1

THREAT MODELING OVERVIEW

DEFINITIONS

> [Application] Threat Modeling – *a **strategic process** aimed at considering possible **attack** scenarios and **vulnerabilities** within a proposed or existing **application environment** for the purpose of clearly identifying **risk** and **impact** levels.*

Definitions for any type of terminology are necessary evils. While seemingly elementary and potentially annoying, they provide a common ground from which to build. Providing a well-constructed definition also level-sets threat modeling's intended design as a process-oriented control for application security, versus interpretations that mutate its intent and true capability.

In this book, the expression "threat modeling" is reserved for software development and application security efforts. Within the topical boundaries of application security, the aforementioned definition provides some fundamental terms that should resonate with anyone who understands the very nature of security risk management and has implemented the threat modeling machine.

A closer examination of the definition provided reveals greater insights into the essential components that are threat modeling. The first emphasized term, *strategic*, describes a quality of threat modeling reflected in its ability to anticipate threats via

Risk Centric Threat Modeling: Process for Attack Simulation and Threat Analysis, First Edition.
Tony UcedaVélez and Marco M. Morana.

calculated and simulated attack patterns. Each major function within the threat modeling process requires a great deal of consideration and anticipation of multiple risk factors influenced by threat, vulnerability, and impact levels.

Process is one of threat modeling's key, distinguishing qualities. A chain-like reaction of tactical events is conducted across multiple domains (business objectives, system/database administration, vulnerability management, etc.) where additional review, input, and contribution is provided by other stakeholders within the process – all in relation to a protected application environment. To date, the lack of process within information security efforts has accounted for several shortcomings in mitigating security risks introduced by deficiencies in application security, and in many cases acted as causal factors to those noted deficiencies. Although there are isolated victories in traditional security efforts, a growing sentiment is that the war against software exploitation is being lost. Threat modeling is intended to greatly revitalize the effort in securing data via a collaborative, strategic process.

The next term, *attack*, reflects a major science to threat modeling – the discipline of researching how attack patterns can potentially exploit software vulnerabilities and/or poorly designed countermeasures. The hierarchy of an attack becomes dissected via threat modeling techniques, exposing faults in application design and/or software development, as well as other practical yet key areas, such as unveiling plausible motives for which an attacker initially sought to launch their assault.

Vulnerabilities is a term used far more prevalently within other information security efforts. In the scope of threat modeling, however, its use extends the manner in which software vulnerabilities are understood. Vulnerabilities at the platform and software levels are aggregated and correlated to possible attack scenarios. As a result, this term is an essential component to its definition, as we will see in later chapters.

The *application environment* expression serves as the object of the threat modeling process. Other traditional security procedures simply address a single aspect of an entire application environment, thereby negating a more holistic approach to application security. This is not to state that these more isolated procedures are not important, but rather that the sum of their individual benefits is encompassed in the process of threat modeling and applied to the entire application environment.

The term *risk* serves as the object of key interest to threat modeling. Threat modeling, as a supportive role in fulfilling business objectives, seeks to identify risks associated with the cumulative effects of an ever-evolving threat environment, compounded by software/network vulnerabilities, and fueled by attack motives or interest in business information – all managed and/or driven by an application environment. Threat modeling provides greater precision in conveying risk through providing a clear path on how a business application environment could be compromised and the probability of the actual risk. In essence, risk becomes the common glue that unifies security and business professionals in a collaborative effort to protect the enterprise.

Within the threat modeling definition, *impact* is the ability to answer the question "How bad is it?" Unless security professionals consider all possible threat scenarios in order to generate a prioritized, risk-based analysis, they cannot provide an effective and credible answer. As answers morph into speculations and continue downhill, security professionals are again unable to convey an adequate and plausible answer

to this question. Threat modeling divides a threat into multiple attacks, making it easier to see how each attack scenario unfolds. For each scenario, impact of any adverse aftermath can be ascertained with greater accuracy, thereby reestablishing the credibility of the security analysis. The ability to understand impact is central to reporting a threat. Devoid of this capability, identifying and communicating threats merely becomes an exercise built around hype and fear factor.

ORIGINS AND USE

> *It is only one who is thoroughly acquainted with the evils of war that can thoroughly understand the profitable way of carrying it on.*
>
> *Sun Tzu, Art of War*

Despite its trite and oversensationalized use in numerous other security publications, Sun Tzu's quotation is still very relevant to application threat modeling, particularly in its goal to imagine attack scenarios from possible adversaries. Although we are focusing on threat modeling as it applies to software development and application security efforts, we must also consider the origins of threat modeling and other ways it is applied. This chapter provides a comparative look as to how threat modeling, in its original form, has been applied in hostile environments that encompass both physical and logical attacks, most notably in tactical military operations. Though looking at threat modeling in a context outside of application security may seem irrelevant, it is important to understand a historical use. Threat modeling's past uses are not only useful to learn and remember, but also provide an appreciation as to how strategic analysis becomes a fundamental part of the process.

Topicality of Military Threat Modeling

By understanding the historical usage of threat modeling, security professionals at large can evolve a mindset built around strategy rather than segregated and disorganized knee-jerk responses. Thus far, the outcomes of reactive methods have fallen short of adequately addressing a growing number of threats to application environments worldwide. The gap between the complexities of attack patterns and advancements in countermeasures continues to widen. Lending from military origins, threat modeling develops the discipline behind threat analysis. For decades, the US military has leveraged threat modeling to obtain improved insight as to how an enemy could adversely affect US interests or military forces. This analysis encompasses the examination of an enemy's motives, capabilities, and likely attack scenarios as part of an overall objective of defending against as many viable attack scenarios as possible. Similarly, application threat modeling extends the capabilities and resources of security professionals. Lending from this process, professionals can dissect and understand attacks, correlating them across multiple application vulnerabilities. Security professionals who learn from the military's application of

threat modeling will be able to introduce innovation where it has been significantly lacking – intelligence correlation. Specifically related to the ability to correlate exploits and vulnerabilities and ultimately map these factors to possible misuse cases prove to be a key value-add to threat modeling.

Profiting from Threat Modeling in War

In Sun Tzu's quotation, the phrase "*profitable way of carrying it on*" noticeably stands out. While profit is not usually associated with war, here it refers to the gain or reward received from understanding the *evils* of war. The gains are the avoided risks that could have introduced mission critical impact levels. In essence, most military strategists adhere to the philosophy of profiting from the realities of war via improved preparedness. A military's application of threat modeling is able to provide this capability in part through the use of threat modeling techniques. Threat modeling allows the *evils of war* to be better recognized using thought-out simulations. Although not all possible scenarios can be considered and modeled, the military seeks to play out the most probable attack scenarios. Ultimately, threat modeling is not able to eliminate the possibility of attack, but instead increases the state of readiness for which a military unit can effectively respond to a threat.

Threat Modeling @ DoD

Several divisions within the US Department of Defense have effectively applied threat modeling techniques to identify war's collateral risks such as casualties, illnesses, and adverse economic and environmental effects. The US Army and NASA have used Ballistic Missile Threat Modeling for more than 50 years. By applying intelligence gathered from foreign missile systems, the United States fortified their overall missile defense system. Over the years, the DoD used threat modeling to build a stronger missile defense program by identifying threats (with underlying attacks) that were able to permeate US defenses. Deriving impact levels and correlating them back to the threat model quantified the level of risk associated with branches of the attack tree models. Impact levels are critical and complicated pieces of information that require thorough understanding to effectively apply the appropriate level of countermeasure to the identified threat. Overcompensating controls can deplete resources in other areas where threats are potentially more probable and damaging. As a result, reliable threat models of foreign missile systems are periodically studied to determine likely threat scenarios in an ever-evolving global arms race.

Ballistic Missile Parallel

Similar to application threat modeling, ballistic threat modeling revolves around the necessity for good intelligence. In broad terms, intelligence refers to pieces of information that can be used to reveal strategy, strengths, and weaknesses of a force's military capabilities and assets. Within the framework of various application threat models, intelligence takes the form of a vast knowledge of attack patterns (via

a growing and up-to-date attack library) as well as access to a well-managed and continuously updated vulnerability database. Information surrounding application vulnerabilities and attack patterns provide two key areas of intelligence for building a strong application threat model. Each varying source of intelligence is correlated to other sources by using a tree model where a root threat is supported by multiple branches of attacks and corresponding, perceived vulnerabilities that facilitate the introduction for an attack. A threat may encompass various branches of attacks (as part of a studied attack tree), each with a vulnerability for which the attack's probability of success is elevated. It is evident, therefore, that an extensive supply of intelligence (understood attack patterns and vulnerabilities) needs to be present to provide for realistic threat simulations. Limited insight into existing or evolving attack patterns or, conversely, the understanding of vulnerabilities in infrastructure, can greatly diminish the worthiness of a threat model. Related to the military example of ballistic threat analysis, the military has sought the assistance of internal and external experts to best understand both current and projected missile threats. Intelligence is key. The establishment of and interaction with intelligence communities greatly assists in itemizing what existing and future missile threats are likely. In turn, missile defense teams leverage the gathered intelligence to refine their internal missile defense capabilities. Comparatively speaking, these efforts are synonymous to the attack/exploit research in today's application security. Acquired intelligence is correlated to one or many vulnerabilities or defects by software systems that could be labeled as targets.

A Continuous Process

The military applies threat modeling as an ongoing process aimed at assessing both internal capabilities and external threats. Continuous evaluation has many advantages over one-time or interval evaluations: namely, it allows for more accurate data via increased frequency in which data is obtained, reviewed, and reported. The unique characteristic of the military's threat modeling process is that data research, review, and reporting are incorporated into many job duties, particularly in defense areas where threats are more probable. Nearly all personnel are required to report threat data, regardless of job function. For example, status reports are deliverables within the US Army that reveal the condition of a designated group, combat unit, or military installation. These efforts take place daily and provide up-to-date synopsis of capability. These reports (or assessments) provide a current status on physical and/or logical infrastructures capabilities, integral to offensive or defensive strategies. This aspect of integration is quite interesting when correlated with existing security efforts at most global organizations. The majority of companies have opted for a different approach by filtering out information security procedures from daily business processes and assigning accountability to segregated security groups. As a result, security groups are predestined to assume an adversarial role when interfacing with business groups. Security professionals are faced with numerous Chinese walls from business and technology groups who serve as the audience to their assessment efforts, thereby limiting the critical first research step to initiating an effective assessment. The schism between managing and evaluating capability inhibits the overall ability to effectively develop a continuous process for accurate assessments.

Looking Within

Internal assessments within military operations take many forms. Readiness reports, for example, reveal in-house technical and physical abilities for offensive, defensive, and/or supportive efforts. The Army leverages readiness reports to measure the capabilities of its troops and to provide flexibility for those who access this data. Military personnel at multiple levels use an infrastructure to input their respective readiness reports, reveal changes in capability, or report problems. Multiple layers of military personnel can review the information gathering using various computer-based systems that centralize threat intelligence. Moreover, the continuous assessment process cultivates strong countermeasures against security breaches, constantly evaluating data on internal capabilities. US Army officials use any readiness gaps found for clear direction on what countermeasures are needed to address adverse changes or declines in readiness levels.

As previously mentioned, the US Army has taken the time and investment to develop an internal system that manages data associated with internal assessment efforts. The Defense Readiness Reporting System (DRRS) catalogs personnel, logistics, and equipment readiness from a centralized location. The DRRS (along with other systems) gives the US Army close to real-time assessment capabilities, maintaining various reports that are frequently updated with new information. This information repository assists in addressing changes in process and/or resources that may adversely affect defensive and offensive tactics. Overall, these ongoing internal reviews of resource and/or process level changes will undoubtedly reduce the viability of possible threat scenarios against the US Army and its military installations.

Private or publicly owned companies would do well to imitate a similar process for which continued assessments reveal up-to-date platform, control, and process changes. Organizations where software development is central to client-facing product or services would benefit most from a program that periodically makes gap analyses of ongoing technical and security assessments. Such a program would expose process or technical deficiencies more quickly, hastening the rate at which countermeasures are applied to discovered vulnerabilities. Devoid of such a program, the status quo manner of conducting assessments on infrequent timetables will needlessly elongate the remediation time on existing vulnerabilities. Queued vulnerabilities, compounded by potential internal threats, may produce highly viable threat scenarios if outside interest groups can be certain that vulnerable targets are not scheduled for remediation within their attack time frame.

Thus far, we have addressed the introspective look within an organization and seen how the military assesses their resources and capabilities to provide a readiness measurement. An inward regard of capabilities at most organizations (albeit outside the context of threat modeling) may encompass points related to awareness programs, governance, and audit programs. Internal compliance to one or more baselines is already common practice. The frequency that such assessments are made, however, is not to the level necessary for a solid foundation of up-to-date information sources. Next, we will look at how the military looks outward to its adversaries to understand

their capabilities, vulnerabilities, and potential interests – all key variables within the context of threat modeling.

Art of Espionage

Surveying internal readiness is parallel to the necessity of gathering information about an enemy's intent and capabilities. Reconnaissance exercises within the military follow several degrees of complexity and sensitivity to time, risk, and available resources, among other factors. Threat models must account for various critical factors such as an enemy's attack motive, capabilities, vulnerabilities or flaws, and amount of information. The complexity of threat modeling lies in expedient analysis and process development. In ballistic threat modeling, for example, the process must allow intelligence gathering to feed missile defense designers in a sufficient time frame so they can defend against future threat scenarios. A race condition emerges between two intervals. One-time interval relates to when information from reconnaissance efforts is evaluated and used to guide designs efforts in a missile defense system. The second interval is the time associated with a rapidly maturing threat scenario, accompanied by underlying attack sequences. Adding to the complexity, sometimes reconnaissance efforts do not yield credible information. Misinformation can derail a threat model. Following an incorrect set of attack scenarios also misleads defense efforts from designing an effective countermeasure. While the stakes are not as high as those in ballistic threat modeling, the ability to obtain highly reliable, recent data will better equip threat models to convey probable threats and impacts with greater accuracy. In turn, the ensuing security requirements serve as guidance for the development of countermeasures that reduce risk scenarios revealed by the threat model.

Reconnaissance is multifaceted. Espionage requires covert operations behind opposing lines, often requiring the ability to perpetrate enemy actors or personnel. Finding good, reliable information often takes extreme conditions and efforts. Within the military, reconnaissance carries its share of risk: jeopardizing mission objectives, involved resources, and even compromising sensitive information. In application threat modeling, reliable information is also vital, although the risks are much less extensive. External information sources may include application/platform vulnerabilities, as well as a thorough attack library containing current and past exploits that could be used in the form of an attack. An attack library would encompass the exploit or series of exploits that are necessary for the attack to be successful. These information sources drive the robust application threat model, similar to how missile defense designers rely on good intelligence for developing a successful ballistic threat model. Both models depict realistic threat scenarios that a defense system should be prepared to defend. The effort becomes even more daunting for missile defense designers who base much of their design efforts on a baseline threat models that have been affected by intelligence reports. Obtaining good information is easier said than done when fueling application threat modeling efforts and similar to ballistic threat models, are highly dependent on solid information. Similar to the problems that missile defense designers face in adjusting missile defense programs

to an evolving threat model, software architects and developers will also have to consider flexibility in their products so they can respond to changing threat scenarios presented via application threat modeling. This makes threat modeling a "living" or ever-changing process that requires updating. An already constructed threat model is rigid in form but assumes greater flexibility by the inputs it receives in terms of threat intelligence. Ultimately, countermeasures designed to incorporate a "living" threat model will have to either evolve in capabilities or give way to newly developed countermeasures that extend beyond a countermeasures current state of defense measures.

Designing Countermeasures

Beyond good intelligence, ballistic threat models have employed good design. According to the Aerospace Corporation, some of the best threat models developed combine both good intelligence of foreign ballistic systems and *superior* knowledge of defense designs. Designing effective countermeasures in software applications is one of the key differentiators of application threat modeling over other traditional security efforts (which may only address a portion of the overall threat and associated risk). Designing good countermeasures in ballistic defense systems involves not only addressing perceived threats via good information and attack assumptions, but also foreseeing how the same threat may evolve or assume a different form. At times, attack patterns may revert to historical, classic attacks that are perceived to be ineffective. This perception provides a false sense of security and a way for attackers to revert to more classic attack patterns. In 2007, a decade old boot-sector virus, named Stoned.Angelina, infected many Vista machines being sold at retail stores. The machines were equipped with A/V solutions; however, the signature sets that were loaded onto the machines did not include defense for the classic virus because it was not perceived to be a threat. This simple example demonstrates that countermeasure design must be (1) ongoing, (2) based upon both historical and new data, and (3) flexible to encompass changes in design. The same type of flexibility is required by defense system designers who must understand what factors periodically change relative to the original threat. To ensure a good defense system, designers must address static and dynamic criteria of the threat that are likely to change (behavior of missile, projectile path, etc.) and those that are not (i.e. – size of missile). Similarly, in application threat modeling, there are threat elements that are more consistent in nature as well as those that are more variable. Application threat modeling users will have to diligently ensure that changes in a threat model, previously used to create adequate application-level countermeasures, are regularly updated so both the model and the countermeasures used are commensurate to the threat.

