resolve or mitigate threats.

To provide a better understanding of the Threat Effects Analysis process, we illustrate the steps of the analysis using a
2.1 Software and System Familiarization

Threat Effects analysis is most successful when performed by an outside analyst with no prior knowledge of decisions and tradeoffs made during system development and no personal bias. This presents the analyst with the intricate task of attempting to understand someone else’s work. Software code will frequently have insufficient comments and misleading variable names [5]. Models assist in familiarization with a system by illustrating the physical and functional relationships between system components and the interfaces within the system. The types of models that are most frequently used for design representations in FMEA are: hierarchical models, structural models, functional models, and reliability logic diagrams [6]. Producing use cases will give analysts a greater understanding of intended functions of a system and broad user scenarios. The primary user scenarios for the Internet Catalog Company example are:

- Unknown user views catalog and product details
- Unknown user searches through catalog
- Unknown user adds item to cart
- Unknown user logs in to become authorized
- Unknown user creates authorized user account
- Authorized user places an order
- Order clerk processes order

Many threats to a system attempt to attack sensitive data, therefore developing Data Flow Diagrams (DFDs) is particularly important when analyzing threats rather than failure modes. Microsoft security expert Michael Howard explains that creating accurate DFDs is critical to their company threat-model process [4]. To achieve the most precise Threat Effects Analysis, the DFD of the highest possible level should be modeled. To produce a more general analysis, only a context level DFD or level-0 model is needed. Figure 5 exhibits a simple context level Data Flow Diagram of the Internet Catalog System that shows how the main system interacts with external entities such as the users of the system and the database. In Figure 5 arrows represent processes or data that is being transferred to or from the system.

2.2 Developing Rules and Assumptions

The next step of the software FMEA process is to create rules and assumptions. Generating rules and documenting assumptions for a Threat Effects Analysis is far different than for software FMEA. Threat Effects Analysis is ultimately concerned with how a system will interact with other systems and less concerned with internally caused failures. The Microsoft threat-modeling process provides a guide for developing rules and assumptions to be emulated by Threat Effects Analysis [3].

2.2.1 Identify External Dependencies. All except the most trivial software systems run on operating systems and almost all cooperate with other systems to achieve a common goal. Documenting every potential interaction with any external system is imperative to identifying how or where an attack could be launched. Some of the external dependencies for the Internet Catalog Company in our example are:

- operating system
- web-based client using an Internet browser
- database Server(s)
- network Infrastructure

2.2.2 Define Security Assumptions. Security assumptions are the guarantees expected from the external dependencies. An example is assuming that a database server interacting with a system will adequately protect sensitive information by using appropriate authorization technology or encryption. Defining security assumptions is essential because it specifies the required behavior of the external entities that the system depends on, but may not be able to control. Some appropriate security assumptions for the Internet Catalog Company example are:

- the database keeps authentication information
- the database server protects authentication data by using authorization technology or encryption
- sensitive data will be sent over SSL/TLS connections
- only valid administrators will have physical access to the Web and database servers

![Figure 5 - Data Flow Diagram for Internet Company Example.](image-url)
2.3 Developing Descriptive Threat Reports

A Failure Mode is a situation that can: 1) cause an unintended effect; 2) prevent an intended effect; or 3) both [5]. In analyzing Failure Modes, the most important elements to discover are the cause and the effect of the failure. Similarly a key component of Threat Effects Analysis is identifying the source and forecasting the effects of potential threats. Implementing a formal process to identify threats in descriptive detail is the core procedure of Threat Effects Analysis and is necessary to finding the effects of threats upon the system and discovering the best way to resolve their consequences.

2.4. Classifying Threats

The most important procedure of Threat Effects Analysis is to develop a system to classify and group together similar system threats. In this way classes of countermeasures for known types of threats can be developed. At present a universal taxonomy of malicious security threats does not exist in the computer industry, however some useful taxonomies have been developed. Perhaps the most well known taxonomy is CIA which addresses the three key requirements to security in any computer-related system: confidentiality, integrity, and availability [7]. Several threat classification strategies utilize CIA to classify threats into groups. One such strategy by Bierman and Cloete [8] categorizes threats first into threat classes (Confidentiality, Integrity, Availability, Authentication, and Non-repudiation) and then into sub-classes based on techniques that a threat would use to breach those classes. The strategy then uses the defined subclasses to introduce sets of appropriate solutions to alleviate those types of threats. Although originally developed for threats in Mobile Agent Computing, the Bierman and Cloete classification strategy can be applied to classify threats to almost all software systems. We will utilize only the Confidentiality, Integrity, and Availability classes to classify threats and examine in detail how Bierman and Cloete break these classes into further sub-classes.

