# 1

# Penetration Testing Web Applications



At the end of the day, it all comes down to code. There are few information security issues out there that cannot be traced back to bad code, lazy coding, ignorant programming, something having to do with bad software, or bad practices in the creation of software. The fact that it all comes down to code is one of the deeper points to pick up about application and software security because programmers hold those keys. Whenever a security engineer sets up a firewall, she is running someone else's code. Whenever a network engineer configures a router, she runs code someone else wrote. Whenever a network security professional runs a port scanner or an automated vulnerability assessment tool, she runs code someone else wrote. Flaws can exist in those sets of code that can go undetected until someone knowledgeable in that area takes the time to audit them (or they stumble upon the discovery). Finding flaws in applications and software is especially challenging with compiled code that is closed in nature. On the other side, even if code is open sourced it sometimes takes deep programming knowledge and experience to read some source code, understand it, extend it, modify it, and secure it.

## Security Industry Weaknesses

Since the proliferation of the Internet, Web applications (referred to simply as *apps* in this book) — from simple personal and corporate informational web sites to complex applications rich in functionality — have been popping up at an astonishing rate. In the business world, it seems as if any entity without a solid Internet presence is looked on as amateurish. However, although these apps might all share a common presence on the Internet, they are all created differently, on different stacks of technology, and exposed to the world in radically different ways.

Vulnerabilities arise from many sources, from the pressures caused by unrealistic development deadlines and limited consideration for security in the application specification typically delivered to developers to the unsuitability of network-level security to provide adequate protection. The following sections introduce each of these problems, which are explored further throughout the book.

## Application Development Considerations

These apps are typically written with a focus on functionality, but minimal security. This, unfortunately, is the nature of programming and development, although programmers and developers are amazing at creating great functionality against some finite set of requirements. And typically, as long as those setting forth the requirements for the apps are happy, the programmers and developers have done their job. Security will either be added in later or dealt with on a network level. Historically, this approach to application development is pervasive and has created tons of software full of security holes.

The bottom line is that it is difficult to code as quickly as our jobs require while maintaining security as an area of focus. Creating secure code may mean that the crazed deadlines we are all given cannot be met, and that is a problem — none of us like to miss deadlines. Now I am not stating that developers and programmers are irresponsible — I am sure that if we were all given realistic time frames to code we would all do the right thing in reference to security. But business rather than IT or engineering teams typically drives software deadlines. Ultimately we all pay the price for the rushing and cutting corners.

## Limitations of Edge Security Models

Applications are typically deployed behind some thick edge, or network level, security infrastructure (firewalls, Intrusion Detection System (IDS), Intrusion Prevention System (IPS), and so on). Traditionally, they are then left alone, facing the world 24/7 via the public Internet, to do what they have been programmed to do. The problem with this approach is that there always seems to be some level of mystery about the inner workings of these apps for the network, security, and general IT teams that maintain them. These folks most likely have not been forced to get involved at a level of depth that would expose the core of apps and so their mystification is understandable. It also means that they are at a disadvantage when it is time to look under the hood of apps and tell if something is problematic.

The majority of commercially and personally deployed applications write transactional data to some logging mechanism. They traditionally have had very little other inherent security built into them. To make matters worse, it sometimes isn't even the app that is writing data to a logging mechanism, but rather the web server is doing that. The web server is usually a shared resource among numerous apps and sites, which makes these logs that much more difficult to contend with. Reality in the IT industry has proven that most software and security engineering/operations teams do not have the bandwidth to actually review anything that is readily and regularly logged. So the problem isn't only the breach; it is also that internal resources don't realize their apps have been breached. These situations tend to get investigated in a reactive manner.

> *Breaches are taking place constantly — you can get an idea of the extent by visiting SecurityTracker (*`http://www.securitytracker.com/archives/category/4.html`*) and Zone-H (*`http://www.zone-h.org/en/defacements`*).*

This typical scenario brings to light some of the deficiencies of a security model based strictly on the edge. Unfortunately edge- or network-level security does very little in terms of actually protecting applications. Yet most modern-day IT teams and operations are blissfully comfortable with edge security supposedly providing protection to their data and functional resources. The reality of the matter is that most edge-level security mechanisms identify some web-based attacks as legitimate traffic and they happily watch the traffic flow on by.