SUMMARY

Unfortunately, the threat modeling within the Software Development Life Cycle (SDLC) has not reached a maturity level comparable to that of the military. However,

agencies within the US Department of Defense have had a lot more time to refine their process and have a few more resources at their disposal. This vast difference in maturity levels of applying threat modeling across two distinct environments allows software development teams and security professionals to leverage the many lessons learned by the military and see how their procedures for intelligence gathering, threat assumptions, and design can be achieved as part of an integrated process. The chances that a banking institution, utility company, or even software provider will incur the costs of managing one too many standalone processes that support threat modeling efforts is far fetched; however, the roles and responsibilities that each subprocess follows may be easily executed by members of existing resources. This will indeed be yet another difficult, process-related challenge that companies will have to face when adopting threat modeling as part of their strategic security initiatives.

Perhaps the most difficult challenge for today's security groups is the need to change the status quo perception that security equates to compliance. This viewpoint quickly negates more strategic approaches for application security. It particularly undermines threat modeling as a possible enabler to a strategic security assurance program. Ironically, a type of mutiny takes place within organizations as security professionals attempt to convince information owners that achieving compliance is not the same as achieving security. Within the military, conflicting or competing objectives would never provide meaningful threat modeling results if the process was challenged or stunted from fulfilling its full potential for analyzing threats.

RATIONALE AND EVOLUTION OF SECURITY ANALYSIS

> "*Other than a nuclear device or some other type of destructive weapon, the threat to our infrastructure, the threat to our intelligence, the threat to our computer network is the most critical threat we face …*"
>
> *FBI Director, 2009*

Cyber warfare … zero-day … botnets. These terms depict the insurmountable challenges facing information security professionals today. The FBI's quotation reveals a growing rationale for bolstering technology, innovation, and collaboration in the area of information security. This quotation for many should simply be a trite expression of the obvious – a dire need to secure information borders within the public and private sectors. The intent behind this chapter is not to overplay the same incentives, business cases, or moral justifications behind information security efforts. This chapter will bolster the rationale of evolving application security to a new paradigm that extends beyond the mentality of equating security to compliance, rather than be content with entrusting the reigns of security to the latest prominent security vendor, regardless of magic quadrant ratings. The intent is not to minimize these efforts, but to learn from them – building upon their use to a new echelon of applying strategic thought to information security.

Although freethinking groups exist across various security disciplines throughout the world, this sort of progressive thinking erodes within the walls of many companies where more practical, stale security philosophies are driven by the concept of best practices. There is nothing best about "best practices." Such catch phrases have misled organizations into a false sense of security by encouraging them to only strive for a basic maturity level of security controls and processes. There is no question behind the intent of *best security practices*, as well as the many frameworks, policies, standards that are omnipresent within our industry. The responsibility truly lies with executive leadership and the follow-through that needs to take place beyond a primer application of best security practices. The shortcomings in adopting new forms of security strategy may be attributed to the perception of additional cost factors in technology, resources, or services. Most security leaders, perhaps due to higher level influences, are reluctant to break a good thing. The colloquialism "*if it ain't broke, why fix it*" is pervasive across security management, especially when having to justify new budget numbers. As senior executives continue to only live in the now, their adversaries are quickly looking ahead at the future of their attacks. Given all of the aforementioned information, the rationale for introducing threat modeling is to evolve security processes to a higher level of strategy, efficiency, and foresight, as well as being conducive to improved fiscal responsibility. Could application threat modeling point to a new utopia between security and business enterprise? Not exactly, but it is definitely a good start.

Environmental Threat Factors

Both opportunities and motives are key elements of threats and attack plans. Both are affected by environmental factors within the global ecosystem of politics, business news, and events. The opportunities for exploitation and/or well-defined attack motives can be greatly influenced by these environmental conditions and ultimately alter the following characteristics of an attack:

1. Intensity of a planned attack.
2. Sophistication of an attack.
3. Probability for successful exploit.
4. Ability to distort/eliminate forensic evidence.

In this section, we will examine motives and opportunities in relation to environmental factors. By understanding their roles in originating threats and attack plans, we can apply a stronger preventive and strategic program via threat modeling.

Product of the Environment

Even before these attack motifs become produced, environmental factors provide the trace of accelerant to ignite motives into fully operational attacks. The term *environment* is not to be confused by the application domain or application environment, which is limited by the functions of its authorized and unauthorized user base.

Instead, the term *environment* describes the social, political, economic, belief-based, and/or financial factors that serve as key drivers upon which software adversaries act. Revenge, spite, corporate espionage, and fraud are motives fueled by environmental conditions such as war, layoffs, recessions, financial distress, social injustices, and much more. This is just a short list of environmental examples for which hypothetical attack plans can evolve into mature attack plans. Coupled with opportunity, a motivated attack becomes even more precarious as environmental factors increase the probability of an attack. Environmental factors tremendously facilitate attack windows of opportunity similar to how they inspire attacks. Events in the social, political, environmental, or economic climate can provide a ripe occasion for conducting attacks. Changes in the environment oftentimes reduce barriers or obstacles that naturally or artificially exist to mitigate threat scenarios. The following diagram provides a cause and effect flow of events stemming from environmental factors and resulting in attack patterns.

Figure 1.1 provides a visual representation of how environmental factors serve as additional intelligence when identifying probable attack scenarios during application threat modeling. The environmental condition of an economic recession creates multiple motives for attacking a financial application, per se, where the attackers may fulfill their multiple objectives. In this minor example, these objectives reflect a growing need for either financial self-preservation or gaining auxiliary income to offset financial shortfalls. Each of these motives becomes associated with possible attack scenarios against an application environment, along with the targeted asset(s).

Threat models in application security traditionally address threats and underlying attack patterns, along with their intended targets (as well as other variables that will be covered extensively throughout this book). None of these other variables within the threat model preface the phase in which threat assessments and attack analysis

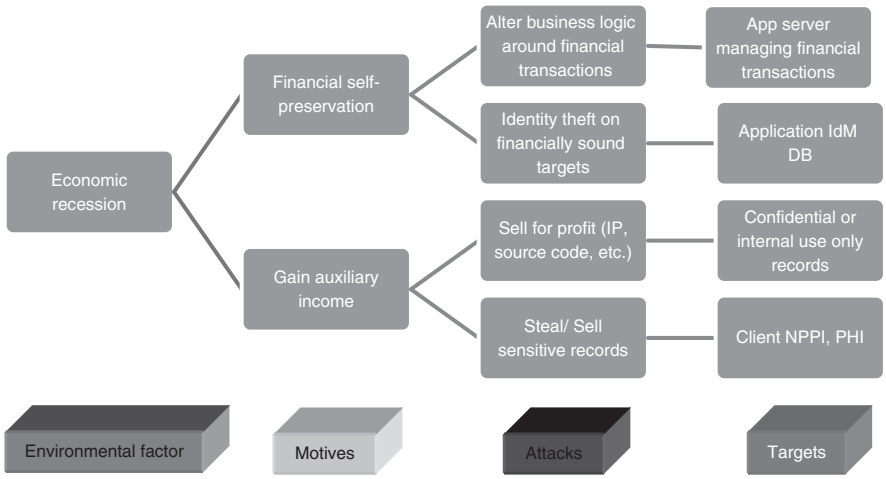

Figure 1.1 Relating Environmental Factors to Attacks

TABLE 1.1 Correlating Environmental Factors to Attack Motives – SAMPLE

Industry	Environment Factor	Possible Motive
Government	Increase antigovernment chatter	Upholding political or personal beliefs
Utility (nuclear)	War	Retaliation
Financial	Downtime economy	Financial gain
Software company	Increased turnover	Revenge, spite

occurs. Threat modeling exercises should include environmental factors as variables. Incorporating these factors may be simply anecdotal to any given threat model or may serve as key evidence in substantiating threat claims. In either case, environmental factors, motives, and opportunities are elements that undoubtedly affect threat characteristics and greatly influence the ability to better forecast the timing and probability of attack scenarios.

Forecasting attack scenarios is accomplished by first having a thorough understanding of environmental factors that may encourage certain types of motives. Table 1.1 lists examples across multiple industry segments and relates them to possible attack motives.

Qualifying environmental factors, as a precursor to threat modeling exercises, bolsters the strategic forethought associated with threat assessment efforts on application-based attacks. In the following section, we examine how motives, combined with ripe environmental factors, can compound attack probability levels and even exacerbate the sophistication level of an attack against an application.

Judging by Motives

Behind every threat is a motive, even if the motive is simple curiosity. Application-based attacks differ no less. Before a scan is run, payload is altered, or business logic is abused, the attack design must have an objective. Even seemingly benign attack probes or reconnaissance efforts against an application environment carry their own set of motives, quite possibly ulterior motives. From random injection attacks fueled by curiosity and bragging rights to elaborate plans to circumvent layers of security protocols, motives propel threat scenarios to attack plans. Most importantly, they begin by serving as an initial probe against any defense mechanisms that can foil attack plans or complicate goals for repudiation. Motives should be analyzed within the application threat model because they had better identify probable attack scenarios plotted against an organization's application environments. Additionally, identified motives can assist in forecasting the attack's sophistication level.

Assuming that all attack scenarios are driven by financial gains is flawed. Although these gains do represent the primary motives behind most attacks, universally presuming all attacks are financially motivated could mislead those responsible for defending

against them. Understanding attack motives provides clarity to possible targets, attack vectors, and, consequently, related countermeasures to defend against attacks. For example, a politically charged attack against a government site could involve attacks related to site defacement and Denial of Service (DoS) instead of those attempting to compromise data sources. Another example is that of a disgruntled employee at a financial firm who, given the right opportunity, may focus on high-impact business applications. In instances where motives are driven by revenge or spite from a company employee or former employee, high-impact targets are susceptible to attacks that affect data integrity, business continuity, and confidentiality.

An added layer of complexity to understanding motives is attacks solely devised to distract or deceive by simply serving as a diversion tactic. These types of attacks are exceptionally difficult to decipher since they may be launched from disparate networks, making event correlation difficult to accomplish. In these cases, the intent is for one or more attacks to lure resources and attention away from intended targets. These types of attacks are generally highly motivated and prefaced with a significant amount of planning. For this level of sophistication, counterintelligence efforts are invaluable in order to isolate possible sources. Counter-intelligence provides a preventive approach to understanding the greatest threat to an application environment. Tactics such as threat profiling are used to profile attack sources, entities involved, motives, capabilities, and access to resources.

Unfortunately, most organizations do not have their own counterintelligence groups to uniquely identify and qualify threat agents, particularly in the area of application security. Such an effort would require an enormous amount of time, effort, and money – all of which most organizations have sparingly. Some companies may obtain such intelligence via threat monitoring service providers, who aggregate growing lists of threats and attack exploits and deliver them via data feeds, or threat feeds. Threat feeds help build robust and up-to-date attack libraries that can be leveraged during threat modeling. However, companies employing threat feeds should be wary of overly depending on such feeds as the sole pieces of information for determining probable threat scenarios. Threats observed over public networks, honey pot farms, or in-the-cloud service providers only reveal a breakdown of threats to public infrastructures and do not precisely assess what may affect a specific organization. Although some threat feeds reflect data obtained from deployed network or host-based sensors across relevant industries, such data should not be taken as gospel for threat analysis. There may be other motives for unique and targeted attacks. As a general rule of thumb, a single source of information should not drive preventive application security measures, but simply serve as an added form of intelligence in building improved application countermeasures.

In an industry driven by benchmarks and outside influences, a balance must exist within security groups to leverage external research data and internal self-assessment exercises. Skimming the top attack scenarios from a threat feed and adopting it as the main source of information from which to build countermeasures follows a misguided mindset: whatever is good enough for the security masses is good enough for my security strategy. Such a myopic form of threat assessment places a greater emphasis on external sources for threat intelligence (as a basis for forecasting threat scenarios)

over a company's own ability to assess analyzed threat scenarios, including unique characteristics of a company's physical and logical infrastructure. Adhering to the "Top 10" approach can easily give an organization a false sense of security based on the belief that relevant threat scenarios have been adequately addressed. An organization might learn a harsh lesson if lower ranked threat scenarios, not detailed within a received threat information feed, proved to be the most likely threat scenario to them. In light of the fact that all company resources and efforts may have been placed on top-level threat scenarios, countermeasures in other areas of the physical, or network infrastructure where the likely attack took place may have been overlooked.

The point to be made is *not* that these threat intelligence subscriptions are ineffective – on the contrary, they are extremely capable of identifying prevalent threats that have been observed and reported by a multitude of sources. This form of intelligence is highly useful when applied in the uniqueness and context of an organization's application environment. They are also precious resources in curtailing the time and effort to prepare for blanketed attack infrastructures, namely botnets, which may exhibit an array of threats identified by a threat aggregation service. However, beyond using such threat feeds, which may only encompass high-level or "Top 25" threats (depending on subscription), companies must consider other threat agents that make up their respective threat landscape. This may very well be some threats not provided by the threat aggregator service provider. In general, security information sources and tools should always be used after having established a strong understanding of the application environment and most importantly, the data with which it interfaces. Application walk-throughs, along with data flow diagramming, greatly develops this level of understanding for the threat modeler. These exercises can also attract a broader audience, fostering collaboration among developers, architects, business analysts, system administrators, security analysts, and QA team members. As each team becomes better acquainted with an application environment in review, vulnerable points can be collaboratively identified. Since the SDLC process should already encompass these individuals, all having varying insight into an evaluated application environment, application walk-throughs and data flow diagramming can quickly achieve the following objectives:

- Improve understanding of an application across multiple levels
  - Platform Level
    - Interrelated software dependencies
    - Local/Domain level privileges
    - Required ports and services
    - Hardening system requirements
  - Application Level
    - Use cases
    - Business Logic
    - Application privileges
  - Network Level

    - Network-based security (ACLs, network device security policies)
    - Scalability and bandwidth concerns
    - Redundancy
    - WAN/LAN based data requests
    - Use of PKI
    - Whitelisting/Blacklisting requirements
- Identify misuse cases that exploit poor business logic or code in software application
    - Login process
    - Registration process
    - Data requests (example: reports)
    - Application alerts (e-mail, SMS, etc.)
- Identify possible attack motives to data application data sources
    - ID Theft
    - Revenge
    - Financial motivation
    - Intellectual Property Theft
- Correlate attack motives to possible threat vectors in order to depict an initial threat landscape
    - Table 1.2 reveals how threat scenarios line up with motives and a subset of attack vectors.

Table 1.2 is an example of a well-designed matrix of how possible motives against an organization can be uniquely defined through simple assessment efforts. These assessment efforts may already exist either internally or through an external group. Surveying a diverse pool of technology and business users will help determine what potential threats are perceived by the organization's members and help identify those unique, targeted attacks. Table 1.2 reveals how threats encompass motives, target assets, and possible attack vectors to be used against an application environment. The table is meant to serve as a template for future use by threat modelers and risk analysts in beginning to correlate environmental factors to the vectors in which an attack would ultimately be introduced.

Attack environments will undoubtedly vary among one another and will be driven largely by unique factors, related socioeconomic conditions, and other personal ideals. Combined with strong motives, the makings of targeted attack plans began to unfold in the minds of the attackers. Many argue that the focal point should simply be the actual application attacks that target an application, such as session hijacking or elevation of privilege type exploits. The problem with this approach is that two essential questions remain unanswered: (1) who are they? and (2) what do they want? Understanding environmental factors and motives fueling attacks allows security groups to create multiple attack profiles. This leads to answers on the most likely profile types to conduct an attack with the greatest impact. Such a profile will also

TABLE 1.2 Correlating Motives to Application Threat Vectors

Threat	Target	Motives	Attack Vector																		
The following table represents simply a subset of possible threats and relevant targets, driven by motives over well-defined and understood attack vectors.			Web Server(s)	Mail Server(s)	Intranet Sys	Authentication Trust Model	App Server	Portable Media	Wireless APs	External DNS	Network dev./app	Offshore Dev.	Social Eng.	Call Center (domestic)	Phy. Intrusion	Emp. Insider	LAN Based	IVR/VRU	API's	VPN
Corporate espionage	Source code	Imitate a competitive application																		
		Inject objectionable code																		
	Business plans	Gain insight into service roadmap																		
	Emerging technologies	Replicate patented solutions																		
	Business processes	Business disruption																		
		Product/service imitation																		
	Confidential business records	Identify weak points																		
		Establish leverage																		
		Create competitive adv.																		
	Network schematics	Understand weaknesses in application architecture																		
Identity theft	Personal identifiable information	Facilitate repudiation																		
	Financial records	Financial leverage,																		
DoS, DDoS	High-impact/visibility business systems	Competitive reasons																		
		Politically charged																		
	WAN/LAN	Fueled by beliefs																		
	DNS ext./int.	Retaliation																		
Elevation of privileges	Access control mechanisms	Data access																		

reveal attackers' interests in specific application environments. Validated by log/incident data analysis over a sustained period of time, companies could prepare and refine threat profiles as a mitigating step against targeted attacks. This approach largely benefits the targeted attacks, which are often the most damaging and costly.

