SECURITY INNOVATION

The Art of Threat Modeling for IT Risk Management:

Solving the Software Application Risk Riddle

Introduction

The modern enterprise runs on software that is primarily provisioned and maintained by IT. The software ecosystem is dynamic, complex and hostile. As a result, IT security is ultimately a software problem—more so than at any other layer in the information system infrastructure.

Many organizations use network or perimeter security technologies to mitigate risk. Although these controls are tangible, keep a lot of "bad guys" out, and can minimize impact, they are limited in their ability to prevent software-based attacks. Network defenses must let software perform its functions—and the network security solutions usually don't know if the software has been compromised. Compounding this issue is the fact that applications access data unencrypted. Even if data is encrypted in the database, vulnerabilities in software afford an attacker the same privileges granted to the application itself and access to that same unencrypted data.

The need to understand risks from application vulnerabilities is critical and you must understand this risk holistically—in the context of your entire information management infrastructure. In order to understand how to assess risk in the application layer, the concept of threat modeling has tremendous utility.

Threat modeling is a powerful but under-utilized exercise that can help in risk determination. We will discuss two approaches to threat modeling—both need to follow a proven methodology for effectiveness:

1. Threat modeling of an existing application
2. Threat modeling during the various phases of the software development lifecycle

Whether threat modeling is performed on an existing application or throughout the software development lifecycle, it is an essential component in the risk management arsenal because it can help quantify and visualize the otherwise intangible threats an application may carry. Threat modeling is not a trivial exercise and should be done with diligence and precision so as not to miss any aspect of the attack surface applications expose.

> ***The goal of this paper is not a "how-to" for threat modeling; there are many good references for that. This paper is specifically designed for IT Risk Management, Information Security, and Management personnel that seek a more effective way to identify and prioritize risk. It is a description of the activities involved in application threat modeling and the goal of threat modeling in the context of IT risk management.***

Situation Analysis

The Need for Threat Modeling

Threat modeling is a powerful technique that helps to characterize higher level threats and separate them into more manageable sub-threats that can be addressed.

From a business perspective, IT security exists only in the context of risk management. Large corporations are driven by risk management concerns including the risk of non-compliance, the risk of data loss, and the risk of financial loss through IT theft, legal sanctions, and infrastructure downtime.

Although vulnerabilities, hackers, and exploits are compelling reasons to focus on application security, they have been overshadowed by compliance and risk issues in the minds of corporate decision makers. The myriad of regulations and standards are imposing stricter IT security requirements and application security has reached a pinnacle of importance in the context of regulatory compliance.

In an ideal world with limitless resources and budgets, all applications would undergo in-depth assessments and all stakeholders would be trained on how to design, code, and operate software securely. This is why Threat modeling is such a powerful exercise—it can ensure your most critical threats are prioritized and can be conducted on a single application, on a plethora of applications, and at every phase of the Software Development Lifecycle (SDLC).

Risk management is a key requirement of many of these regulations and is one of the most difficult processes to conduct and complete. The difficulty lies in the fact that although the high-level threats are generally well understood (breach of customer data, denial of service etc.), the underlying causes and sub-threats that can lead to each remain obscure.

The perimeter defense concept is fairly easy to grasp—keep the "bad guys" out. However, perimeter defenses also facilitate remote user access to corporate resources; thus, the distinction between trusted and distrusted users is increasingly unclear. The advent of cloud computing and the mass migration from on-premises data center to the cloud has further complicated this issue. IT divisions must remember that once a user passes perimeter defenses, the burden of security lies with the applications that process data.

Developers, maintainers, and users of software applications need to understand the risks those applications impose. The higher-level threats are generally well-understood because they are frequently mentioned in regulatory and educational texts, e.g., inappropriate disclosure of sensitive data; however, it is nearly impossible to address these threats without considering the underlying sub-threats that can (and often do) lead to catastrophe.

Risk management is a balancing act:

- Weight of threat realization and impact
- Weight of countermeasures

IT risk has now become an integral part of the operational risk and a headache for risk managers.