2.4.1 Confidentiality Threats. Confidentiality threats attempt to obtain resources of a system that are intended to be accessed only by authorized parties. The typical goal of a confidentiality threat is for an unauthorized entity to acquire secret or private information from a system. Bierman and Cloete divide Confidentiality threats into three subclasses [8]:

- **Eavesdropping** – Threats in which privacy is invaded when an attacker spies on a user of a system or the system itself to gather information about that user or about intercommunication between the user and the system. Even though the threat may not directly affect the user, it may illegally use the information for personal gain.

- **Theft** – Subclass of threats that looks to not only spy on a user or system like Eavesdropping does, but also to steal information from the user or system and use it for personal gain.

- **Reverse Engineering** – Threats that essentially obtain access to a software entity’s information, analyze it, and use it to create a new similar entity to gain access to a system.

2.4.2. Integrity Threats. Integrity threats involve an unauthorized entity attempting to modify data and other system assets that were intended to be modified only by authorized parties. This can include writing, changing, changing status, deleting, and creating system information [7]. Bierman divides Integrity threats into two separate subclasses [8]:

- **Integrity Interference** – Threats in which an attacker interferes with communication between a user and a system to obstruct the user’s execution of a system function. Interference does not directly alter any information but can result in a system function not operating is it was meant to.

- **Information Modification** – Threats that occur when an attacker interferes with interaction of a system and alters system communication for personal benefits. The goal of this type of threat is to alter, corrupt, manipulate, or delete system assets such as code or data.

2.4.3. Availability Threats. Availability is the probability that a software system is operating successfully according to specification at a given point in time [9]. Availability threats include any malicious attack that attempts to halt a system from completing required behaviors. Bierman and Cloete divide availability threats into three subclasses [8]:

- **Denial of Service (DOS)** – Threats that attempt to make a system crash or become functionally inoperative as a result of being flooded with network traffic. DOS occurs anytime a malicious host attacks a system with so much extraneous material that the system cannot function properly.

- **Delay of Service** – Threats that permit a system to function but only after a certain amount of time has passed, which can sometimes defeat the purpose of particular system functions.

- **Transmission Refusal** – Threats in which an attack interrupts the flow of data within a system and simply disallows the system from transmitting information any further.

Table 2 summarizes the classification taxonomy provided by Bierman and Cloete. This classification scheme was originally intended to classify threats in Mobile Agent Computing but as we have shown, it can be applied more
2.5  Identifying Threats

Although the ultimate goal of Threat Effects Analysis is to identify and eliminate system threats, the actual identifying part of the process is the least important part of the procedure. Threat Effects Analysis utilizes a combination of the three previously mentioned processes (developing rules and assumptions, describing threats, and classifying threats) to identify potential threats and threat targets to a system. To identify potential threats, an analyst must consider: 1) Data Flow Diagrams (DFD) and other models that help build familiarity with the system; 2) the list of external dependencies and security assumptions; and 3) the established classification of threats. The identifying part of Threat Effects Analysis is not an exact science. A threat analyst must take into account the three aforementioned processes to determine how and where potential threats could possibly occur. Once established, a threat should then be added to the table in the appropriate class and subclass. The threat should be well enough defined to determine what type of threat it is, how it could occur, and where it could attack.

Table 3 shows some of the threats for the Internet Catalog Company using the classification scheme in Table 2.

2.6 Determine System Effects of Individual Threats

Once potential threats are put into their appropriate classes and sub-classes as in Section 2.4, determining the system effects becomes trivial. The main attribute that the classification scheme is based on is the subsequent effects that the threat will have on the system. Consequently, little analysis is necessary at this step. Once a threat is classified, its potential effects upon a system are usually apparent. These effects should then be inserted into a new column of the table adjacent to the particular threat being analyzed.

2.7 Resolve Threats or Mitigate Their Effects

For each threat in a Threat Effects Analysis, we can define one or more mitigation aspects by applying appropriate assurance techniques or countermeasures [10]. Mitigations can range from doing nothing, to warning the user, to removing a feature if it cannot be made secure. But the mitigation technique that we are most concerned with is using additional technology to counteract the threat. The classification of threats in Section 2.4 allows us to easily apply countermeasures that will thwart the effects of classes and sub-classes of threats. A comprehensive list of mitigation techniques and the best ways of choosing appropriate mitigations is beyond the scope of this paper but can be researched from a multiple of sources [8, 11, 12, 13]. The next stage in Threat Effects Analysis is to search for mitigation techniques for all the identified threats and subclasses they are within. Then choose the appropriate mitigation technology for each threat and insert those mitigations into a new column of the table. Several mitigation technologies have been listed in the last column of Table 3 that act to offset the specific threats to the Internet Catalog Company example.