A final key difference between analyzing environmental factors and simply responding to top application-based threats is a stronger understanding of intent obtained via the former. Understanding the basis of an attack allows an application threat modeler to emulate the mind of the attacker.

Practical Application

Environmental factors provide improved calculations on attack probabilities as well as the prognosis on severity levels of observed attacks, either before, during, or after they have taken place. Admittedly, these efforts do require a significant amount of time. Across most industries, time and resources continue to deprive security groups from adopting techniques to assess environmental factors. As a realistic approach to executing these recommendations, organizations can adopt one of the following frequencies for analyzing environmental and motivational factors.

Table 1.3 aims to provide some degree of regularity in reviewing new and evolving environmental conditions. Constantly changing conditions heightens the probability for attack scenarios and their associated impact levels. The frequencies and scopes are driven largely by the overall historical and future sense of cyber-attacks against an organization's many application environments. The data obtained from each review should only be valid for a maximum of one year given the onset of new and developing environmental factors, both internal and external, that may trigger or accelerate threat scenarios against application systems and related data environments.

TABLE 1.3 Recommended Frequency for Environmental Threat Factor Analysis

Frequency	Scope	Details
Yearly	All business units	A comprehensive assessment determines the unique environmental factors and motives that adversely affect all business units.
Bi-annually	Alternating top 7 business units	If number of business units is less than 7, then a repetitive cycle of existing business units can be performed.
Quarterly	Alternating top 3 high-impact business units	Review top 3 high-impact lines of business, followed by a new 3 lines of business for each sequential quarter.
Monthly	Single high-impact business unit	Alternate across the organization, addressing one high business impact unit per month. If less than 12, repeat with the highest impact business units beginning the new cycle.

Sources of information during these periodic assessments will also vary greatly and be spurred by available resources. Internal personnel will ultimately be required to conduct analysis of internal factors to the organization. These may include the following types of evaluations:

- *HR Meetings:* Interviews with HR to identify cases where previously reported personnel cases provide a level of indication that certain employees may wish to act against the organization or its objectives. Obtaining information on disgruntled employees, for example, may provide early threat detection capabilities for certain types of information and operational threats.
- *Personnel Surveys:* Any HR surveys that seek to identify personnel viewpoints on the organization. This will help define enterprise- or department-level issues that could become organizational and/or environmental factors to consider for insider-based attacks.
- *Threat Feeds:* Threat feeds from external sources in which data reflects recent and aggregated attacks against similar companies, companies within the same industry sector, and companies of similar cultural and organizational makeup.
- *Third-Party Assessments:* External assessments performed by third parties help identify environmental factors and possible insider attack motives that would not have been discovered via internal assessments or employee surveys. Existing service providers may provide such assessments if their core competency includes those services.
- *Ingress Traffic Analysis:* Comprehensive review of ingress traffic across multiple entry points and correlated by geographic source, date/time, protocol, and separated by authorized IP sources (authorized third-party vendors) and unknown/unrecognized sources. Existing software may already detect and log network anomalies.
- *Access Audits:* Sensitive applications with logs set to record successful and failed logins. Correlate successful logins to time of day and frequency and perform the same correlation for failed events. Unify both data sets to cross-check for failed and successful logins on certain days/times. Anomaly detection would also be useful if it is inherent to the application or to a security product that interfaces with authentication application events.
  - *User Entitlement Reviews:* A subset to this is to review the current entitlements of users on a periodic basis. This may take place monthly, quarterly, or yearly. This exercise alone does not point to environmental or motivational factors; however, they may provide clues to unauthorized operations in user provisioning, which may point to future security circumvention.
- *Socioeconomic Analysis:* A review of external environment factors outside of the organization will ultimately affect employees in order that they behave either more or less rational with respect to their job functions and the due care that they would need to have with application environments used within their job functions. Economic distress, fears for personal security, personal beliefs fueled

by current events may all play a part in triggering some degree of action that can adversely affect an application environment.

All of these practical exercises are simply a subset of what can and should be analyzed as a part of a periodic assessment. Studying such environmental factors that may ignite attacks against application environments, either internal to the organization or against other application environments foreign to the company, is a vital part of any assessment. Unique factors to each organization will ultimately help create a customized assessment plan for ongoing evaluations.

SUMMARY

Attacks against applications are influenced by environmental factors and driven by motives. Socioeconomic conditions may provide a ripe time for attacks against application environments to yield either greater results or improved probabilities for success. Assessing these factors in conjunction with technical threat analysis within any given threat model provides greater readiness levels on behalf of the defending application owners.

BUILDING A BETTER RISK MODEL

"*More people are killed every year by pigs than by sharks, which shows you how good we are at evaluating risk.*"

Bruce Schneier

Identifying risk should always be the key objective to application threat modeling. Threat identification and attack mitigation via countermeasures are important, but of greater importance is the ability to identify and mitigate business risk stemming from threats to application environments. Despite the many advances in security technology, understanding how existing and emerging security controls mitigates true risk is elusive.

The Inherent Problem

The problem with measuring risk today is that it is clouded by fear. Perception and subsequent reaction to perceived threats draws misguided conclusions for many attempting to mitigate risk. Information drives perception, and in application security, the manner in which information is handled determines whether or not appropriate risk mitigation efforts are properly executed. Fear has crippled many organizations into becoming less effective in dealing with application security. Organizations are paralyzed in HIGH-risk remediation queues and compliance gaps. Instead of adopting a strategic approach for application risk mitigation, reactive

TABLE 1.4 Key Reasons App_Sec Fails Today

1. Discrepancy between perceived and actual threats
2. Gap between current threats and existing preventive measures
3. Misconception of attack exploits against software
4. Greater use of detective/reactive app_sec controls versus preventive security controls
5. Inability to apply security controls as designed and/or as intended
6. Nonsecurity professionals involved in software development efforts or other key areas have limited security knowledge
7. One-dimensional approach to app_sec
8. Inability to factor in insider-based attacks
9. Misguidance attributed to FUD factors
10. Applying general security principles to a specific app_sec problem

responses drive one-dimensional security plans centered on those same HIGH-risk areas or compliance shortcomings. At the helm of this misguided approach is senior management, continuously seeking the best silver bullet at the lowest price. Over the years, fear has obscured reason and ingenuity within the realm of application security. Although great strides have been made in app_sec[1] related tools and technologies, their introduction and use within the context of reducing application risk has been nominal. The key reasons are mentioned in Table 1.4.

Table 1.4 is not meant to be a comprehensive listing of variables that lead to certain failures in application security, but instead a synopsis of commonly observed factors that inhibit a more strategic approach to app_sec. All of these factors, in varying degrees, affect the accurate depiction of application risk, which in turn limits that ability to derive any degree of business impact from identified application risk factors.

The rationale for threat modeling is to achieve a level of risk mitigation via a preventive, strategic approach to app_sec. This is a clear breakaway from the status quo mentality, where ingenuity is replaced with popular security trends, regardless of the unique nature of their industry, business, observed attack patterns, and environmental factors. In the next couple of chapters, we will take a look at how security strategy can evolve beyond a *keeping up with the Joneses* mentality. In the upcoming sections, we will explore the rationale or business case for threat modeling, which primarily revolves around the following paybacks:

- Better Form of Preventive Control
- Improved Application Design
- Effective Remediation Management

Business Case for Threat Modeling

Among all the rhetoric surrounding maturity modeling, six sigma–inspired projects, and ISO-driven benchmarks, one would think that implementing a framework for

[1] Application security.

improved application design and reduction in application risks and remediation time frames would be quickly adopted by members of the business community. However, the reality is that the majority of business groups are more concerned with the implementation of functional requirements versus security requirements. Traditionally, security requirements featured in the country that originated and mastered the fast-food concept, rate of service is paramount, especially for software development. The race to market with new products and features is always a high business priority. One of the numerous risks of such a speedy application development tempo is introducing multiple viable application exploits, which can jeopardize customer or business information. Unfortunately, these risks are generally lumped into an acceptable risk category. For many businesses, mitigating legal, financial, and/or regulatory risks take place via alternative countermeasures, such as improved contractual language or insurance policies that protect against financial fraud, loss of intellectual property, or unauthorized data disclosures.

Recognizing that process efficiency does not sell like cost savings, and realizing that compliance FUD has lost its luster in validating security investment, particularly when trying to lobby for executive sponsorship, it is important to highlight the various cost saving opportunities that can realistically be achieved through the adherence of a threat modeling program.

The following is a list of key benefits in developing and sustaining threat modeling efforts within an enterprise.

1. *Business Applications as Attack Vectors:* Software applications are low-hanging fruit for cybercriminals since software vendors do not have the same level of maturity in testing and patching as platform vendors for various operating systems. The differences in disclosed application vulnerabilities versus those at the application level are worlds apart. Microsoft's MSDN site has an excellent blog revealing numbers from their annual Security Intelligence Report (December 2012), showing that only 12.8% of vulnerability targets disclosed were operating systems. For this reason, businesses need to focus more on addressing threats to business applications, particularly early on in the SDLC. Today's reactive efforts to thwart security-based attacks equates to a rowboat trying to catch up with a motor boat. Strategic forethought has been needed for a number of years now; however, the paradigm for application security has focused on processes and controls on the heels of discovering that a major breach has occurred, or that a critical vulnerability requires immediate remediation. As sophisticated malware artists exploit the power of this knee-jerk reaction, more advanced attacks can encompass diversion tactics in order to spread out the presence and effective use of any mitigating processes and controls. Application threat modeling introduces strategic forward thinking for probable attack patterns and vectors for a given enterprise, allowing organizations to mitigate possible threat scenarios based on current and good threat intelligence – another key component to the overall threat modeling process.
2. *Reduced Remediation Time and Efforts:* Anything that equates to more time in business also equates to additional cost. Remediation, traditionally taking place

in a postimplementation sense, has resulted in a workflow that, for most organizations, is truly insurmountable and costly. Since the threat modeling process addresses the most probable attacks and vulnerabilities that affect an application, remediation of weak or missing software countermeasures is addressed early in the development process. Most organizations have reached a level of remediation backlog almost matched by the number of security exceptions filed by business unit managers who oppose remediation efforts on their own information assets. Adding more chaos to this broken process are the current methods for tracking and managing remediation tasks, which continue to operate without any major changes to a highly ineffective and inefficient process. The amount of time and money consumed supporting a process that yields little to no risk reduction is immeasurable. Remediation and exception management – two out of control GRC efforts today – are both costly and ineffective in their production of security controls and risk reducing efforts. Nearly all information security and enterprise risk managers can truly identify with this problem today and welcome a new era of greater risk management efficiency. Application threat modeling lends to an improved risk model by injecting itself into a process that prefaces actual development efforts, thereby addressing security concerns up front in the SDLC. When platform vulnerabilities and software/service components are built and hardened to the specifications of the supported business application, remediation tasks are greatly curtailed and risk levels are reduced. In either case, time and money are saved through the proper use and application of a threat model to identify attacks, vulnerabilities, and key information assets of the greatest business impact. As with any security control or process, nothing completely eradicates risk. However, much of what is mitigated up front via application threat modeling will ultimately provide hundreds of hours in savings within the realm of exception/remediation management as well as change control requests that formalize any and all remediation tasks.

3. *Collaborative Approach:* Security risk assessments have historically taken an adversarial approach to both finding and addressing security risks in application environments. Threat modeling workflows foster more of a collaborative approach since they include all constituents that are normally a part of the remediation process. Via threat modeling, these key members are able to truly appreciate how existing application flaws translate to vulnerabilities that can be exploded by defined attack patterns. As a result, teams work together and learn much more about application security compared to simply being told to remediate unclear issues. This is currently the sentiment felt by most IT professionals (in development or system administration). Traditional IT professionals truly wish to understand the viability of how vulnerabilities foster exploitable attacks. The limited direction and guidance for corrective actions on hosts systems and software applications, however, leave most feeling that they are expected to automatically understand and quickly correct obvious security holes. Part of this problem is attributed to the poor guidance provided by security professionals to both information owners and asset custodians. Along with this intrinsic flaw is the antagonistic rapport between security professionals and those actually

delegated to address remediation efforts. Application threat modeling revolutionizes this approach by tackling two key fundamental flaws: (1) the timing in which vulnerabilities or configuration gaps are communicated and (2) the manner in which they are communicated. Namely, under the threat model approach, security professionals, and IT professionals work together to identify, validate, and rectify vulnerabilities and configuration flaws that introduce risk scenarios depicted by plausible attack patterns, as shown by the model. Needless to say, the unison approach in application threat modeling is refreshing and far more strategic than the current divisive ways that security flaws are identified and queued for remediation.

4. *Building Security In:* Contrary to security requirements previously established and socialized by separate and adverse groups (in either security governance or security architecture), security requirements now become an innate part of software development. Coupled with developers who would much rather know what to build in first than fixing bugs postproduction, building security in is a philosophy inherent in any secure software development life cycle (or secure development life cycle) – a highly recommended foundation for application threat modeling. Threat modeling could thrive in the absence of an S-SDLC/SDL process; however, it would be activated during the pseudo definition and/or design phases in which an application is being contrived. The presence of S-SDLC/SDL-IT efforts does award application threat modeling the proper context to operate within, versus a more ad hoc development culture, which would not properly assign responsibilities in various processes depicted by a threat model. For example, who is responsible for creating the proper attack library to be used within the threat model? Who will perform the various data flow diagramming exercises and what application boundaries will they encompass? Who will enumerate the actors, assets, and data sources that are applicable in the threat model? The answers to these questions are more streamlined within various phases of an S-SDLC/SDL-IT, or accomplished more haphazardly within an ad hoc development methodology. In either case, application threat modeling introduces attack considerations during a time in which functional requirements are being designed and outlined. Threat models help to determine attack vectors, inherent vulnerabilities (attributed to employed software or platform technologies), as well as an understanding of high-impact application areas that need to be protected. Incorporating this knowledge incorporates the premise of building security in and furthers the rationale for employing application threat modeling for key business applications.

The aforementioned points simply touch on a comprehensive list of points for a business rationale for threat modeling. More targeted benefits, appropriate to various security functions (operations, emergency response, risk, etc.), can easily be derived from these four points as well as others not mentioned. In the following section, we will expand on and correlate multiple business and security use cases. We will expand upon application threat modeling's ability to influence improved application

design – yet another rationale for which enterprises should further consider adopting application threat modeling.

Improved Application Design

Application design has been more of a conceptual idea versus an actual work effort funded by most IT organizations. This may explain the poor state of application security that we find ourselves in, or at the very least serve as one of its contributing factors. Even when implemented, application design considerations always seem to be one sided or built primarily around software features, diluting other variables that should influence the overall application design, including key business, IT, and *security* objectives for the application. In recent years, security groups have slowly been allowed to provide input to application design, but the effort is still scarce, spotted, and inconsistent at best. With the fruition of S-SDLCs (*Secure Software Development Lifecycles)* and Microsoft's SDL-IT (Security Development Lifecycle) Methodology, a stronger security voice will hopefully continue to grow over time and build a rationale within corporate IT boardrooms.

If a business rationale for application threat modeling is going to take flight, metrics have to be incorporated into any given threat model. Although metrics can be a key ingredient in building a business rationale for application threat modeling, the criteria in which metrics are understood and utilized within the vernacular of IT and business groups needs to be properly defined. For example, we can migrate over many traditional security risk variables that include single loss expectancy (or annualized loss expectancy), attack probability percentages, business impact levels, asset value, cost of countermeasure, and more. As expected, these variables will ultimately vary in importance and use across various organizations given their preferred set of metric values that are consistently monitored, either formally or informally. Overall, metrics for application threat modeling need to encompass the following requirements (as shown in Figure 1.2).

More guidance on metrics and threat modeling will be provided in Chapter 8. For now, simply consider metrics as a valuable by-product from application threat modeling. Improved application design will provide the consistency across any application environment in order to repetitively extract metrics. The following sections reveal qualities in software applications that are fine-tuned via the procedures applied from application threat modeling. Just some of the application-related traits that act as beneficiaries from the structure and analytical rigor of application threat modeling include factors related to application scalability, support of application components, and information/application security. Each of these three areas encompasses several factors within traditional IT objectives as well as goals in information security, further illustrated in the following sections.