Factors that make IT risk assessment difficult include:

- Software internals (especially 3rd-party) remain a bit mysterious
- Attacker techniques seem like black magic
- The impact of software vulnerabilities isn't clear

Threat Modeling Overcomes Common Pitfalls

Risk Management is not simple and when application security is involved it can be complicated and filled with uncertainty. Today's enterprise depends on 3rd-party software more than ever (open-source, COTS, GitHub, etc.) This means you have less control of source code and need to rely on techniques such as threat modeling to qualify software-based risks.

Penetration testing vs. Automated tools

The number of applications used by any given company makes the cost of penetration testing ALL applications prohibitive. Automated tools and services are expensive, inaccurate, and slow.

- Results need to be interpreted by security professionals
- Not easily deployable for large scale applications
- Fail to understand the business logic of applications

In the end IT risk is misevaluated

- Money is spent on unneeded countermeasures
- Real threats are not recognized
- Already mitigated threats may be flagged as serious, wasting valuable time evaluating them, e.g., a SQL injection flaw in code that is mitigated by a WAF in deployment.

Threat modeling assists the risk management process by helping you through common problems, such as:

1. ### Too many applications, too little time
   Many risk management and IT audit teams have hundreds or even thousands of applications to assess for risk and vulnerabilities. This is an almost impossible task without something like threat modeling.

2. ### Viewing applications in isolation
   There may be a highly insecure application that is currently risk-ranked low because you don't realize how vulnerable it is to attack given its shared resources and/or infrastructure. Conversely, you may have an application that is risk-ranked high when it is extremely secure because of compensating controls you already have in place. Often, companies have no idea which applications bring the most risk because they have been assessed in a haphazard or inaccurate manner.

3. ### Knee jerk reactions
   Organizations often make investments in a reactive manner, e.g., an incident has occurred that causes fear and security investments are driven by this fear. The risk here is two-fold:

   - You make an investment that delivers little protection in return (emotional decisions are dangerous ones)
   - You're likely to ignore an area of higher risk because you're deploying limited resources in the wrong place

   You need to understand where your greatest risks are first, so you can make informed security spend decisions.

Threat Modeling for Better Risk Management

Effective risk management for application security is based on the following key principles:

- Knowing which applications introduce the greatest risk
- Calibrating breadth and depth of assessment to application risk
- Understanding how good your teams are when it comes to security
- Knowing what to do with security assessment results

Performing security assessments improperly is costly. It is also difficult to determine whether your teams are improving or repeating their mistakes. Finally, it is not always clear what your repercussion steps are. For example, if an application were developed by an outsourced partner or purchased "off the shelf", what recourse do you have to ask the software vendor or partner to fix the issues? Must you accept the risk? Do you know how to assess the extent of those risks?

Threat modeling provides a map of risk areas in software applications quickly; and the maps created are re-usable assets. Thus, threat modeling can help in two important areas of risk management:

1 Identifying important application criteria that can become enterprise risks

- Technologies
- Bad/Good practices
- Entry points
- Users
- Countermeasures
- Assets

2 Prioritizing countermeasures

Understanding what threat modeling can't do is just as important as realizing its benefits. It cannot identify all vulnerabilities that can become threats in an application and is not a replacement for detailed security assessments.
Specifically, threat models are not:

- A replacement for more detailed and technical security assessments
- A representation of how an attacker approaches a system—it represents total system security, not an attacker model
- A test plan—a test plan is driven by a threat model, but threat models offer a lot more than just test planning
- A formal proof of system security—this would be dangerous
- A design review—threat models are the foundation of design reviews, but a design review needs to cover more implementation details and considerations beyond security

Threat modeling of existing applications can be performed by security professionals who perform exploratory security testing to understand the design and implementation of the application as well as identify the most obvious security failures. This results in intelligence that can be used to modify an application's risk rating and implement additional risk controls (i.e. penetration tests, replace/upgrade application, etc.).

Threat modeling throughout the Software Development Lifecycle will help define the proper security requirements at design and deployment phases, facilitate the correct technology decisions at the implementation stage, and lay the foundation for thorough security test plans by defining negative test cases that will ensure the application correctly handles abuse. This is where the biggest return is observed because threat modeling throughout the SDLC will help minimize the cost of developing secure software.