# The Case for Pen Testing

The false sense of security that exists within the IT industry has spawned the much-needed profession of Web application security and penetration (pen) testing. The focus of this book is web-based application penetration testing and how the results of this can lead to a layered approach of enhanced core-level security coupled with specialized edge security for Web applications. The benefits of being a programmer doing pen testing are due to the deeper levels of app understanding gained through this practice. But other IT professionals will get eye-opening information and education as well.

## *Industry Preparedness*

Application pen testing is a critical discipline within a sound overall IT strategy. But there is a serious problem: the shortage of those with the necessary skill set to properly pen test software, N-tier, and distributed applications. Experience has shown that the typical security specialist or engineer simply lacks the depth of application and software knowledge necessary to be entirely effective when performing these types of audits. Knowledge of code is absolutely necessary to do this effectively.

The present state of affairs with Web applications, and corporate software on the whole, is one of quasi-mystery. There is a disturbing gap in the industry between the programming community (which focuses on solid functionality) and the security community (which focuses on protection at the network level and policy). The gap exists because, while programmers and web developers are traditionally focused on the functionality of apps running critical processes, and network and security professionals are traditionally edge and possibly even host specialists, no one is looking for security holes in the application code. The mystery, then, is who properly secures the Web apps? The experience of programmers has been that security is not a priority in their application development workloads.

Many businesses do not have in-house application skill sets or resources. Although they have apps that are, and have been, running their business in the background — and from a business perspective everything functions as expected — these apps have typically been outsourced or off-shored for development. This means that there is very little application knowledge on staff. Because of that, one of the things regularly encountered out in the field is very old and unpatched versions of software. This is a huge problem because the most up-to-date software out there typically has enhancements and fixes built into it. But many entities will not always keep up with the latest and greatest software due to a lack of in-house knowledge and experience.

A further consequence of having applications that no one understands is that the folks in-house that are held responsible for these applications refuse to touch them for fear of breaking them. Things that are seemingly simple, like applying server patches, could conceivably wreak havoc on an application. If they do apply the patch, library dependencies can end up broken and the app could start spitting out nasty errors. Anyone who witnesses a fiasco like this makes a clear mental note not to repeat that mistake, which leads to an unwritten no-touch policy.

Sometimes it is not even a human process that causes a problem. If a server runs long enough without being turned off, there is no guarantee it will come back up after a shutdown. The average professional in the IT industry will not want to be the individual that powered that server down. They don't want to deal with the repercussions if the shutdown causes an app to stop working. So there is a distinct preference not to touch anything that is perceived as not broken irrespective of the associated risk.

Many edge-level techniques and tactics leave great areas of risk exposed. For example, the functionality of IDS systems is impressive, but what they watch is dictated by what they are taught. However, the critical question remains, who is doing the teaching? Do they properly understand what they are looking for? Assuming they do, someone or something then has to make sense of the massive amounts of data these systems typically capture. It is an intensive, time-consuming process that in the real world has proven to be a "nice to have," yet the true real-world security value is arguable. IPS systems come with their own set of challenges and weaknesses. The point is that when it comes to Web apps, there are weaknesses, and risk is generally present.

> *IDS (*`http://en.wikipedia.org/wiki/Intrusion_Detection`*) and IPS (*`http://en.wikipedia.org/wiki/Intrusion-prevention_system`*) systems are network-level devices that aim to enhance an overall security posture.*

Awareness in this area is growing, though. Evidence of this can be seen in movements like these:

- Open Web Application Security Project (OWASP) - OWASP (`http://www.owasp.org`) is dedicated to helping build secure Web applications.
- Web Application Security Consortium (WASC) - WASC (`http://www.webappsec.org`) is an international community of experts focused on best-practices and standards within the Web application security space.

One emerging, very interesting area is that of Web Application Firewalls (WAF). These are devices or software entities that focus entirely on the proper protection of Web applications. These WAF solutions are intended to fill the gap I spoke of earlier. They are capable of properly preventing attacks that edge/network-level firewalls and IDS/IPS systems can't. Yet they operate on the edge and the app's source code may not get touched. You can get information at `http://www.modsecurity.org` and `http://www.cgisecurity.com/questions/webappfirewall.shtml`.