Scalability One key aspect of improved application design is the ability of the application environment to accommodate changes, such as future business needs, infrastructure, and security requirements. As all of these factors may require code modifications, the impact (whether good or bad) to scalability is ever present.

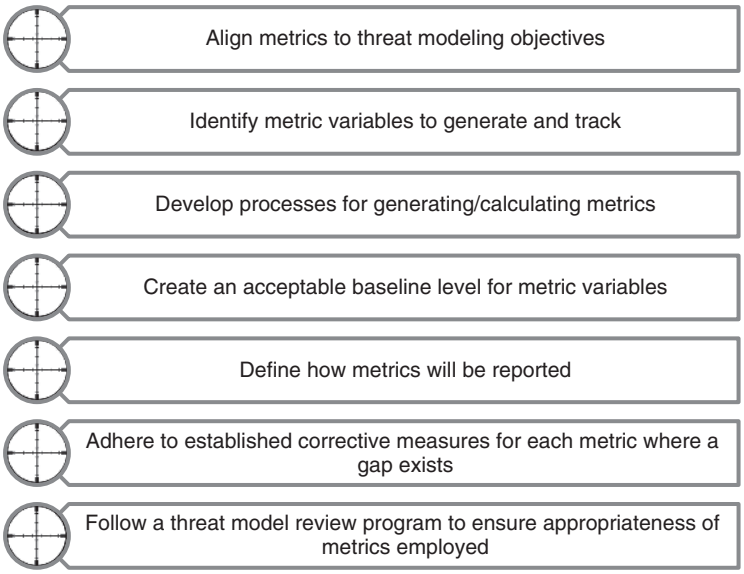

Figure 1.2 Developing Metrics in Threat Modeling

The ability for application architecture to be open and adaptable demonstrates a strong business case for application threat modeling beyond its security benefits. The thought that application threat modeling could provide direct benefits to application scalability may seem far fetched, but not if one unravels the layers that comprise software scalability. Microsoft's online Visual Studio Developer Center does an excellent job of depicting the key factors that impact software scalability. Figure 1.3 provides a graphical representation of these influential variables to software scalability.

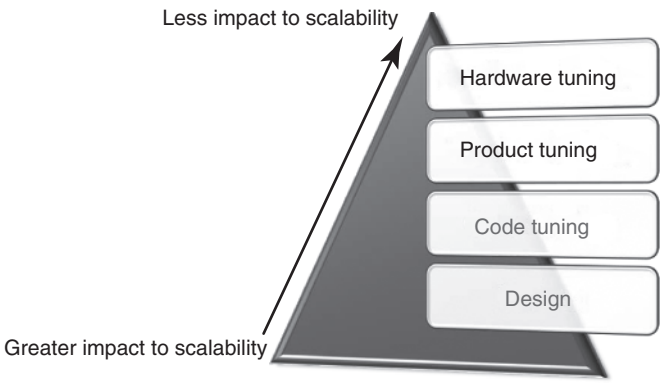

Figure 1.3 Development Factors Affecting Scalability

Design and code tuning efforts pose the greatest threats to the scalability of a software application. Taking this and today's security remediation efforts into account, modifications to code bases (code tuning) or application reengineering (redesign) efforts to incorporate new security countermeasures, such as input validation or error handling functionality, may unknowingly undermine any level of scalability that a given application may have had prior to such changes. A major reason is poor regression testing that encompasses all possible use cases that were initially tested during the first major roll out of a software build. Security changes implemented after the fact may ultimately resolve security gaps found by traditional security scans or assessments, but their objectives are simple and isolated. The reality is that security code modifications today are *quick and dirty* even when conducted through a formal change control process. Change control in most organizations has become so ritualistic that many of the considerations for how changes can affect a software environment are settled in a conference room instead of via a formal model. An application threat model provides the medium. It begins by addressing or readdressing the business objectives of the application and filters its way down to specific use cases, possibly impacted by the newly introduced security countermeasure or control. Additionally, it focuses on permission sets that may have been awarded inadvertently through design changes or code tuning. Since application threat modeling essentially *walks through* software application components (assets, communication channels, data repositories, and permission sets), a smaller degree of risk exists for when changes need to take place, thereby sparing possible setbacks in software scalability. Threat modeling essentially provides a higher degree of rigor in the analysis needed to determine adverse impacts to code tuning or software design changes. Threat modeling's differentiator is its systematic approach for breaking up the application security analysis into a hierarchy of key components, beginning with business objectives and ending with proper countermeasures for security gaps. In between is analysis to data sources based upon business impact or criticality levels, communication mediums, permission sets, plausible attacks, and clearly defined APIs. As a result, any considerations for design modifications and code tuning can take part within the boundaries of a threat model to ensure that previously defined functionality and objectives for application scalability are preserved and retested.

A more obvious relationship exists among the two worlds of scalability and security: nonscalable software can introduce future and serious vulnerabilities to software applications. Let us take the following scenario of a growing and profitable online retailer whose business focuses on ergonomic furniture. After years of perfecting their online store to reflect their vast inventory of ergonomic office furniture, primarily focused on the commercial sector, they are getting many inquiries from the federal government on their service line. In an effort accommodate this change quickly, project managers push new application requirements to development. Developers will in turn churn out new code to accommodate the desired changes on the retail portal. At this juncture, an effective application model is critical, particularly when ensuring that security controls are present. An application threat model therefore provides a framework where not only security countermeasures can be developed, but also processes related to continuity of service and scalability can be preserved by

the manner in which modifications are validated against application intradependencies. As a result, a heightened level of application design adds further rationale for employing the use of application threat modeling. Most importantly, security strategists will be able to recommend (with greater ease) what, when, and where security countermeasures should be incorporated.

Support Supporting software, such as any other IT-related process, must be properly aligned to a business objective. Such a lofty, idealistic goal may seem impractical if all support efforts will be validated against a broadly defined business or IT objective. Ensuring that support efforts on software applications are in alignment to these objectives, however, will ensure that not all supportive product efforts deviate from principal features of an application. If a process for supporting code modifications or application design changes does not revert to an initial blueprint of business objectives, the supporting code modifications can mutate into fractured and disjointed support efforts. Essentially, the faulty action that may result is scope creeping in supporting software. Good and even excellent ideas can easily take a quick turn to spawn unintended features or functionality. An application threat model will not catch such deviant actions until functions or features are reviewed from within the threat model and found to be discordant with defined business objectives as defined for the application.

What does support mean in this context of application development? Key members at the focal point of *supporting* software, examples of their related work efforts, and the benefits reaped from application threat modeling (ATM) are summarized in Table 1.5.

The proliferation of modular development efforts today makes supporting any application-related modifications nearly impossible without a proper framework. Application threat modeling is not aimed to be a replacement for proper application architecture and product management. However, since its process is embedded within the review of software features and functions, it provides an ongoing check

TABLE 1.5 Threat Modeling Benefits for Various Roles

Support Role	Responsibilities	Benefits From TM
Developer	Make changes to source code based on new or revised functional and security requirements	See related impacts from support-related changes to the application threat model
QA engineer	Validate new code through test cases	Understand the severity of application components and adhere to security test cases
Support personnel	Address questions related to the application's features and functions	Have a holistic reference to the application, from a security context as well as a feature/function point of view

for new features that may stray from intended objectives. Added functionality is a security risk because it typically introduces new interfaces, which may or may not include a new set of privileges for data access, among other types of application use cases. Improperly managed software modification (either from code tuning efforts or application redesign) can introduce tangents in functionality, which in turn introduces new doorways for attack vectors (i.e. – new forms, data interfaces). Essentially, supporting new code requires newfound oversight for secure coding practices, security architecture, and secure interfaces. Application threat modeling – an absolutely necessary security framework for addressing application risk on all of these levels – fosters improved application support by providing a context for new features and changes to abide by defined business and IT objectives. Unsanctioned features give way to process deviations in support operations, which are only as effective as the scope of features in which they are trained or introduced to support. New features or changes to an application, if not properly corralled back to operational project managers, will ultimately slip through the competency of support personnel who find such foreign features difficult to support. Moreover, application threat modeling, as a qualitative security process that is in line with validating newly developed or altered code, is positioned to identify anomalies in functionality and features, and then communicate outliers in application changes to support. This will preserve consistency and knowledge base in supporting the application.

With multiple parallel development efforts taking place, it is easy for code ownership to get lost amidst a sea of domestic and even offshore developers. However, improved application design can result from application threat modeling via its organized assembly line approach addressing multiple functional components as part of the application security analysis. Given most fractured development efforts, application threat modeling pieces together the various platform, database, network, and software-related components, which are all relevant support vehicles for change at some time in the future, thereby providing an excellent understanding of the application's design. In doing so, applications can be better supported in the future from various perspectives, including QA efforts, support operations, IT audit, project management, and software development. Application threat modeling provides an architectural view to support personnel in understanding how various application components (Web Services, Databases, Web Servers, Applets, etc.) interact among other application areas. This inherent holistic approach allows greater introspection to support the application by both developers and support personnel as threat models provide both high level and intimate details on an application's functionality. Lastly, support personnel at any level will be able to refer to deliverables or artifacts from an application threat modeling exercise as a key point of reference for understanding the following critical aspects of an application environment:

- Criticality of the software application
- Functional requirements as they relate to defined business objectives
- Security countermeasures incorporated into the application
- Type of data managed by the application

Although many will inevitably argue that it is not the place for application threat modeling to provide any level of blueprint for an application, the process does provide an updated overview for an application's various components, *particularly from a security context*. It cannot be emphasized enough that application threat modeling *is not* being taken out of context when it is depicted as a benefit to support operations for software applications. Ultimately, application threat modeling still preserves its security-focused objectives via improved application design by enhancing support efforts in software application:

- Elevates support teams' knowledge of security provisions, as identified by the application threat model:
  - Features related to access control
  - Controls related to confidentiality, integrity, and availability
  - Countermeasures that ward off spoofing, tampering of data, repudiation, information disclosure, DoS, and elevation of privileges
  - Superfluous features or functions that extend beyond objectives as defined within the application threat model
- Fosters a healthy validation of what features and functionality are actually to be developed. Also helps to limit out of scope software features that impact the following:
  - Stray from business objectives
  - Deviate from core competencies of the software application or environment
  - Introduce security risks via the expanded scope
  - Augments the scope of knowledge and expertise that is potentially required to support the application

Security Application Threat Modeling yields improved application design, driven by security efforts via strategic, streamlined, application hardening efforts, ideally all within the context of a secure software development process. Application threat modeling provides an architectural advantage over more traditional security assessments on software applications through the use of data flow diagramming techniques and application walk-throughs. It also embellishes traditional IT architecture by incorporating functional requirements for service delivery, continuity, and scalability; all obtained by threat modeling's collaborative workflow that fuses security analysis with traditional IT architecture and software development.

The key security contrast between application threat modeling and more traditional application assessments (achieved via automated scans or qualitative assessments) is that identified risk issues are derived from attack possibilities that are unique to the application environment and not solely to the discovered vulnerability. Motivational factors for launching specific types of attacks are conceptualized in a library in order to provide the most likely description of an attack landscape for an application. In essence, application security today does not truly map out specific attack scenarios for given vulnerabilities or series of vulnerabilities associated for an assessed application. As a result, an incomplete portrayal of risk is presented to information owners

for remediation. Unfortunately, the owners do not understand the nature and likelihood of possible attack scenarios to their particular application and what likely attack vectors would be launched to introduce these risks. Supported by vast attack libraries, threat modeling provides a process for multiple security threats to be addressed, each encompassing a set of possible attack patterns, and corresponding vulnerabilities. Security professionals can walk-through an attack that specifically relates to an application use case, represented within the threat model, which is invaluable to the process. Threat modeling goes further by addressing weaknesses in business logic that should be reconsidered and IT components that may also introduce additional attack vectors (at the platform level or via third-party software). These security-weak areas are discovered prior to a production release or production build. Ideally, these efforts should take place within the early stages of an SDLC, thereby allowing remediation of vulnerabilities to be addressed prior to production.

Effective Remediation Management

The English saying of "an ounce of prevention is worth a pound of cure" is very appropriate when applied to remediation management in security risk management. Since application threat modeling is best applied within the early stages of the SDLC, it naturally adheres to this preventive philosophy and truly enhances remediation efforts by reducing the amount of time required for remediating software vulnerabilities as well as by correcting security gaps prior to introducing the application to end users. The following is a brief list of key factors that reveal how application threat modeling triggers effective remediation management.

1. Defines security requirements to be baked into the application
2. Incorporates security requirements into application design
3. Fosters the development of security countermeasures as features
4. Allows the development of security test cases

The aforementioned factors are far more difficult to achieve after an application has been developed, for reasons previously mentioned within this chapter. Besides limitations in time and availability, the process of reactive remediation efforts forces development teams to remediate production software (in a test environment) that may be *n* versions behind the current set of software. As a result, developers may not be too inclined to address software vulnerabilities in older versions versus alpha or beta releases that are currently being developed.

Adding further complexity to late remediation is the decentralized manner in which many developers write code – each focusing on a specific aspect or module of the application environment. This may force the necessity of an application architect or technical project manager who can oversee remediation efforts across all vulnerable areas. Since application development generally encompasses the involvement of multiple developers or even development teams, understanding an application and its environment may prove challenging, time consuming, and

ultimately ineffective. Doing so at this junction may not include other key members who affect the integrity of the environment, such as system administrators, network engineers, or members from IT architecture who may have valuable insight and knowledge on how an application was built, behavior and reasons for any applicable data interfaces, technical/security exceptions made, and more. The likelihood that all (or even some) of these members will have time to address security vulnerabilities, particularly within a relatively similar time frame, is near negligible. Even if achieved, the window of time is small for both their initial feedback and corrective actions (programmatic or configuration related).

The need for evolving beyond current remediation management efforts in security is timely, given the increased need to reduce application security risks. Regardless of application environment (web, mobile, client-server and/or fat-client), threat modeling has its use and benefits, as we will later see in future practical applications of various methodologies and tools. Most methodologies can be applied parallel to maturing SDLC or SDL-IT processes. The key challenge is whether there is a formal SDLC process that repeatable application threat modeling efforts could become embedded within. Statistically speaking, most organizations do not adhere to any form of SDLC methodology or, if implemented, they are in an early stage of adoption. A formal SDLC process is a prerequisite for implementing application threat modeling as a repeatable security process; however, a fully functional QA process may also anchor and support a developing threat modeling program as well.

In some instances, an SDLC process is not uniformly adhered to across all business units involved with developing business applications. As a result, disparate security levels may exist across implemented application environments that share data. Applications disassociated with the application threat modeling program may introduce APIs that actually serve as ripe attack vectors. Internal application domains often use trusted authorities across application environments, thereby exacerbating the disparate security posture between the two application domains. This is important to consider when and if minimal gains in improved application design are witnessed, subsequent to the implementation and use of application threat models.

SUMMARY

As reflected in this chapter, there are several factors that account for the business rationale for threat modeling. These factors are both process and technical in nature and extend beyond the benefits of traditional application risk assessments and vulnerability assessments today. Although traditional risk assessments and vulnerability assessments provide ways to identify risk issues, they do not ultimately translate into new security requirements for the existing or even subsequent application development efforts. Conversely, threat modeling is able to address what security requirements must be present across multiple levels of the application environment as well as identify new attack vectors and potential exploits during the testing and validation efforts within the threat modeling process. All of these efforts take place prior to code migration into higher application environments, thereby reducing remediation

efforts and risk exposure levels. Additional factors for its implementation relate to the following:

1. *Outline of Application Use Cases:* Use case scenarios have never truly been tracked or managed from the inception of an application's life cycle. As a result, the overall intent of use for an application may encompass functional aspects that were never meant to be pervasive over the life of the application. Use cases help define exploitable misuse cases, most notably through the rise of attacks based upon the misuse or abuse of application business logic. Threat modeling brings to light the need to address misuse case scenarios from within the testing stages of the SDLC or SDL-IT process as well as the necessity to disable features that should no longer be made available to the intended and unintended user base. Threat modeling brings greater focus on both use case and misuse case scenarios within an application.
2. *Discovering Application Security Land Mines:* Application walk-throughs are virtual simulations of an application's functionality and greatly assist discovering errors in business logic or vulnerabilities in the code. Walk-throughs are intended to be very thorough and aimed at identifying how object or resource calls can be compromised at various points of the application. This simulation allows a well-defined attack tree to develop and serve as a baseline of attacks for future threat modeling exercises.
3. *Comprehensive Data Security via Data Flow Diagrams (DFDs):* DFDs are nothing new to software development, but they do provide a fresh perspective to mapping out design and coding flaws within software applications. Essentially, DFDs provide a visual on how data moves between functional points within an application environment. These exercises provide insight into what actions against data are happening at various points and if additional controls for protecting the integrity and confidentiality of the data should be applied. Similar to application walk-throughs in the sense that they are thorough and comprehensive to the various features of a software application, DFDs are different in that they focus more on the data object being called than the functionality and parameters of the caller resource.