Threat Modeling Existing Applications

For many organizations, security is still an afterthought. In this model, the priority is software functionality where security is bolted on in the later stages of the software development lifecycle (SDLC). This is also the relevant model for software users who wonder about potential security failures in the products they have purchased.

The key areas threat modeling assists with are in Risk Assessment and Risk Mitigation:

Risk Assessment

- **System characterization**: Define system boundaries, hardware, software, interfaces, data, etc.
- **Threat identification**: Hackers, terrorists, industrial espionage, insiders
- **Vulnerability identification**: Design flaws, coding mistakes, improper configurations

Risk Mitigation

Threat modeling facilitates the decision to accept, transfer, avoid, or reduce risk with respect to the application's security assessment because (a) it was created specifically for software applications, and (b) it relates directly to risk management in approach and business-related risk identification.

A common approach for threat modeling existing applications is a three-step process consisting of: (1) analyzing the application, (2) determining threats; and (3) rating the threats.

1 Analyzing the Application

Consists of identifying the application's features and user/attacker entry points. During this process it is important to note feature characteristics such as access level required to perform related tasks. System or component diagrams can help here, as can sequence diagrams that visualize how actors interact with your application.

2 Determining Threats

This is the most challenging aspect of threat modeling. This step consists of breaking down the higher-level threat into sub-threats that can be more easily addressed. Various models can be used to categorize threats.

- The STRIDE[1] model can be used as a starting point for the characterization and identification of high-level threats. An example of a high-level threat would be when a malicious user escalates privileges. You must identify failure conditions (the sub-threats) that would lead to the realization of this threat. One way to gain elevated privileges is for an unauthorized user to bypass the authentication mechanism; another way is for an attacker to steal credentials from a privileged user. There can be many more ways this attack could be conducted, but for the sake of simplicity we'll focus on these two. Again, sequence diagrams can be useful here when you model numerous threats.
- Bypassing authentication may seem like an unrealistic attack vector, but we often find alternate entry points that can be used to circumvent authentication mechanisms. Intercepting another user's credentials during transport or because of improper storage is another all too common way of elevating privilege.

3 Rating the Threats

This step occurs after all the threats and sub-threats have been identified. One such technique is feature rating. Here you score application features according to their characteristics (is it a new feature with the current release? is it a security feature? is it installed by default? etc.). You then assign each feature a score that is a combined measure of the likelihood it will contain a security flaw and its potential for damage. This feature score allows ranking of each identified threat by averaging the scores for features that are relevant to the threat.

Once broken down, the threats listed above are much more easily addressed when it comes to risk mitigation than the higher level threat. Bypassing authentication can be addressed by ensuring that all access routes are locked down. Theft of credentials can be prevented by encrypting credentials during transport and storage. The threat ranking can be used to prioritize risk mitigation, and the completed threat model can drive application security testing. A completed threat model should provide the supporting material to prioritize risk mitigation and the framework for security testing.

Threat Modeling in the SDLC

For organizations that want to integrate security at each phase of development and ensure key threats are mitigated, threat modeling can be used to its full potential. Leveraged as such, it will help guide the development process and provide a framework for developing secure code, security test plans, and proper deployment configurations. In a DevOps model, creation of a robust threat model as far "left" (early) as possible is paramount. This threat model can be iterated as the DevOps cycle moves from point to point.

The process by which threats are characterized and ranked can be similar to the one described previously, however the threat model will grow and be refined as the product progresses through its lifecycle. For example:

- At the requirements phase, authentication using username/password may be rejected or augmented with 2FA because of attack vector(s) it creates.
- At the development phase, encoding user input and output may be deemed necessary to avoid reflective cross-site scripting.
- At the verification phase, new security tests (attacks) will be devised based on abuse cases to validate the inability to steal sensitive data.

Getting Started with Threat Modeling

The most important prerequisite to threat modeling an application is to understand what the application does and how it does it. This will help identify the assets that it uses and what needs to be protected (software composition analysis may be necessary to understand today's complex and 3rd-party dependent software applications). You need to understand the business purpose of the application, its architecture and components (a system or component diagram is most useful here), and the assets that the application touches or manages. This will help you identify actors (both intended and unintended) as well as the high-level threats facing the application. When identifying potential attackers, don't dismiss the most damaging class of unintended users: the insider.