As we all know, not all products are created equal, and this is especially so in software. Ivan Ristic and the WASC are going to the great length of formalizing the evaluation criteria of these types of solutions so that anyone can at least have a baseline understanding about the effectiveness of a given solution. As stated on their web site: "The goal of this project is to develop a detailed Web application firewall evaluation criteria; a testing methodology that can be used by any reasonably skilled technician to independently assess the quality of a WAF solution." They have a strong Web app security presence within the consortium, and the criteria seem solid. You can learn more at `http://www.webappsec.org/projects/waf_evaluation/`.

## *Finding the Right Mix of Experience and Methodology*

Because the security industry is predominately staffed with network professionals who have migrated to security via firewall, IDS, IPS, work, and so on, it is easy to find a great many misconceptions in the way application pen testing is described and practiced. For starters, there is sometimes an unfortunate misconception that pen testing doesn't follow any disciplined methodology. The thinking is that there is a strong dependence on the tester's experience, and therefore there is a direct correlation between that experience and the difficulty of the specific target. However, even with a good store of relevant experience, without a defined methodology it is easy to make mistakes, generate inaccurate results, waste time, waste money, and finally lose the client's (when servicing external clients and not auditing your own shop) confidence that they will receive an excellent end-result product.

Now, some entities performing this type of work operate with no methodology and this is certainly not a good practice. It is almost as bad as using a methodology that is far too general, or pen testing based on the instinct or knowledge of specific individuals. A common case of this sort of flawed approach to pen testing can be seen in methodologies that preach information gathering, penetration, and documentation. Unfortunately this methodology is pervasive. Far too many companies and IT personnel have the erroneous idea that a penetration test constitutes nothing more than running a security scanner and getting a nicely formatted report with colorful charts at the end of the run. The main fault with this model is that the results depend only on how many problems were discovered, which depends on what the scanner has been taught by its programmers, and that depends on the experience and knowledge base of those particular programmers and possibly some analysts associated with the project. To view one of those reports as complete and comprehensive is a grave oversight to anyone familiar with the way security scanners and other automated tools work and who truly know about the false-positives and false-negatives they produce.

There are countless examples of sloppy and inaccurate pen tests done by big and small consulting companies that lack the right skill set, depend on automated tools exclusively, or both. The reports they provide are based on the superficial results of automated tools without any deeper analysis. This is downright irresponsible and potentially leaves a trusting client needlessly exposed because the tool(s) used, and the people using them, may have missed something.

### *The Role of Social Engineering*

There is great value in coupling Web app pen tests with other social engineering efforts, such as shoulder surfing. Even though this book exclusively focuses on the technical aspects of Web applications, the value of a successful social engineering campaign cannot be negated. Many times the app security experts are fed information from the social engineering efforts of others on a Tiger Team (`http://en.wikipedia.org/wiki/Tiger_team`). It is a flow of information that could certainly speed up the whole process; just don't rely on it because without it you still have to get results. It is also your responsibility to feed back to the rest of the team any data you discover that may be relevant.

## *The Bottom Line*

There is much more to application pen testing than blindly running a few tools and producing a report. It is imperative that organizations make themselves aware of their risk level in the arena of web technologies. Acting on the awareness is not only critical but now is becoming a legally-based demand. Web-based vulnerabilities and the potential attacks and exploits are growing at alarming rates and they require attention today. The business-related consequences for any organization doing business on the public Internet who fails to take application and data security seriously could be devastating, especially considering the repercussions of non-compliance within areas such as Sarbanes-Oxley.

Organizations need to implement awareness programs through effective pen testing, and they need to implement solutions based on the results of that testing. To protect against potential attackers and breaches, a proactive, layered defense strategy is a must. Truly thorough defensive postures can always beat out the offense in these scenarios because there will just be an easier target elsewhere. This works out cleanly when security is implemented as a legitimate area of attention within a given project's Software Development Life Cycle (SDLC). More often than not, security is not implemented during the SDLC and so an objective assessment is in order via external (to the target entity) penetration testing. This book provides you with the necessary knowledge and tools to execute this objective assessment.

# The Mindset

It is critical for anyone getting into Web application pen testing to understand the necessary mindset. It is not as simple as getting into a hacker's state of mind and just blatantly attacking your target. True Web application pen testing requires a very diligent, methodical, and scientific approach. Moreover, diligent documentation is critical at every successful step along the way. Success here is established not only by the discovery of a vulnerability or possible hack, but by discovering it, documenting it, and ultimately educating those who are responsible for the respective target in order to mitigate the related risk. So yes, you will be emulating the techniques of a malicious hacker but you will do so with good intentions and solid documentation practices. The intentions are that of giving your client the best possible feedback from your testing so that her site becomes less of a target to those out there with malicious intent.