Most notably, the reduction of software vulnerabilities reduces remediation time and efforts. Less time translates into less cost. In order for threat modeling's business rationale to evolve from the theoretical to the practical in this area, key metric values must be collected and trended over time. These metrics should include residual risk levels, loss expectancy ratios, number of vulnerabilities for beta versus production versions, remediation time, and so on. These values will help provide choice metrics that can be used to sustain the business value of such threat modeling efforts. Application threat modeling embellishes much of what has been lacking in application design by fostering a greater intimacy with application requirements across business, IT, security levels, and beyond.

THREAT ANATOMY

> "*A little while ago, the Pentagon demonstrated in an exercise that it was possible–even easy, actually–to hack into the power grids of the 12 largest American cities, and to hack into the 911 emergency system, and shut all of those off with a click of a button. Now, that isn't somebody getting shot, and you don't see the blood coming out of the body, and the body collapsing on the ground. But I can assure you, tens of thousands of people would have died.*"
>
> *PBS Interview with former iDefense CEO, James Adams*

Earlier in this book, we discussed attack motives in order to answer the question of why attacks occur. The range of answers to the why question are vast, but with a strong degree of overlap among various key factors. Before we delve into the answers on how attacks are planned and launched in cyber warfare, let us quickly revisit the list of drivers that propel white hats to black hats, hobbyists to criminals, and script kiddies to wanted cyber felons.

No matter how much we have advanced technologically, the elements of war and attack are still age-old intrinsic human sentiments rooted in hatred, greed, envy, or simple idle curiosity. This chapter aims to dissect the elements of cyber threats and attacks for the purpose of selecting the proper countermeasures.

One motive not represented in Figure 1.4 is the motive geared toward creating a diversion for a simultaneous or delayed threat or attack. These become more sophisticated and may or may not encompass a clear motive. More sophisticated diversion attacks seek to create a false motive for which opposing resources can take time, money, and effort to investigate, while core threats and attack plans continue to evolve. Now we build upon this notion to dissect the elements of attacks within an application context and related threat model.

In this chapter, we will dissect cyber-related attack patterns. We begin by understanding the progression of attacks with the encompassing threat and how understanding cyber threats can help a security professional to identify probable attack plans. Building upon the brief recap on attack motives, an understanding of threats to an application threat model is the next sequential step to see what attacks comprise an overall threat. It is important to understand the hierarchy of terminology used thus far, particularly *motives, threats,* and *attacks,* as they each represent both a unique and interrelated component to the application threat model. In software applications, a *threat* is very much like *risk* in that it will never be zero or nonexistent. There always is a degree of *risk* primarily due to the fact that *threats* are always present within or around an application. With enough *motive*, *threats* serve as mobilizing agents to conduct *attacks* against software environments. Table 1.6 provides a threat stack that emphasizes the hierarchy and interrelationship between these factors.

Motives, software/platform vulnerabilities, and risk levels stand independently; however, threats are comprised of viable attack patterns. Devoid of any probable attack, a threat becomes near negligible and is only retained as a theoretical or possible threat scenario. Table 1.6 reflects the interrelationship between attacks and threats and the dependency in which they coexist. No threat equates to no possible forms of

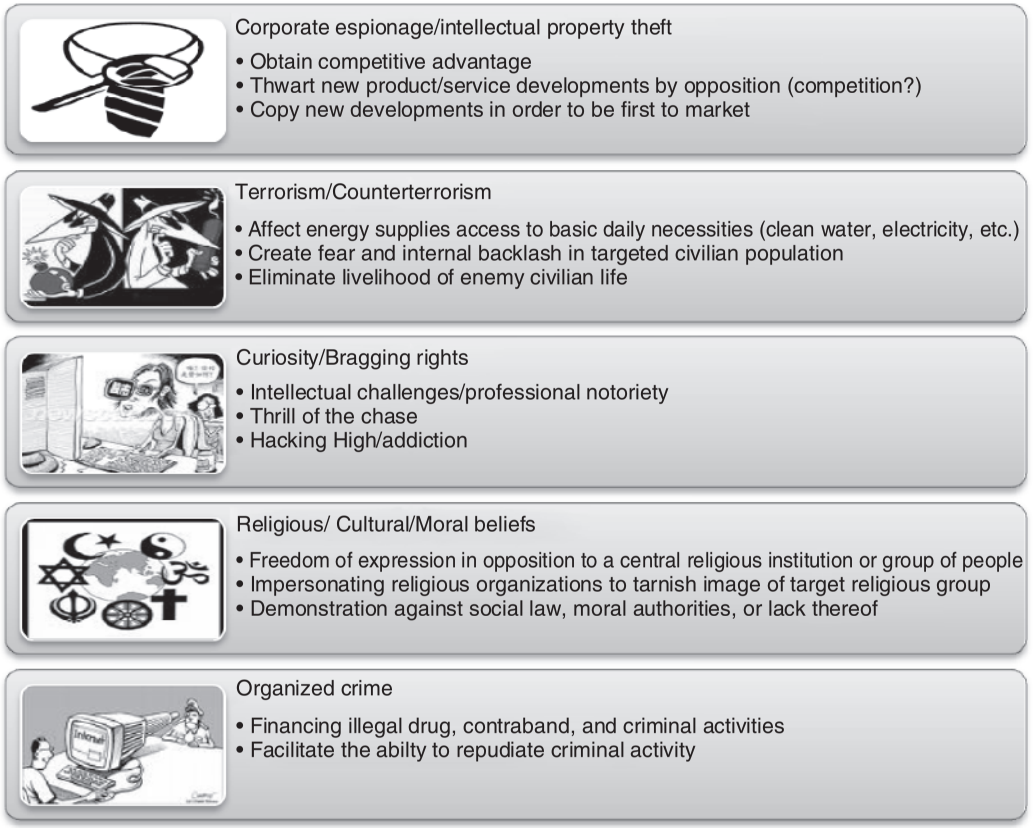

Figure 1.4 Cyber Crime Motives

TABLE 1.6 Threat Model Stack

Threat Model Stack
−Motive +Threat(s) −Attack(s)
• Probabilities • Vulnerabilities • Assets
+Risk

attack. The absence of an attack or series of attacks reduces a threat to only conceivable or theoretical threat levels. Although reflected by any application threat model, it is important to note that a given threat model is evolving or only valuable for a defined period of time as the sophistication and plausibility of application-based attacks will ultimately evolve over time, as will the other components of the threat model stack.

The Threat Wrapper

Threats' complexities lie in bundling varying degrees or attack types, vulnerabilities, and impact levels, and there is variation among application types. For this reason, we will explore a handful of threats and varying types of application environments, and dissect the encompassing attacks that could accompany them. First, let us look at a very simple threat model that expresses a highly generic flow of input/output from a user base, between two trust boundaries, to a target information source.

Unrelated to any methodology, and assuming illicit data access is the primary motive, the foremost question should be: How can an attacker complete their objective? Now that the threat of data compromise is assumed, the focus becomes where and how the threat will be carried out. Revisiting our data flow in Figure 1.5, we have to identify how data sources can be leaked via the boundaries of the application environment. In this case, the trust boundaries are neatly drawn between the client or

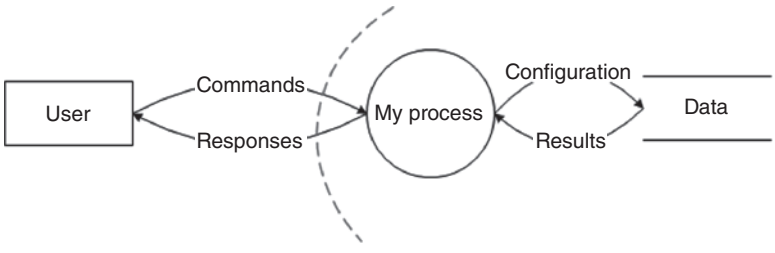

Figure 1.5 Simple Data Flow Diagram supporting Threat Model

user environment and the application environment. At this point, although we have not defined the business or IT objectives that should provide governance, we are proceeding to understand how the imminent threat should be addressed. Incorporating these objectives ultimately allows us to understand the appropriate countermeasures that equate to a formula reflecting the probability of each attack identified, the business impact if successful, and costs associated with implementing security control measures. In this case, we assume that all threats will be mitigated to the best of our ability.

Upon understanding the objectives of our threat model as well as all plausible motives for the identified threat, we need to evaluate the threat landscape. The threat landscape is comprised of target areas (client, server, middleware, or proxy), communication channels (wireless, Ethernet), layer seven[2] protocols (SMTP, SNMP, HTTP), physical security considerations (easily accessible server closets), and services probed to be present across the application environment. With these variables in mind, the previous threat model can now be updated with an overlay of a hypothetical threat landscape. Items represented in red reflect potential malicious misuse of the application environment.

Referencing the aforementioned figure, we see how a slightly more evolved threat model can manifest the components of possible threat scenarios against a generic application. In reality, the threat model may reflect any number of motives, as discussed earlier in this book, and those motives might shed some light into the types of attacks that are most likely to achieve a given motive. In Figure 1.6, we begin to understand some of the components enveloped within a threat. Motives trigger actions on behalf of malicious individuals or irresponsible employees to create some degree of threat. These threats may be geared toward target assets or information sources, as part of their objective and will ultimately rely on intel to discover software vulnerabilities or misconfigurations to exploit via attacks. As the threat traverses across public, semipublic, private, and restricted application zones, other variables related to threat begin to take form such as probability of successful exploitation, business impact of compromised business data, presence, or void of security countermeasures, and much more.

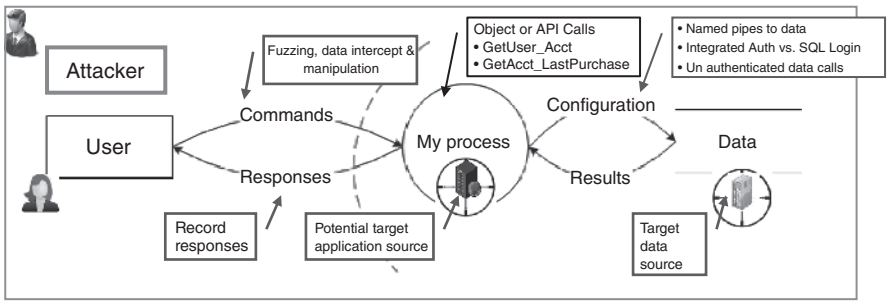

Figure 1.6 More Evolved Data Flow Diagram supporting Threat Model

[2]Related to the OSI model.

Understanding threats begins with understanding the attacker and the available information and expertise they may have to conduct their targeted attacks. Most times, an attacker's identity or the profile of an assumed attacker cannot be derived until after the attack has happened. The timing in which this information is obtained does not undermine its value; it can be used to create an attacker profile for future events, particularly if their actions are recorded in the application server log, network logs, or at the platform host level. Most private or commercial organizations do not have an attacker profile database; however, government or military IT operations may find this worthwhile in order to predict attack patterns based on commonalities in attack patterns. Banks and financial institutions may also find this essential.

Beyond this type of attacker profiling, threat classification provides the most common form of analytical and preventive defense that any organization can begin as a formal security operations effort. Threat classes are preventive in the sense that they help classify types of threats from any security control that provides both alerting and logging of actions taken against a system. Threat classes help to create "bins" for organizing attack data into decipherable forms of attack. Injection attacks, elevation of privilege attempts, and DoS attacks all become organized into appropriate classes for analysis and reporting. Coupled with external threat feeds, any organization has the ability to prioritize concretely their security controls for the *short term*. An emphasis on short term is made here because attack patterns and exploits that are en vogue may be blasted across target sites few months to several years. Overall, the idea is to have both a process and technology that can aggregate and classify threats appropriately.

From the threat classification efforts, an association map can be made by correlating attack scenarios and vulnerable application components. Additionally, both the possible exploit and vulnerability can be mapped back to the application within the threat model to obtain business impact values and risk levels. At this level, even before taking a deep dive into the practical logistics of the attack, such as attack vector, exploit, or associated vulnerabilities, obtain a high-level picture of risk and business impact, which may help formulate preliminary risk strategies. After all, the end goal associated with any threat model should be to mitigate risk.

Brief Intro to Threat Classification Models

Some threat classification models include STRIDE and DREAD – two Microsoft-originated threat classification models focused on identifying business impact and risk, in varying degrees. Additionally, the Web Application Security Consortium (WASC[3]) periodically revises its threat classification, which is a great technical reference for grouping various types of threats by their technical nature in lieu of any business impact or risk model. The WASC's listing is more of a technical briefing of the latest web application-related threats, and less of a threat classification model. A model could easily be built, however, from the classes defined within this periodic reference, as can one be built from the Open Web

[3] http://www.webappsec.org/.

Application Security Project (OWASP[4]), which also releases a top ten list of threats aimed at web applications. The OWASP top ten listing is updated every few years and reflects the most prevalent threats to web applications and is an excellent start for a technical-based threat class model.

Several threat models may also be built with the help of product-based security solutions from both open source and commercial grade products today. Many network- and host-based solutions have threat intelligence modules or feeds. Security operations centers then analyze and aggregate the provided data to understand what threats are traversing various types of networks and interfaces over a defined period of time. A security incident and event monitoring solution within an organization, or a managed service program where companies send logs of alerts and events to a security cloud for threat analysis can provide this information. More autonomous organizations can operate in a self-contained manner by leveraging threat feeds from large security organizations that leverage deployed security products and monitor networks from around the world. This gives these vendors great visibility into active threats as well as provides trending data for such recorded events.

In the end, threat classes are useful for categorizing vulnerabilities and attacks identified by the threat model. Figure 1.7 provides a visual synopsis of how threat classes can organize a laundry list of attacks and vulnerabilities. This simplified figure depicts how threat classes cannot only encompass elements of the threat, such as attacks and vulnerabilities, but also the countermeasures or controls that mitigate their associated risks.

Vulnerabilities – The Never-Ending Race

Dissecting any given threat reveals a number of vulnerabilities that serve as windows of opportunity. Without them, acting as a threat agent proves to be a lot more difficult and less rewarding given the decreased likelihood for success. Hackers and cybercriminals value their time as much as anyone else does, and if no clear vulnerabilities in process or technical controls are present, it is very likely that they will threaten other information doorways.

The evolution of vulnerabilities has migrated in overwhelming numbers from platforms to applications, making vulnerability management exponentially more difficult to track and manage simply due to the sheer number of applications that are present across enterprises. As a result, threat modeling is a bit more complex, needing a more extensive and up-to-date vulnerability listing.

As part of an application's threat model, an inventory of up-to-date vulnerabilities is key. Vulnerabilities can be linked to asset and architectural elements in the threat model through the inventory. These elements include both software and hardware assets and their related software or firmware that can be misused by released exploits. Considerations for zero-day exploits should also be made within the model, but they are more difficult to predict. Automated and continuous vulnerability scans should provide a good amount of vulnerability information quickly for aggregation,

[4]http://www.owasp.org.

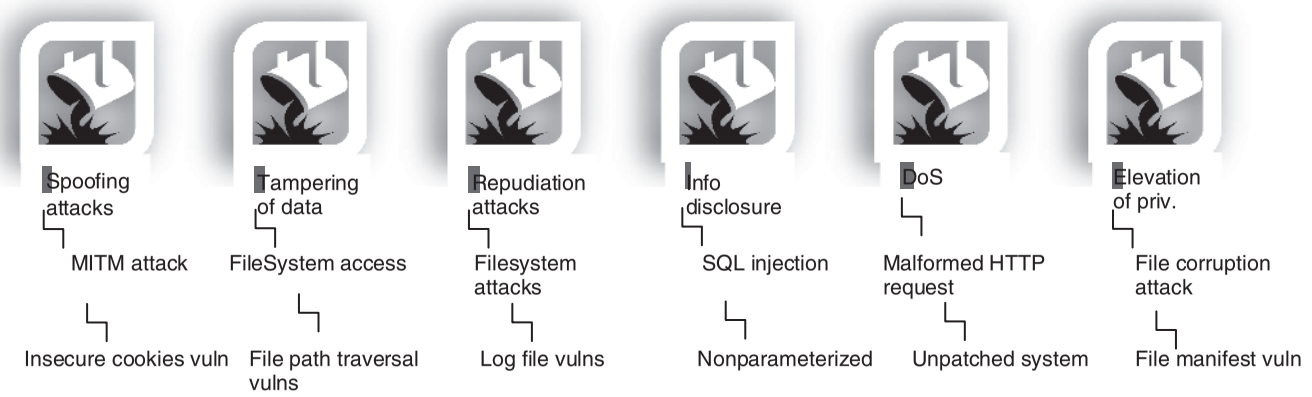

Figure 1.7 STRIDE Threat Classification Visual Example

Figure 1.8 Incorporating Vulnerabilities within the Threat Model

analysis, and use within the threat model. The following suggested workflow reveals the necessary steps needed to leverage vulnerability data within an application threat model (Figure 1.8).