Once the application assets and potential attackers are identified, you can derive the high-level threats to the application. These may include denial of service to system components, data theft, data corruption, and so on.

Identifying Entry Points

From a network perspective, entry points are identified as those external or internal routes that ultimately lead to an application running on a server or virtual environment. Identifying entry points from this perspective is fairly straightforward. Assuming the proper network protections are in place and that the application hosts are fully patched, there is an important residual risk that comes from the applications themselves.

Identifying the entry points for an application is not always simple because you have to consider the environment in which the application operates. Applications may handle input from the operating system, user interfaces, local or external databases, other applications, APIs, and shared components. One common mistake is to assume that all attack entry points reside in the user interface.

Identifying Realization Conditions

One of the most challenging and critical aspects of threat modeling is identifying the conditions in which the high-level threats can be realized. However, just knowing the high-level threats does not necessarily help prioritize risk mitigation efforts—many applications will have the same high-level threats (e.g. denial of service, database corruption, data theft and so on). The real benefit is understanding the likelihood of the threat being realized and which mitigating controls you need to deploy for your specific business application.

To do this, you must examine the many aspects of an application including the development language, network protocols used, use of cryptography and so on. Other factors to consider when gauging the likelihood of an attacker reaching the next step include existing controls (internal or external), auditing and logging, the use of coding "best practices," attacker techniques and attacker profile.

An application that resides behind a corporate firewall, for example, that is available only through the intranet can be considered relatively safe from external attacks ((assuming the firewall is configured properly of course :-)). This would mean that we can assume the attacker profile is 95% internal and 5% external because some other outward facing application could be misconfigured or vulnerable and could give an external attacker access to the corporate intranet.

Other things come into play as well, such as your knowledge of the application's history. Was the authentication feature built-in to the application from the start or was it added at a later point? Every interface provides another attack surface; therefore, multi-layered applications with legacy code on the back end and newer code in subsequent layers, (i.e. transaction server, application server, web/EJB server, user interface layers, etc.) are particularly difficult to assess because there are so many attack points to consider.

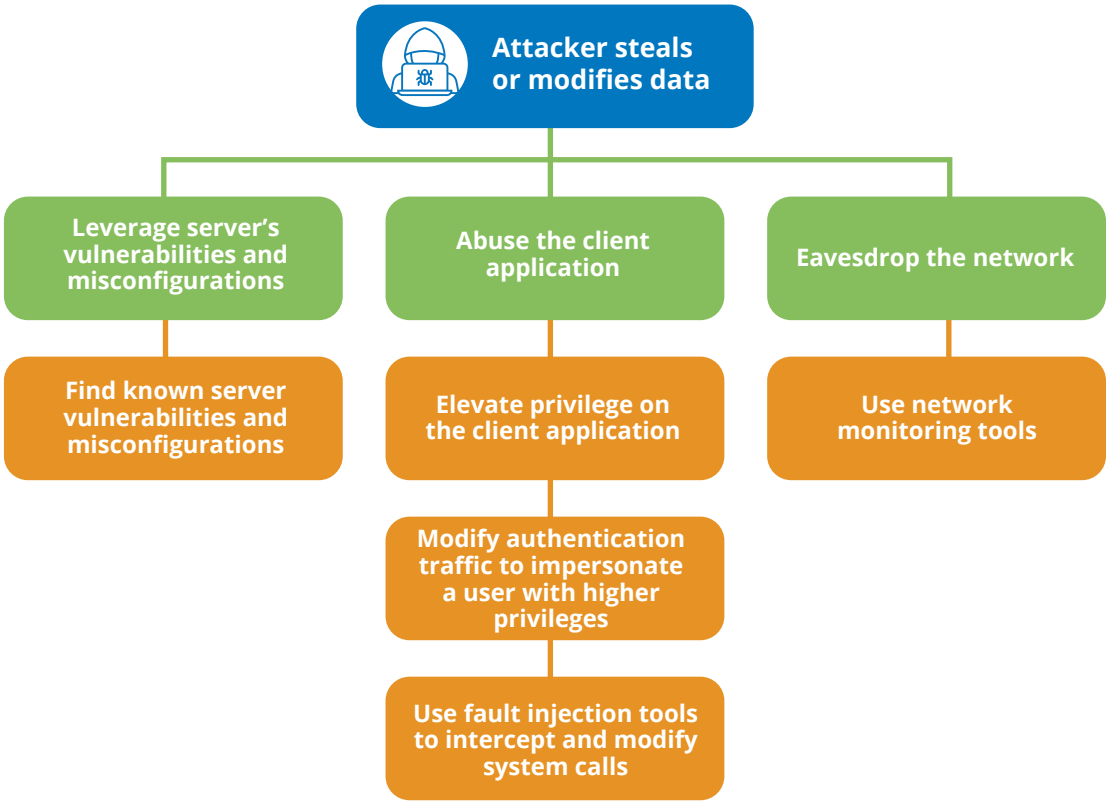
Once you have developed an understanding of attack surfaces, you need to determine realization conditions (the damage potential of each attack.) Damage potential can be assessed by combining users' access level (anonymous external attacker, internal user, admin, etc.) with the attack path, the likelihood of occurrence and the criticality of the data/system at risk.