As a pen tester you are not really a bug hunter in the Quality Assurance (QA) sense. What you are is:

- **A Hunter:** Your objective is to track down an elusive adversary that may lie deeply hidden in some obscure section of code, and it may live within some heavily used application.
- **An Educator:** You must expand the knowledge base of those who are not intimate with software and application.
- **A Warrior:** As a warrior it is your job to gauge the preparedness of your targets. You must make sure they do not sit exposed as an easy target in the battleground of information warfare.

Pen testing Web apps will be a great test of your tenacity, perseverance, patience, and creativity. However, although preparation is priceless there will be times when you exhaust your entire arsenal and find nothing. But understand that there are tons of tools out there to facilitate your work, so if you do exhaust your arsenal, my advice is to simply write a solution yourself. Coding some programs to facilitate your pen testing could prove very worthwhile, and you will see many examples throughout this book. Also understand that you don't have to do everything from scratch. You will be using others' work, especially in the arena of known exploits. People are out there doing amazing work in research and sharing it with the rest of us. Learn how to use it.

One final note about mindset — your mindset has to adjust a bit to be a double-edged sword. As a software professional, for instance, new functionality, technology, and coding methods should excite you. As a Web app pen tester (a software professional with a security twist) these things should excite you, yet at the same time you have to start thinking about how all this can be broken, breached, abused, and so on. This is how you will stay on the edge mentally.

## *Creativity*

If you think that pen testing Web apps is a matter of running some tools with magic "Find" buttons and getting solid successful results, you are pursuing the wrong field. If you think someone out there is going to give you a formula and related toolset so that you can be mindless and follow "X" steps to get the results that are going to benefit your targets, again, move on! Pen testing Web apps requires great thought, creativity, and perseverance.

How do you think known exploits are discovered? Did someone following all of the rules of the HTTP protocol, for example, discover them? Buffer overflows are another good example of an area of exploit that has probably been discovered creatively. Some programmer probably looked at the way data was handled by some set of source code and most likely was curious about how the program would react if

more data was pumped in than it was written to handle. You will need this "pushing past the edge" attitude to be a good Web app pen tester. You must think creatively and seriously about breaking things — after all, breaking things is what you are doing. You are breaking Web apps (that you have been given permission to break), and then educating those who are responsible for the target's security about how you did it and how they can avoid its being done again. One caveat here is that for you to be intelligently creative in this realm you have to understand how these Web apps operate.

## *Digging Deep*

I will not give you a motivational speech about digging deep within yourself. That is not my role. What I will tell you is that if you are the type of person who tries a technique once, encounters failure, and that failure makes you throw your hands up and say it just doesn't work, then move on. This field requires people who see a failure as an indication that a different approach is needed, or a different technique altogether. The golden rule is that there is always a way; you just have to be really bull-headed in your pursuit of finding it.

Another area where digging deep will prove to be critical is in your hunger for knowledge and understanding of the mechanics of Web apps, and technology in general. If you are the type who accepts things working in some blackbox fashion, then you'll never reach the great depths of this interesting and fast moving arena. You must never be satisfied with the fact that some technique or tool worked — it is how, and why, it worked that is key. Now I am not saying you will have the time to dig deep into everything app related. That is not realistic. But there are certain areas that you will encounter repeatedly, and it is those areas where you will want a non-superficial understanding of what is taking place under the hood.

Another aspect to this notion of depth is to become as intimate as possible with the mental state of your enemy. One of the benefits of curiously pursuing in-depth knowledge is that it will expose you to the process and mindset of hackers.

Curiosity is what many times breeds a new hacker. Many things motivate one with an inclination toward hacking or cyber crime. Notoriety, boredom, revenge, and loving a challenge are just a few of the motivations for a future hacker. Hackers can begin their training innocently and are often curiously hacking to see how far they can get. In some cases they may not even realize the repercussions of their actions because they are just following their curious instinct. But as time goes on, and their skills increase, they begin to realize the power of what they have gained in knowledge. There is a misconception that hacking is done predominately for personal gain, but that is probably one of the least reasons. It is usually the hunt.