The aforementioned diagram assumes the following as part of incorporating vulnerability details into the application threat model:

- A proper application scope has been defined, limiting the threat modeling analysis to logical boundaries of the application environment.
- Sufficient insight into vulnerabilities can be obtained on a periodic and regular basis to evolve the threat model's risk landscape for the application.
- The expertise to identify false positives within a vulnerability assessment is available as a repeatable process.

Apart from product-based security solutions that specialize in vulnerability scanning, multiple external data sources help any security operations group to build a "living" vulnerability database that can be used and correlated to an asset inventory of both platforms and software. SecurityFocus™ provides a vulnerability listing that encompasses multiple vulnerabilities for both open and closed platforms and software types. The National Institute of Standards (NIST) also provides a National Vulnerability Database (NVD) that includes a free listing of up-to-date vulnerabilities across multiple platforms. NIST's site lists all vulnerabilities by their Common Vulnerability and Exposures (CVE) reference, which is a useful data identifier that allows for

some interoperability among security products and solutions. The Federal Government encompasses CVE as one of its criteria for the Security Content Automation Protocol (SCAP). More references to SCAP are included later in the book; however, this notion of common security language is key for application threat models and any security solution, particularly as these standards evolve and become more widely adopted within security products. Their intent is aimed at receiving security data from multiple sources in order to have a complete and accurate vulnerability and attack library.

As part of the threat components, multiple vulnerabilities may be relevant, which require countermeasures and risk mitigation. The following tree focuses on a hypothetical spoofing threat to a utility company's use of a Smart Card to gain access to the sensitive, central operation center. Although this example embellishes aspects of physical security, its simplicity helps to define the various components of a threat revealed thus far, namely, a definite threat, series of attacks, and software vulnerability. Ultimately, this hypothetical example aims to dissect a particular threat to the level of isolating related vulnerabilities that should be encompassed within the threat model.

GIVEN: Employees use the MiFARE Classic Smart Card to gain access to various control rooms where power distribution is controlled and managed. The data that it stores and transmits to physical receivers is related to authorized personnel.

THREAT OBJECTIVE: Gain illegitimate access to one of these control rooms by leveraging a legitimate key code from a Smart Card.

ATTACK VECTOR: Wireless transmission of Smart Card over the air broadcasts.

VULNERABILITY: The card uses a weak cryptographic scheme for encrypting data over the air (OTA). As a result, data-transmitted OTA can be intercepted and cracked.

ATTACK: An off-the-shelf reader can be used to query or probe the card for its information.

In a more prevalent attack scenario (e.g. web application, web service), a vast range of vulnerabilities and attacks should be itemized in order to map out all possible attack scenarios and corresponding vulnerabilities that would be used. Similar to how attack libraries should be built, a relevant listing of applicable security vulnerabilities should be tracked through their existences within the application that serves as the object of the threat model. This process is not easy to instantiate; however, it is foundational for any threat model to work properly since possible vulnerabilities will reveal the likelihood of various attack scenarios for an application. Given the exhaustive list of vulnerabilities, an underlying process to support technical reviews is foundational to employing a threat model. This does not mean that threat modeling forces the need for additional or available resources; it all depends on the number of threat modeling efforts that are conducted across an enterprise or business unit. As previously mentioned, vulnerabilities are typically discovered automatically via

an internal security operations group or managed service that provides vulnerability scans against application environments. From this preexisting and nearly parallel process, vulnerabilities found can be mapped to assets within the threat model as well as attacks within their respective libraries. The following figure reflects a short sample of how this process should unfold.

In the single vulnerability mapping that is accomplished in Figure 1.9, we see a single vulnerability that is mapped to both a subset of attacks (within a larger attack library) and the assets (either hardware or software), which the vulnerability affects. Ultimately, as the vulnerability is understood to be a material weakness for the application and ultimately the data it controls, risk mitigation efforts should proceed via code-related modifications or application redesigns.

Making logical groups is essential for the application threat model to efficiently use the vulnerability findings from preexisting and preventive security operations. This figure portrays a micro level version of the comprehensive level mapping that should take place among vulnerabilities and target assets. A more macro-level portrayal would encompass a large mapping tree that reveals a list of relevant security vulnerabilities to possible attacks, thereafter a map to affected software and hardware assets.

Threat class has been purposely left out of this and other examples thus far so that we may focus on mapping *known* security vulnerabilities to possible exploitable target end points and attack patterns. Predefined threat classification models such as STRIDE, DREAD, or Trike (an open-source threat modeling methodology) would successfully encompass vulnerability data presented from a preventive standpoint, meaning that discovered vulnerabilities are mapped to possible attack scenarios and

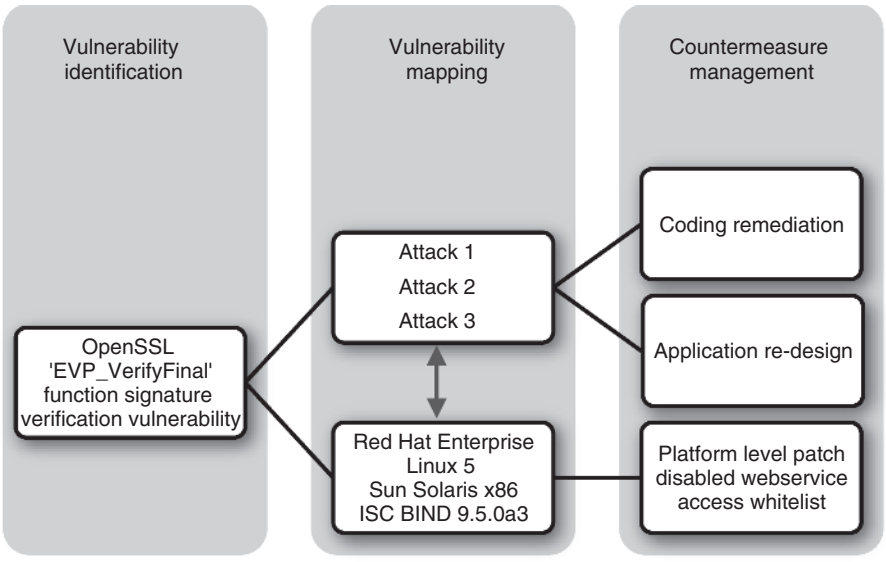

Figure 1.9 Vulnerability Mapping

then appropriately categorized. However, threat classes would be best derived less from preventive security processes, such as vulnerability management, but more from detective security measures, such as via security incident and event monitoring systems or threat feeds. Both reveal possible weaknesses for an attack within the threat model; however, detective security controls and processes reflect recent attack data that has taken place historically across the dark Internet abyss within the networks of internal application trust boundaries. This information *does not* replace but supplements the mapped information, linked back to possible attacks or exploits and affected technology assets. The notable difference is that a more precise threat categorization model could be developed for an organization versus one that may only have some relevant threat categories.

Attacks

Attacks are difficult to predict and understand uniquely. This takes us back to the motive discussion – something rarely addressed in information security and honestly not a traditional component to most threat models, although it does have a parent component to identify attack motives at the root node of an attack tree. At some point, however, a list of likely motives has to be maintained and correlated to information types to imitate the use of attack libraries within an application threat mode. Some governments are investing in such efforts to thwart possible attacks before they happen, recognizing that their adversaries are in the planning stages and waiting for an opportunity or particular data. Overall, it is a science of foreseeing the inevitable and the utmost damaging. *Counter-hacking units* have been developed in Great Britain to detect and counteract threats from Russia and China as well as many other countries (1–56). Part of what a counter-hacking unit does is study predictive patterns against government targets and private businesses with highly sensitive intellectual property. Great Britain's MI5 (Military Intelligence Group, Section 5), as well as the Singapore Intelligence Agency, have established counter-hacking units that are responsible for such efforts.

Profiling attackers helps to derive plausible attack vectors that could be sought to achieve such motives. In some cases though, the true motives behind an attack are not easy to decipher. In January 2010, Google Inc. (GOOG) reported that they were the victims of an elaborate attack against their infrastructure and that intellectual property was stolen. Within the same vein of communication, it was openly revealed that these attacks originated from within China, potentially organized by the Chinese government. The investigation grew when 33 other companies said that they were affected by the attacks and that information may have been compromised since the summer of 2009. One of those companies, Adobe Systems Inc. (ADBE), announced in early January 2010 that they also detected attacks from China against their infrastructure but declared that no information was compromised. As more and more details surrounding the attack surfaced, many security researchers involved with the actual forensic analysis released details on the attack vectors. Forensic experts and security researchers actually traced the attacks back to two key hosts that served as the

command and control centers. Exploits related to PDF attachments and IE flaws were cited as part of the attack vector (57).

Initially, a human rights inquisition was said to be the central motive for this and other attacks that encompassed comparable attack vectors, vulnerabilities, and exploits. However, parallel to the theory that these attacks were fueled by a persecution of human rights activists, the notion that communist China had more capitalistic intent, specifically profiting from IP (intellectual property) theft, clouded the true motive and intent behind the Google attacks. In deciphering the true motive (the basis for these attacks), the following questions should preface a threat model in order to gain a focus on target data, applications, and related infrastructures in the future.

- *Are the attacks interrelated?*
- *Is IP theft the true object of these attacks?*
- *Were these attacks aimed at probing intrusion detection, response capabilities, and protocols, along with any formal communication afterwards?*
- *Are these attacks diversion attacks – secondary targets used to occupy time and resources in order to conduct even more sophisticated attacks on primary targets?*
- *Is the true target for these attacks aimed at US corporations (Google, Adobe, etc.), government, or even citizens?*

All of these questions revolve around motive and allow a threat modeler to focus on the possible end goals for the attacker, namely the type of data being sought. Naturally, many may wonder why we even worry about motive since the possibilities are endless. There is a difference, however, between likely motives and possible motives. Within a threat model, identifying the likely motives may ultimately draw focus to a key piece of an application or application environment that is the central target. Attacks alone will not provide such clarity to the possible target source, particularly with more sophisticated layered attacks. Attacks are the means to an end and motives are the keys to understand the desired end.

The motives in the Google–China case are most likely both IP and human rights related. Given that these attacks have targeted many US companies and government agencies since 2005, China may be desperately seeking to supersede advanced and existing software and hardware technologies by building upon IP theft and leveraging it to offer new, "original" alternatives to its billion-plus population and the global market. Business Daily Handelsblatt in Germany writes:

> Behind Google's threat to cease business activities in China, one motive stands out: The company, whose business is that of collecting and storing highly sensitive data, must protect itself from being spied upon by a country which seeks to play a major role in shaping the next generation of Internet standards. Beijing is following a strategy meant to prove that an authoritarian regime can survive in the Internet age. But the Chinese are lacking expertise, which is why they are seeking access to protected source codes.
>
> (58)

Perhaps IP theft is the focal target of these attacks and they were disguised as persecuting human rights activists. After all, the negative rap that China has had on human rights violations has something that multinational companies and world governments have come to apparently accept via their tepid and inconsistent reactions. Due to China's continued prolific success and sustainable economic utopia, doing business with MNCs only to target their IP, is an image that would quickly dissuade many from speaking to Chinese government and businesses. Regardless, the previously mentioned attack motives still remain likely motives and help to identify target assets for their past, present, and future attacks. Considering motives at the inception of a threat model will help shape countermeasures and controls across data sources and related infrastructure. These targets serve as the assets, a formal terminology in a threat model that will be discussed later. For now, these assets require software developers and application architects to respectively code and design countermeasures within the application environment to safeguard these target assets. These actions reflect security disciplines related to data and process classification techniques, where data sources and business processes are identified, mapped, and classified for their business impact level to devise controls commensurate to their worth.

Identifying Attacks

Attacks carry out threats, while threats are driven by motives. Digressing into application-based attacks within a threat model will encompass a greater deal of structure and formality. Although understanding motives within a threat model is not commonplace, it has prefaced the introduction of attacks (within the threat model) to introduce a comprehensive visual of how threats become actionable via motives and access to attack resources and opportunities. Application attacks build upon motives in the sense that hypothetical attack scenarios and applied exploits are correlated to the targets of these motives. Stringing all of these elements together will ultimately improve the overall readiness of the security professional who must create a threat model for an application environment. Motives undoubtedly influence the complexity in attacks that are launched against an application. Some attacks encompass known security exploits that target vulnerable applications, while others are fueled by zero-day use cases. Layered attacks are even more complex, as they use all of the aforementioned, coupled with other characteristics that make forensics challenging (attack source(s), timing, and collaborative actions).

Understanding attacks within the context of application threat modeling requires common terminology that security professionals note so that they do not confuse the vernacular associated with the use and execution of application threat model. The following table provides a short yet important list of terms leveraged by the threat model, specifically attack-related components.

As shown in the list of terms to describe application attacks, the term *attack* has been dissected into many components that capture its characteristics. This level of analysis is essential to understand the behavior of each attack cataloged by the application threat model. Many of the previously listed terms may be synonymous to other terms referenced by threat modeling methodologies, tools, or frameworks. Attacks

within a threat model are adverse actions taken against an application or its environment for the sole purpose of sustaining or realizing a given threat.

Once all possible application threats are clearly understood, an attack tree encompasses all of an attack's characteristics, as depicted in Table 1.7. Building an attack tree involves creating a vast and comprehensive library of attacks or exploits that correlate to an equally vast and comprehensive list of vulnerabilities. The complexity of managing an attack library extends beyond its initial conception into its ongoing management and upkeep. The importance of an up-to-date attack library runs parallel to a well-maintained vulnerability management database. A broad attack and vulnerability library should ultimately allow an application threat model to address probable threat scenarios and underlying attacks and vulnerabilities. This laborious effort can be eased under the right environments, particularly within larger organizations. Security Operation Centers (SOCs), for example, may already be aggregating threat feeds and identifying repeated exploit attempts from outside and inside the company network. Additionally, such groups often administer vulnerability scans across the enterprise, which provide an inventory of discovered network, host, and even application-level vulnerabilities. From these large information sources, associations can begin among discovered vulnerabilities and attack libraries. The actual

TABLE 1.7 Taxonomy of Attack Terms

Term	Definition
Attack tree	A model that encompasses multiple attacks that may or may not be related to one another but that all support a given motive. Oftentimes interchangeable with *threat tree.*
Attack vector	A channel or path that encompasses an exploitable application vulnerability. Seen as the multiple hierarchical nodes that also encompass entry points in an application environment or system, which may facilitate an exploit execution or malware attack. Also known as an *attack path*.
Attack surface	Refers to the area in which an attack has the opportunity to introduce itself.
Attack library	A catalog of possible attacks that could be launched against an application environment. Used by security professionals to identify the likelihood and impact of attacks as well as possible countermeasures for such attacks.
Vulnerability	Preexisting weakness in an application component that allows the successful execution of an attack.
Attack (Exploits)	Malicious payload executed against a known or unknown vulnerability. Follows a many to one relationship.
Target (asset)	The focal point for a threat and the object of attacks or exploits.
Threat landscape	The logical surface area in which an attack or threat can be conducted. The threat landscape does not need to be continuous, meaning that threat components can be a part of different environments and not physically or logically connected.

repository of attacks will build the attack or threat tree for the application threat model. The tree itself encompasses a multitude of branches and nodes (or leaves) that describe associated vulnerabilities, attack vectors, and targets (assets).

Unlike traditional preventive security models that mitigate attacks by incorporating *best practice* guidelines, application threat modeling depicts probable threat scenarios along with their associated attacks. Given the proliferation of targeted attacks, threat modeling is an essential ally in thwarting their possibility of success. Even when addressing nontargeted attacks, the threat model lends strategic readiness via its attack library, which can correlate exploitative attacks to target assets and vulnerable hosts or networked systems. As a result, possible attacks or exploits need to be "tagged" and inventoried for research and applicability to software use cases, platforms, and networking services.