3. CONCLUSION

Modern computer systems operate in increasingly hostile environments in which criminals attempt to gain unlawful access to sensitive information or use the system for their own rather than other than its intended purpose. The interconnected computing world of the 21st century is highly susceptible to dangerous threats that intensify risk for all software users. Consequently, creating more secure software is extremely significant. Using an FMEA style approach to model threats to a system provides an effective framework for assessing threats to the operation of a computer system. The systematic process dubbed “Threat Effects Analysis” analyzes the system, classifies threats, identifies threats, determines threat effects, and applies changes to minimize risk. A methodical process such as this makes certain that evaluating threats will be done in a systematic and thorough manner, which is more likely to result in a secure and reliable system.

ACKNOWLEDGEMENTS

This work was done at the University of South Carolina as part of the Research Experiences for Undergraduates in Multidisciplinary Computing project and supported in part by National Science Foundation Award # 0353637.

REFERENCES

<table>
<thead>
<tr>
<th>Threat Class (2.4)</th>
<th>Threat Subclass (2.4)</th>
<th>Threats (2.5)</th>
<th>Effects (2.6)</th>
<th>Countermeasures (2.7)</th>
</tr>
</thead>
</table>
| Confidentiality Attack | Eavesdropping | Network eavesdropping between browser and Web server | • Customer information and credit history is not secure | 1) Encryption  
• Sensitive data is no longer private | 2) Access Control Lists |
| | | Attacker acquires encryption keys used to encrypt sensitive data | • Customer Information and credit history is not secure  
• Unauthorized parties gain access to private data | 1) Access Control Lists |
| | Theft | Attacker captures an authentication cookie and spoofs a customer identity | • Unauthorized user obtains an existing customer’s user account and all details  
• Sensitive customer information is stolen and used | 1) Encryption  
2) Access Control Lists  
3) Trusted Execution Environment |
| | Reverse Engineering | N/A | | |
| Integrity Attack | Integrity Interference | Attacker injecting malicious script code by cross-site scripting | • System does not perform intended functions properly  
• System performs tasks it was not intended to. | 1) Access Control Lists  
2) Digital Signatures  
3) Trusted Execution Environment |
| | Information Modification | SQL injection where an attacker exploits vulnerabilities in the database to execute commands | Sensitive data is accessed and modified. | 1) Detection Objects  
2) Digital Signatures  
3) Encryption  
4) Access Control Lists  
5) Message Authentication Codes |
| Availability Attack | DOS (Denial of Service) | Attacker manages to take control of the Web server and run commands against the system’s will (Can be used to carry out all three availability attacks) | • System crash  
• Network traffic  
• System cannot function | 1) Authorization  
2) Access Control Lists  
3) Filtering  
4) Trusted Execution Environment (Suitable countermeasures for all availability attacks) |
| | Delay of Service | | System functions but only after a certain amount of time | |
| | Transmission Refusal | | • Data flow is interrupted  
Redirection of information | |

Table 3 - Threat Effects Analysis Table


BIOGRAPHIES

John B. Bowles, Ph.D.
Computer Science and Engineering
University of South Carolina
Columbia, SC 29208 USA
e-mail: bowles@engr.sc.edu

John Bowles is an Associate Professor in the Computer Science and Engineering Department at the University of South Carolina where he teaches and does research in reliable system design. Previously he was employed by NCR Corporation and Bell Laboratories. He has a BS in Engineering Science from the University of Virginia, an MS in Applied Mathematics from the University of Michigan, and a Ph.D. in Computer Science from Rutgers University. Dr. Bowles is a senior member of IEEE and ASQ, and an ASQ Certified Reliability Engineer.

William P. Hanczaryk
Winthrop University
Rock Hill, SC 29733 USA
e-mail: hanczarykw2@winthrop.edu

William Hanczaryk is a senior computer science student at Winthrop University in Rock Hill, SC. He will graduate with a bachelor’s degree in Computer Science in May 2008. He serves as the president of the Winthrop University student chapter of the Association of Computing Machinery (ACM). In addition to his interest in network security, Mr. Hanczaryk is working on another research project at Winthrop University that deals with the analysis of trouble ticket logs to identify self-inflicted problems. William is originally from Crofton, MD.