Another reason is that hacking applications is an intense intellectual challenge, kind of like a tough puzzle. Discovering deep and unknown vulnerabilities, finding a hole nobody else has, and so on; these are exercises for an advanced technical mind. The effect of this can be seen by the public competitions and challenges that have spawned off on the Internet. Programmers are generally eager to accept an intellectual challenge — just go to Google and search for "hacker challenge."

The point to take here is never to be satisfied and hunger for the truth; blackboxes only stay black to those who let them.

# The Goal

At a high level, these are the goals of a Web application pen tester:

- ❑ Discover, document, and expose your target's areas of risk.
- ❑ Educate those who are responsible for the target.
- ❑ Remediate the situation, or assist in the remediation process.
- ❑ Assist in ensuring that target Web applications and related data are in compliance with relevant guidelines and laws.

The goal of any successful pen test is to expose (which depends on discovery and documentation) and potentially plug (or assist with plugging) the holes found in the client's target applications. Part of the exposition is educational — it is the responsibility of a good pen tester to educate those responsible for a target. You must raise the necessary awareness for your clients to either contract you (if you have the right skill set), or find someone else who is qualified, to mitigate whatever risks have been exposed.

The ultimate pen test goal is that any attacker will face layer after layer of solid security. The hope is that the attacker will simply move on to a less hardened target that has not benefited from your scrutiny.

Of late there is a new goal: securing the data stored and distributed by applications and databases within compliance law guidelines. This goal is important because in the event that data is stolen, the individuals responsible for that data can be held liable for the loss and unnecessary exposure, thanks to new laws. As a result of these laws, organizations have great incentive to clearly understand and resolve any weaknesses in their data protection capabilities. I have yet to meet anyone who would rather risk jail than implement sound data security policy.

# Methodology

While I don't preach, or subscribe to, any particular named methodology, I do expose you to a solid slate of useful information, techniques, and tools. The goal is to empower you enough so that you can perform this service for yourself or at least act as an educated consumer when engaging outsiders in these types of services. Moreover, the knowledge and techniques exposed in this book are applicable to any sensible documented methodology; they are based on real-world experience with a myriad of targets. You will actually find tremendous similarities if you analyze the methodologies that are out there and documented. They generally all follow these steps:

1. Discovery: A phase where information on the target is gathered
2. Attack planning: A phase or phases based on the results from the prior phase or phases
3. Attack: The attacks planned in the previous phase or phases are launched
4. Remediation: Plugging the holes that were found in the attack phase

A great example of a formalized and documented pen testing methodology is the Open Source Security Testing Methodology Manual (OSSTMM) (`http://www.osstmm.org`).

The informal methodology presented in this book is not formalized in any way, unlike that of the OSSTMM. The material I present is based on real-world experience gained doing this type of work in many capacities, including some of the highest sensitivity. The techniques can certainly be used with any formal methodology you prefer; I just stay away from theory and practices that could ultimately be wasteful. Though this is not a named methodology, the phases presented in this book are as follows:

- **Discovery** — Dig up and gather logistical, network, and administrative data about your targets
- **Analysis of Discovery** — Analyze the discovered data so as to understand your targets
- **Launch Attack Simulation on Target** — Pen test to probe for areas of weakness
- **Analysis of Attack Results** — Analyze the results from the probes
- **Documentation of Results** — Document the analysis from the probes
- **Presentation of Results** — Present your findings and recommendations to the project stakeholders
- **Remediation (optional)** — Handle whatever aspects of remediating your findings you have been tasked with

This loose methodology gives you the freedom to adapt to just about any environment. When you approach this type of work with a rigid methodology you will find that some environments just don't respond well, so you and your methodology must adapt to the target environment.

## Rolling Documentation

Please don't fool yourself into thinking that you will remember all of the steps that led to a given result — or the result itself — accurately. Rolling documentation means that you will be making notes along the entire life cycle of the pen test. If you are good at this type of work, you will be doing multiple projects at the same time, and this makes it difficult to accurately remember all details later on in the process. Get in the habit of making methodical notes upon every discovery per target. Make a file jacket, digital document, chapter in a notebook, something organized and dedicated per target. It doesn't matter how you do it as long as it is organized and readily accessible to you during your work. For the context of this book, I will make mention of these documentation points so that you see where you will be taking notes during your pen testing endeavors. You can simply emulate a freeform notebook-style concept. You will start making notes in Chapter 3, "Discovery."