The following figure provides a visual representation of a simple attack tree. As demonstrated in Figure 1.9, a well-defined threat encompasses multiple attack branches or nodes, which in turn encompass targets (or assets) and their associated vulnerabilities. The term "assets" should not be misconstrued and solely relate to workstations or servers. Attack targets can also be related to network appliances, network devices such as Firewalls, Intrusion Prevention Systems (IPS), network or application proxies, content filtering devices, web servers, databases, and more. *Assets* are any exploitable hardware or software target for an attack or a necessary component to persist with a layered attack. The attack tree is used to visually represent the logistical manner in which single and layered attacks can be conducted against these targets. Apart from dissecting attack patterns and mapping them to assets and vulnerabilities, attack trees offer a conceptual understanding as to where countermeasures should exist and where they should be applied within the context of the threat. These countermeasures lessen the overall business impact as well as the associated risk or impact levels introduced by the threat. Such attack models are best developed at the inception of the application development process.

As shown in Figure 1.10, not all attack vectors introduce exploitable technical vulnerabilities. Some of the attacks take advantage of process-related weaknesses that are very difficult to mitigate, therefore making them more attractive to attackers. For example, a vishing attack is a technical threat introduced via an e-mail that lists a phone number for the target user to call. The exploit is deceitful messaging and the vulnerability is a trusting reader. For this scenario, there are few countermeasures that would prevent a user from having to defend against this ploy. Mail scanning technologies are not yet sophisticated enough to counteract vishing attacks, which contain no URL or images with hyperlinks. The e-mail simply includes a phone number and a misleading message, which may state that the company would never ask its members to click on links for their own security or divulge sensitive e-mail via e-mail or a website. With this disguise, recipients of vishing attacks would unknowingly call into a malicious VRU or IVR and provide sensitive data through those channels. Some of the other vulnerabilities associated with technical-based attacks involve software or platform vulnerabilities, as may be shown with any vulnerable e-mail attachment that can introduce or carry the exploit. In the chapters to follow, we will cover attacks in detail and show a sample of attack-vulnerability mappings via data flow diagramming.

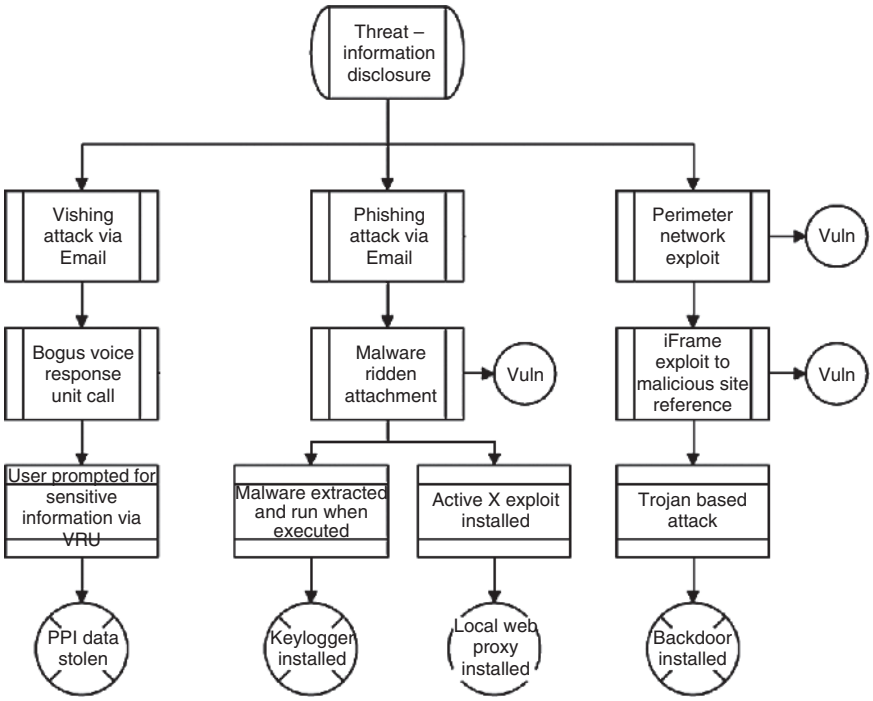

Figure 1.10 Sample Attack Tree

SUMMARY

Stepping through a threat requires a great amount of analysis and perception as a security threat modeler. Threats are driven by motives and are comprised of several dynamic pieces of content (exploits, vulnerabilities) that each require a light to heavy degree of research. These dynamic components force the threat model to be updated periodically to make sure libraries of attack exploits and vulnerabilities are up-to-date. It should be more and more apparent that application threat models can be effectively integrated into multiple IT and IS processes, such as security operations, IT change control, and SDLCs. As changes to an application environment are introduced, and as new threats or incidents are observed from centralized security logging, the threat model can evolve into an integrated security assessment model for key applications.

CROWDSOURCING RISK ANALYTICS

"*It is not a question of how well each process works, the question is how well they all work together.*"

Lloyd Dobens

"*The evil that is in the world almost always comes of ignorance, and good intentions may do as much harm as malevolence if they lack understanding.*"

Albert Camus

Collaboration does not seem to be a word that effectively describes processes that support information security efforts today. In fact, many security and nonsecurity professionals will agree that a lot of effort is wasted on security initiatives today. The security industry as a business continues to leverage fear, uncertainty, and doubt, particularly those whose intentions are profit-driven rather than altruistic goals of personal data security or even national security. Gloom and doom type marketing efforts continue to push product-based solutions, particularly in the United States where the idea of simply injecting secure process into any business operation is devastating, forcing many to gravitate to the "quick and dirty" fix. The infamous "silver bullet" continues its path in the security market, even as its benefactors argue against the premise in open forums, yet celebrate it behind closed doors. Many will argue that security solutions have in fact given way to improved security process. Although this may be true to some degree, the improvements have been primarily within security operations, compliance, and internal audit. Today, those same processes are stunted with inefficiencies and generally embellish an adversarial role toward the rest of the enterprise that inhibits collaborative work.

Isolated security groups, with their respective isolated security toys, have created multiple forms of tunnel vision – each group only seeing the value of their processes, objectives, and related technologies. Often overlooked is the ability and opportunity for integration and building a more comprehensive value-added security solution to the larger picture. Threat modeling provides the opportunity to reshape all of these inefficiencies. From a process standpoint, many groups benefit from threat modeling efforts as they receive valuable insight into risk factors associated with any application environment. Process-wise, threat modeling fosters a high degree of collaboration across the following groups:

- Developers
- QA Engineers
- Governance Leaders
- Project Managers
- Business Analysts
- System Administrators
- Security Operations
- Network Engineers
- Risk Management
- Security/IT Architects

In this chapter, we will discuss how each member benefits from the application threat modeling process and understand how the generated workflow creates a repeatable process that security professionals can leverage.

The Developer, the Architect, the SysAdmin, and the Network Engineer

Developer, architect, system administrator, and network engineer are traditional technology roles that provide integral support to application environments. The holistic picture of how the application, network, and platform all interact will ultimately be driven from the application designer or the architect. From a functional standpoint, developers bring life to the application, in all of its forms and functions. Upon having a successful software build, both the network engineer and the system administrator focus on addressing network and platform level configuration efforts to secure the application environment and the various protocols that the application will support from both a user and administrative perspective. As a result, their inclusion in application threat modeling is essential in order to contribute to the overall security posture of the application ecosystem.

Wired for developing feature-rich components, developers are focused on feature-rich applications that reflect both their creativity and the list of business requirements for the application system. Security measures that counteract any adversity aimed at infiltrating or misusing their application are absent from their development approach. Today, software development takes on a new shape and form as many of the most popular coding frameworks have prepackaged modules that address common software traits such as concatenation, mathematical formulas, and even authentication. Undoubtedly, software development today is less of a disciplined art form than prior years. Much of this is attributed to the advancement of development frameworks, which have evolved greatly to facilitate application development. As a result, a floodgate of subpar developers have flocked to developing mobile, server-side, client-side, and web apps, with little experience. The demand for software developers in the United States has been overwhelming, introducing challenges for security brought on by the shortage of qualified coders. The shortage of experienced developers has allowed looser restrictions on what is expected of software developers. This has forced many companies to look overseas for more experienced coders or domestically for average coders. As a result, the requirement for improving proper coding disciplines, particularly in security, has taken a lower priority. Given the rate at which application development needs are being sought and the rate at which software builds need to take place in order to match demands in the marketplace, a retrofitting action to build security is far-fetched. Additionally, training alone does not provide any incentive for developers to code securely since that is not what they were hired to do nor are they paid to do this. Furthermore, it is not always the developer's fault; there are system, database administrators, as well as software implementors that all share the same sentiment that security is an auxiliary component to their primary focus in building a technical solution. This perception requires a recalibration of various variables that exist in the mindset of developers that include (but are not limited to) viability of attack, impact of vulnerabilities, significance to the business.

Beyond training, security assessments have attempted to bridge the misunderstanding of some of these variables to the developers, but with very limited success. Traditional assessments against application environments take an adversarial

approach when interacting with development teams – they highlight any flaws that could *possibly* be exploited. Application threat modeling provides a process in which security professionals can address developers in a more collaborative manner to address likely attack vectors and vulnerabilities within their software applications. Developers are traditionally very responsive to these types of efforts, provided the security professional conducting the threat modeling exercise has the ability to transcend between security concepts, software development frameworks, and languages in use. To date, most experienced developers are well aware that their applications are under attack; however, they lack the understanding of how they are attacked and what type of measures they can take to limit the probability and risk that these attacks succeed.

A developer's undeclared adversary is the hacker. An experienced hacker has a solid IT background that encompasses software development, thereby allowing him/her to be intimately familiar with native methods, functions, and library objects that may be used to mitigate application threats. Unfortunately, most developers do not have such a well-rounded background and the ability to think like both a developer and a hacker, thereby creating an uneven playing field in the realm of application security. Developers are not able to think with a destructive mindset against their own application. They are focused on building up the features of their application. In this builder-like mentality, the developer does not spend time thinking of the destructive ways that their application could be compromised through various nefarious forms of attack. The purpose of threat modeling is to provide an ongoing process that allows them to understand the destructive vision of an attack against a software application by dissecting their own creation to find the weak areas or vulnerabilities. If nothing else, threat modeling allows developers to think destructively about their own application. The methodology employed by most threat models provides developers the opportunity to see their own application in the eyes of a likely attacker. It also allows them to think like an attacker while reverting to the mentality of a developer who now has a better understanding of possible attack patterns and what vulnerabilities may exist within their code structure.

Last, the threat modeling process allows the formal introduction of security requirements at the inception of the SLDC life cycle. Building security into the various stages of any SDLC process reflects a new movement in secure software development practices to design and develop security controls from the early stages of the SDLC process. The Building Security In Maturity Model (BSIMM) is a security framework that allows development groups to measure what security measures they currently have in place versus those that are recommended. The Software Assurance Maturity Model is another framework that development teams may leverage to continuously measure the security and effectiveness of their developed applications.

If developers are the artistic minds behind any given application, the system administrators serve as guardians of their creation. The security requirements that were alluded to earlier help form the necessary guidance in which system administrators should maintain the various platforms that encompass application components. There is nothing new with security requirements. Their traditional

and dependable downfall today has been attributed to the lack of process of socializing the information and requiring their use by IT management. Again, human error and inefficiency is to blame for well-intentioned security requirements not becoming implemented as a realistic practice. This is most readily observed in larger enterprises where security leaders author standards, typically from industry renowned sources such as NIST, CICS, or the platform manufacturer. From there, the socializing of these standards to IT groups, who most likely were never a part of the drafting process, begins to fail miserably as yet another adversarial approach from security attempts to dictate how IT should do their job in the name of security and compliance. The message that threat modeling fosters is one of collaboration among IT and IS professionals to mitigate risk factors. People usually want to assist or help if they have a better understanding of what their threats are as a company and as a group of system custodians (or system administrators) charged with maintaining and safeguarding IT assets. Since they do not currently have a glimpse into whom or what their adversaries might be, system administrators today are less cooperative in light of the compliance and FUD (Fear, Uncertainty, and Doubt) communication that they receive from their IS counterparts. Threat modeling's ability to depict potential attackers, their profiles and motives, likely attack surfaces, and vectors allows for a wealth of information to help system administrators understand the underlying reasons to adopt any suggestive platform guidelines or formal platform standards that need to be leveraged when creating, cloning, or configuring platform components for the application environments.

QA Engineers

Quality assurance efforts test functionality using test scripts and manual methods. QA engineers or analysts have a pivotal role in identifying bugs within their test cases. They test newly developed features and functions from the development team and are theoretically awarded the ability to accept or reject new builds depending on the outcome of their test cases. This workflow generally does not receive the recognition and power that it deserves, mostly because of the rate at which software development efforts take place and the push to migrate code to production. Most software companies accept a level of imperfection when rolling out code to production; however, the level and frequency in which flaws are introduced to a software product may affect the reputation of a software company in the long term.

Organizations where QA efforts maintain a well-established process, supported by product and project management, are ideal for incorporating application threat modeling. Given the time and effort that threat modeling imposes against a release cycle, adoption from these management groups is key to convey the value and necessity of incorporating threat modeling in the SDLC process. Client requirements and service delivery goals may influence the manner in which threat modeling is ultimately adopted. There is no question that some internal *selling* is needed to foster faith in application threat modeling and its long-term value to creating better software. This may be accomplished by identifying factors that benefit project and product managers in the end. These factors will be revealed in detail in the next section.

Threat modeling within the QA software process should not simply be added as an additional task to a QA engineer who is performing functional testing. In order to accomplish threat modeling within the same vein of QA process, a dedicated and experienced security engineer should be included. An ideal security tester will possess the following background and skill sets:

- Understanding of application design
- Understanding of multiple development frameworks
- Wide breadth of use and understanding of security testing solutions
- Solid understanding of network protocols leveraged by the application environment
- Ability to create abuse or misuse cases from all identified use cases within an application
- Ability to develop and maintain a vulnerability database and understand how inventoried vulnerabilities can be applied against various network and system level resources
- Ability to develop and maintain an attack library that addresses key threats to any identified vulnerabilities within any tested application environment
- Solid understanding of database related protocols, authentication models, and objects
- Some development experience so he/she can review available source code for possible exploits in logic or information processing
- Ability to conduct application walk-throughs to create data flow diagrams that represent the attack tree, which encompasses related vectors, vulnerabilities, and attack exploits
- Understanding of business impact and risk as it relates to viable attacks that are represented by the threat model
- Strong communication skills geared toward developers, product and project managers, and senior management. Ability to understand and relay risk-related business concerns as well as probable attack scenarios that can be depicted via threat model and data flow diagramming exercises and exploit attempts
- Experienced in risk management frameworks and their application to business environments

Knowledge and hands-on use of various security solutions is a great compliment to a solid foundation of security experience. A *brief* list of such tools is provided in Table 1.8. This list is not meant to represent the best of breed within security testing, but to simply provide an inventory of solutions that will catalyze the overall testing process.

This arsenal of tools tests for application insecurities from which a wealth of information will be obtained and subsequently used within any given threat model. The information resulting from any automated scans must undergo a validation process to extract any false positive findings that may misrepresent the security posture of

TABLE 1.8 Tools for Testing

Tool	Use
Discovery/vulnerability scanner(s)	Nessus (Tenable Security)
	SAINT (Saint Corp)
	NeXpose (Rapid 7)
	Qualys Scanner (Qualys)
	Nikto
	OpenVAS (openvas.org)
	Retina
	NMap
Web application testing	WebInspect (HP)
	Acunetix
	AppScan (IBM)
	Wikto
	Wapiti
	Burp Suite Pro
	Paros
	WebScarab
Penetration testing/fuzzers	Core Impact
	Armitage (www.commonexploits.com)
	MetaSploit
Social engineering/phishing	SpoofCard/Phone Gangster
	Social Engineering Toolkit (SET)
	LittleSis.org
	Maltego Radium
	reconNG
Static analysis	Fortify 360
	Ounce Labs (IBM)
	FxCop (MS Visual Studio plug-in)
	Parasoft
	Veracode (Binary Analysis)
	O2 Project (OWASP)
	Brakeman (Source Code Review – Ruby)
	Yasca (Open-Source Code Analyzer

the application in question within the application threat model. False positives are detrimental to the application threat modeling process since they consume time and resources chasing unsubstantiated threats. Qualifying false positives may take some time and encompasses validation against platform, network, and/or application components. Exploiting vulnerabilities within a QA environment will best qualify attacks and vulnerabilities into legitimate threats, with the ultimate objective of understanding relevant risk factors for an application environment.

Security testing, as with more traditional forms of functional or regression testing in QA, adheres to a very pragmatic approach for finding possible security flaws. As

a result, it is not the most opportune juncture to require risk analysis from a professional who is focused on exercising a suite of security test scripts. This is where the necessity to have a dedicated security professional (embedded within the QA process) is warranted since most QA professionals will have limited to no exposure to applying risk-based approach to their functional testing. This risk-based approach to security testing will foster interoperability of results among the QA and Enterprise Risk Management groups. Security testing today is nothing new for mature organizations that incorporate multiple security processes within the operations group; however, applying a risk-based approach to vulnerable findings is scarcely applied. More information on how application threat modeling leverages risk management workflows is forthcoming in the next few sections as in other portions of this book.