Risk realization example: Data theft/modification from an external threat

Assume that there is a remotely accessible client which authenticates the user with a username-password combination and provides access to a back-end database server. The user in this scenario provides a username and password hash (via SSL) and receives certain permissions as a result. There are several possible ways data can be compromised, for example:

1. By abusing the database and/or server vulnerabilities and misconfiguration
2. By eavesdropping the network using a sniffer
3. By abusing the client application

For these attack scenarios, consider the following to plot how an attacker could obtain admin username and permissions with the attacker's password hash by modifying intercepted traffic and impersonating a user with higher privilege:



Mapping your Findings into your Risk Management Framework

Risk Management is a discipline, but it is also a balancing act of constantly weighting threat realization and impact, countermeasures and controls. Risk Management frameworks are constantly refined to identify, assess, and predict known attack scenarios, uncover new ones, and help security teams decide where to best invest funds to protect assets. **Threat modeling fits into existing risk management frameworks by providing more efficient risk management in the realm of information systems and technology.**

The biggest risk management challenge is determining risks in applications and information systems. When you can identify system components and attack surfaces of applications, you are more easily able to discuss threats in an informed manner and predict possible attack scenarios.

More importantly, threat modeling:

- Helps you determine attack possibility and success probability, damage potential, and the effect of various mitigating controls. This is often a risk management challenge.
- Allows you to emulate different compensating controls before deployment.
- Yields a persistent asset you can update and leverage as new threats are realized. A simple change to the model will tell you if you already have a mitigation control for the new threat.

Communicating Critical Issues to Development and IT Teams in their language

The language of risk, assessment, mitigation, and compensating controls is a foreign one to most application development teams; however, threat modeling provides a convenient bridge. Threat modeling results can be easily communicated to development teams, outsourced partners, and or vendors because they are expressed (often visually) in terms that those teams understand, specifically:

- User roles and privileges
- System components/technology in scope of various threats
- Attack scenarios that can be described in plain language

Therefore, mitigating controls are also easily identified and discussed between IT Risk and Development teams. This is especially critical in a DevOps environment where frequent cross-team communication is paramount to success. A risk management team can express and verify existing controls in terms of attacks on specific system components or technology implementations; conversely, development teams can implement additional or new controls in an informed manner because they understand the risk and attack scenarios being communicated.

Summary

Threat modeling is a powerful visualization and communication tool for risk management and development teams that can assist in many aspects of risk management and software construction. For risk management teams, threat modeling yields rapid risk assessments of software applications with system characterization in risk management terms. Further, it provides threat and vulnerability identification in a language that is instantly understandable to development organizations. Identification of technology controls that are suited to combat the threats identified also assist risk management teams with risk mitigation and provide a common ground for both teams to discuss and agree upon best practices for secure coding, deployment configurations, and application assessment.

Sources: [1]STRIDE: Spoofing identity, Tampering of data, Repudiation, Information Disclosure, Denial of Service, Elevation of privilege. Howard, Michael and David LeBlanc, Writing Secure Code, Second Edition, Redmond, WA: Microsoft Press