Security Operations

There may not be consistent security processes universally represented by a security operations group or center; however, they typically oversee the following efforts:

- Vulnerability management and penetration testing
- Incident response and security event monitoring
- Security log review and auditing
- Threat aggregation and analysis

Security operations often perform the aforementioned list of functions that fuel excellent intelligence to security professionals who are building the various components of the threat model. Specifically, information from this group provides greater accuracy in deriving probability coefficients for identified attack vectors. Alerts from managed network and application intrusion solutions provide an excellent level of information in understanding the following:

- Trend analysis of attacks over a given period of time
- Origin of malicious traffic (IP space, networks, geographic regions, etc.)
- Frequency and intervals of malicious traffic patterns
- Correlation of observed traffic patterns
- Threat feeds from subscribed threat or alert feeds
- Breakout of malicious traffic across certain criteria:
  - Internal versus external
  - Resemblance of targeted versus broad range of attack
  - Distribution of network protocols for observed attacks

Inclusion of security operation groups in the threat modeling process builds upon efforts during the QA or security testing phase of a given software application. Security testing can take place at a time interval that best suits that sponsoring organization; however, it is best incorporated into QA simply because its testing process is very

much akin to the functional security testing conducted during traditional functional test cases within the SDLC.

The efforts from security testing should be comingled with the aforementioned information from security operations to refine estimates on probability coefficients that accompany various attack variables (discussed further later). This information can be correlated to attacks identified from the threat modeling attack library to legitimize further the attack scenario against the assessed application environment. Observed network patterns that resemble variants of exploit traffic are invaluable to the threat model as it helps to refine risk scenarios that are derived from the application threat model.

Observed malicious traffic tells one-half of the story, as it relates to possible threats to a company's application environment. The threats that have yet to be observed are equally important within the application environment. This information can be obtained from threat feeds, typically sent to security operation analysts for tracking and is especially useful if obtained for the company's industry sector. Threat information related to DoS attempts and exploits may be prevalent to companies in the energy sector, while injection-based threats may be more highly reported for those in the online retail business. Threat feeds provide the same level of benefit (to a slightly lesser degree) as the security incidents observed from a security operations center.

Correlation – The Final Frontier?

It goes without saying that information correlated and/or aggregated from security operations will have to have some degree of topicality to the assessed application environment within the threat model. For example, an HR SAS solution that is assessed within the threat model would not benefit from a broader scope of network or application areas to accounting if there is no application programming interface (API) among the two disparate systems. If such is the case, logical networks, assets, and applications that tie the two disparate application environments should be inclusive of the application threat model, but not anything further. This is done to ensure the proper scope of the threat modeling exercise. A larger scope may undermine the time and efficiency of the threat modeling process. The following is a graphical representation of properly defining scope among two unique application environments that are bounded by an API.

Enrique Salas, CEO from Symantec stated in his 2009 keynote RSA speech that one of the differentiating factors of managing security risk is how massive amounts of security information stemming from intrusion detection/prevention systems (IDS/IPS), firewalls, host intrusion prevention software (HIPS) agents, antivirus clients, host-based firewalls, network content filters, data loss prevention technology, web application firewalls, vulnerability scanners, spam filters, threat feed subscriptions, web application scanners, and more are all correlated to maximize the security risk insight across an enterprise. Companies that employ a part of the aforementioned network and host-based technologies can have a plethora of threat intelligence, regardless of whether the information is administered and managed internally, via a cloud-based service provider, or as a managed security service

(MSS) – the information exists and should be to help fuel threat scenarios simulated by the application threat model. Many of these solutions provide a historical view of threats to a given environment that is monitored by these and other security technologies.

(Security) Risk Management

Weeding out false positives within an application runs parallel to the need to understand risk and impact levels from qualified threats. Leveraging the security testing that should take place, preferably within the QA process; a level of unmitigated security risk issues will undoubtedly be present and can easily be manifested to security risk management groups within the process of application threat modeling. Unmitigated risks are those related to clearly marked attack vectors that present viable threats against existing vulnerabilities, which have negligible countermeasures to limit either the introduction of the attack into the environment or the exploitability of the vulnerability. Understanding risk entails a comprehensive understanding of multiple factors, all of which become better understood through a formal risk management process. Since most threat models provide a greater level of application risk by illustrating mappings among attack exploits and vulnerabilities, coupled with business impact values and probability values based upon informed research on attack complexities, ease of access, sophistication level, and so on, variables are largely missed by more traditional risk management efforts in enterprise security.

This section focuses on how introducing a basic liaise among security risk management and application threat modeling leverages the common objective of identifying and managing risk. Application threat models substantiate risk models: they provide greater credibility by simulating threat scenarios and thereafter establishing a full etymology of attack branches, related vulnerabilities, and associated countermeasures where residual risk can be addressed through risk management practices.

Within the realm of traditional risk management efforts, the following security elements are the bare essentials for any generic risk management framework (Table 1.9).

Regardless of the employed risk framework or risk model within an enterprise, threat modeling provides greater precision in some of the aforementioned risk components. Some globally renowned risk frameworks and standards include OCTAVE (Carnegie Mellon), NIST Risk Management Framework (800-53, 800-60, 800-37), the revised AS/NZS 4360 standard, which is now the ISO 31000:2009 Risk Management Standard, COSO ERM, and the new RiskIT integrated risk management framework from ISACA which encompasses many key elements from these more widely recognized frameworks and standards. Although well-known throughout the globe, many of these frameworks lack the technical specificity to provide actionable implementation of effective countermeasures or controls during a remediation phase of the risk management process. Additionally, many of these risk management frameworks or standards do not foster the ability to extract precise technical information to further diagnose application-level risks. Those who argue that this granular level of risk does not convey business risk do not apply a threat modeling perspective, which

TABLE 1.9 Elements of Risk – Generic Listing of Key Risk Components

Security Risk Components
1. Scope of affected Hardware and Software assets
2. Business impact analysis (consequence) related to scope of assets
3. Identified and confirmed vulnerabilities
4. Enumeration of possible attack patterns and supporting rationale
5. Threat model denoting probability or likelihood of exploitation
6. List of physical and logical countermeasures
7. Identification of residual risk
8. Implementing countermeasures and controls
9. Informing and Training
10. Monitoring

begins the process by identifying the scope of business or information assets encompassed by a threat model and later defines what elements of the asset, if not its entirety, are affected by the depicted attack branches. The scope definition also encompasses the business objectives that are supported by the assets or targets in the threat model, thereby allowing business risk analysis to be derived via the threat modeling process.

Beyond some of the more globally recognized risk management frameworks or standards are comparable risk frameworks/standards that have been developed by private and/or public organizations, including Microsoft, Google, Verizon Business, OWASP (Open Web Application Security Project) and more. Although these publications are not as widely adopted and practiced, they are based on the fundamentals of some of the previously mentioned industry standards for risk management, with emphasis on certain types of technology environments. They also incorporate a greater level of technical detail, which incorporates more meaningful content for articulating risk-remediation activities to system/data custodians across a given enterprise. We will take a closer look at existing models, frameworks, and risk management guidelines in further sections of this book, to further correlate existing risk models to the risk analysis capabilities provided by an application threat model.

An application threat model conveys application risk values that can be incorporated into a greater risk model managed by enterprise risk management professionals. The by-product of threat simulations, achieved by application threat modeling, allows a more sophisticated value of application risk. This sophistication is attributed to the application walk-through and attack simulation that gives way to well-defined attack scenarios, which are likely and associated with validated vulnerabilities. Once a well-defined attack tree contains a full set of layered branches (reflecting assets, associated vulnerabilities, attack exploits, and attack vectors), many of these branches then need to be assigned probability and impact values.

Probability and impact variables will ultimately help derive risk levels for the assessed application. The compounded net effect of vulnerabilities to attacks (or threats), along with associated impact or consequence values, probability estimates for successful exploitation, and net of existing countermeasures provides a far more accurate representation of risk compared to more general security risk equations that equate risk to simply a product of vulnerability and threat. Some traditional risk models do incorporate impact (or consequence) as well as probability, but none can truly represent probability variables in the risk equation since there are so many assumptions built on these probability levels. These assumptions have to be made under more generic risk models since the attack is not simulated. Under a threat model, the attacks are simulated in a controlled environment and a greater degree of accuracy can be made as to ease of exploitation and access to attack vectors as compared to a purely theoretic risk analysis exercise.

With an improved risk analysis, obtained by the application threat model, remediation takes on a greater level of importance since the overall risk analysis clearly shows a linear representation of cause and effect of not having existing countermeasures for a given set of assets or subject targets in the application threat model. For nonmanaged risk (meaning nonaccepted or nontransferable risk findings), countermeasures can be developed with greater direction. Ultimately, the dominant objective of any risk model or framework is enabled by the application threat to the application threat model – deriving risk to identify what countermeasures need to be developed, if any at all. This is the light at the end of the tunnel for risk management professionals since it focuses on completing the life cycle of risk management for discovered risk issues. Remediation efforts via countermeasures in process or control fulfill risk mitigation efforts, greatly aiding enterprise risk management professionals in fulfilling their group goals and objectives.

Elements of risk bolstered by an application threat model are depicted in Figure 1.11. Most traditional risk assessment efforts within a risk management practice are inherently qualitative, making it difficult to get complete adoption by some of the target audience of its deliverables. Via an application threat model, the following formula can be applied to substantiate risk designations, via its inclusion of impact values and greater precision in probability estimates, both influenced by the actual threat modeling exercise, whether they adhere to a quantitative translation of qualitative risk or simply a traditional heat map of risk levels.

Elements of quantitative risk analysis are concentrated around probability and business impact values, which encompass projected values for financial loss. Probability values in threat modeling encompass any statistical reference that supports ease of exploitation as well as successful exploitation attempts realized during the application threat modeling security testing process. The ability to exploit an identified vulnerability within the testing phase of the threat modeling process greatly substantiates probability estimates as compared to more theoretical values encompassed in traditional risk assessment methodologies.

A traditional risk formula generally encompasses the following variables for *risk*:

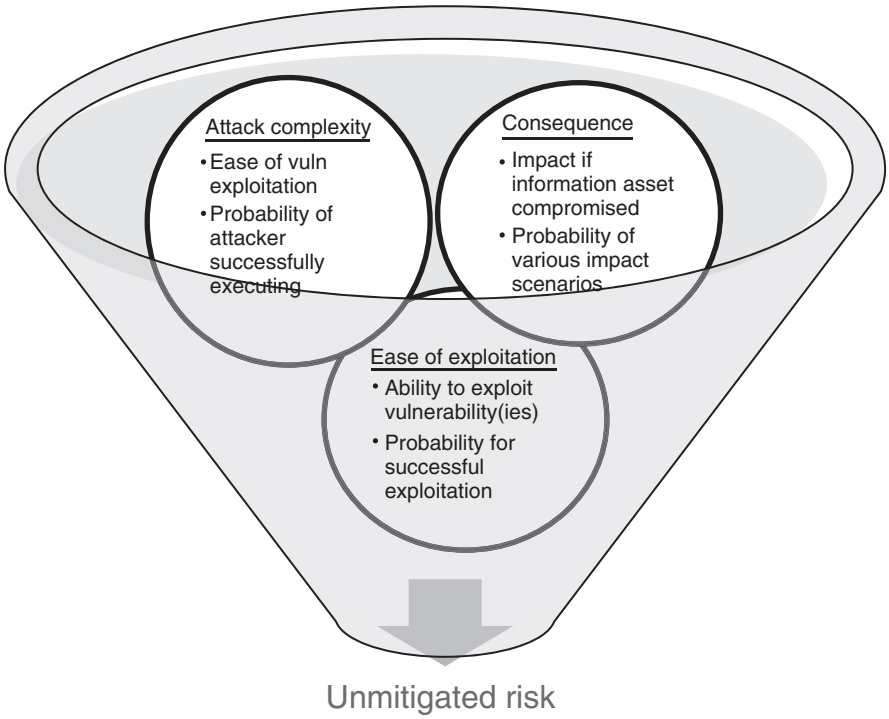

Figure 1.11 Deriving Risk via the Application Threat Model

- *Impact (*or *consequence)*
- *Threat (*or *attack)*
- *Vulnerability*

There are a multitude of risk models used today, both quantitative and qualitative, that incorporate the aforementioned risk variables. Undoubtedly, it would take the remainder of this book to argue each risk model's worth. It would take even longer to demonstrate how an indisputable or universal risk model may exist that properly addresses information security risks across all industry sectors and infrastructure types. Among the various risk models and methodologies, only one universal truth should exist relative to application risk: risk is relative. This is the reason that application threat modeling is essential for feeding an overall risk model to improve its risk analysis capability.

Application threat modeling can feed and bolster the risks maintained and calculated in more traditional risk models due to its ability to supersede traditional risk methodologies in four very important areas. They are as follows:

- Identifying uniquely identifiable threat scenarios
- Incorporating business objectives

- Improving on probability calculations
- Performing attack exploits to simulate real life risk scenarios

Application threat modeling, as a process, allows unique risk factors to be evaluated. It focuses on identifying technical application risks that are programmatic, platform, and network related, while aggregating this information to its relevant impact to the business that the application supports. Application threat modeling's objective centers on the uniqueness of various risk factors: unique threats, unique attack vectors, unique assets, and unique information sources, targeted by nonunique vulnerabilities and nonunique attack exploits. The context of *unique* reflects the fact that distinct application technologies are not, in aggregate with one another, found across other application environments owned by other corporate entities within the same industry segment or business type. A retail site that sells automotive parts will indeed have vast similarities with other retail sites that offer comparable products and services; however, the application architecture, associated platforms and software technologies, development frameworks, and application designs will be largely unique. Most importantly, the application use cases, gateways over which an attacker can interrogate an application (viewed by attackers as attack vectors) will be unique among distinct sites and business entities. Application threat modeling provides a process for understanding these unique variables through the use of the attack tree, where attack simulations encompass all relevant risk variables, including vulnerabilities, attacks (exploits), impact levels, and application countermeasures.

Regardless of whether a risk model is qualitative or quantitative, risk ultimately embellishes unique threats, vulnerabilities, and business impact scenarios that are often organization specific. Two competing banks may offer identical online banking sites, but the initial and ongoing efforts behind those B2C sites will encompass unique development teams, software and platform technologies, and architectural design for interoperability within and outside the overall application environment. IT and IS governance within the two disparate companies may also differ, thereby potentially affecting the security posture of an application, mostly as a result of having clearly defined application, network, and platform configuration standards. More traditional risk models exclude business-related objectives and features when identifying risk scenarios for an application environment. Application threat modeling, conversely, begins with the inclusion of defined business objectives. Risk analysis begins by identifying any underlying use case, feature, or functionality that does not support business objectives for the application's continued support and use. The distinguishing characteristic of risk within an application threat model is that the model for understanding risk is centered on the nature of unique information and technology assets (or targets) for an organization as well as its countermeasures and process-driven controls. The end result is a more precise risk model for deriving information security risks for software.

Probability values in calculating risk are another improvement via the application threat modeling process. As stated earlier, *probability* is a value that is often incorporated into more traditional risk formulas, albeit with less precision than application threat models. Via a threat model's attack tree, and the opportunity to simulate attacks

in a white hat (or ethical hacking) scenario, greater accuracy in estimating probability is sustained. Within the threat model, attack tree branches allow visualization of a threat over a series of sequential attacks, thereby allowing probability values to be assigned to those attack branches. The probability value is still an estimate; however, its integrity is improved upon the opportunity to exercise an identified attack in a controlled environment. Not all attacks can be realized in a practice scenario within the threat modeling process. This does not reduce the value that threat modeling brings in improving probability values for an overall risk calculation. The attack trees within the model still provide a visual flow in which known attack exploits can be exercised over discovered application vulnerabilities. Each branch or layer of the attack tree will allow unique probability assignments for those attacks to be realized against discovered vulnerabilities and their exploits. Ideally, the traditional *probability* variable should be used as multiple coefficient values that reflect the likelihood for the following to take place:

- Likelihood that the vulnerability or set of vulnerabilities become successfully exploited.
- Possibility that the attack vector becomes accessible for exploitation and the attacker has necessary time and resources to conduct the exploit.
- Likelihood of various impact scenarios to become fully realized.

This coefficient use of probability $_{(p1),\,(p2)}$ is best illustrated by the following altered risk formula.

$$\text{Residual Risk} = \frac{\text{Vuln}_{(p1)} \times \text{Attack}_{(p2)} \times \text{Impact}}{\text{Countermeasures}}$$

Overall, enterprise risk management programs will greatly benefit from an application threat modeling process. More details related to application threat modeling's relevance to widely used risk models will be elaborated in greater detail in subsequent chapters.