## This Book

In this book I stay away from theory and practices that may ultimately result in a waste of resources. For example, I explain the process and usefulness of formally modeling threats. There is value in that, but you may find that in real-world projects more often than not the time necessary to properly model and document threats is not given to you. Some clients find this documentation useless. I have found through numerous sensitive projects that clients want the results, not what is perceived as theoretical data. So exposure to this is included but not focused on.

This book is also more than a hacking technique book in that it aims to address penetration testing of Web applications as a professional endeavor. There is a big difference between finding one or two holes in an application and doing an all-out professional penetration test. From that perspective it does expose

you to many different hacking and attack techniques, but it does so in order to add value to your respective target, be it a business, government, organization, and so on. The target could very well be your own; you may want to perform a pen test as an internal effort. You may want to be a more educated consumer when purchasing these types of security services. Whatever the case, understand that this book is not about hacking techniques only.

# The Business

Like it or not, this is a business. It would be horribly irresponsible of me to simply teach you the technical aspects of this trade without mentioning some of the nontechnical issues and challenges you may face out there. The field is in its infancy, and there is a large shroud of mystery surrounding what we do. To help dispel this mystery, you must be clear about what can be expected of you and what you require from your clients.

If you are doing an internal pen test, do yourself the justice of being as objective as you can. Most clients I work with would never even entertain an internal audit due to the subjective nature of the people on their teams. This explains why the bulk of pen testing work is outsourced to entities that specialize in this type of work. An outside entity is typically more objective because its success depends on this objectivity. When you are that outside entity it is in your best interest to remain objective and get to know your clients only in areas relevant to your pen testing endeavor.

Most clients you will work with are probably typical IT staff and management, which means that they have no real in-depth understanding of, or background in, software. (They will probably have a networking and operations background.) Whether your clients admit it or not, if they are not a software engineering team, applications and software are most likely a source of blackbox style mystery to them. On the other hand, they likely have familiarity with tools like Nessus and the results they generate, so you will hear that they are after a Nessus-like scan of their Web applications — (you should find those comments amusing — but not in front of the client!).

> *This is not to say that there aren't some automated tools out there that do an amazing job, and I cover some of them in the attack simulation chapters. But I have yet to see a tool that is all encompassing and gives purely accurate results with no false-positives.*

I have worked on many projects where the Tiger Team reports test results. The client briefly states that the Web app results (usually my area of responsibility) were very useful and that they will get someone to fix all of the issues found. Nothing else gets discussed in the section of the report that relates to Web apps. But when the server, OS, security policy, and network sections are discussed, the client asks tons of questions and displays an enthusiastic interest. The general consensus is that clients pass quickly over the Web application information because they are hesitant to engage in conversations about subjects they just don't understand. Imagine discussing HTML forms and how they relate to SQL Injection attacks with a person whose professional space is based on deploying Windows servers for file and print services. It is just out of their scope. I don't mean any disparagement by this; you just have to understand what you are stepping into when doing this type of work.

## Requirements

If you come from a software background, you already know that getting solid and concrete requirements is one of the biggest challenges with any software engineering endeavor. Well, pen testing is no exception, and in one respect it is even worse than software development. When you're developing custom software, at least your stakeholders and audience have some concept about what the final product should look like. When pen testing Web apps, your audience and stakeholders will most likely be technical, yet they will have very little to offer you in terms of their requirements, and worst off, their expectations. Hence, gathering requirements and setting realistic expectations will be challenging. Unfortunately, typical requirements you will hear out in the field are vague and they go something like this:

- We [want | need] to know how secure [we | our Web applications] are.
- What is our level of exposure?
- Can our business-critical application, X, be hacked?
- Is our application safe from insider attack?
- What risk are we running when doing business on the Internet?

It is your responsibility to gather clear requirements from your clients. And it is your responsibility to convey those requirements back to them for agreement. Do not start any work without clearly establishing the goals and boundaries. The client, assuming you are not auditing your own shop, should dictate the guidelines (such as approved time slots and boundaries for attacks) for the audit even though sometimes they do ask for your input.

One of the critical areas of clarification and agreement is that of *blackbox*, *whitebox*, or *greybox* testing. This really is strongly coupled with your client's perceptions of their risk level. But you need to understand what you are up against in order to provide time and cost estimates for the project. I can tell you that performing a true blackbox test with "zero" knowledge of the target could be a daunting task. It is certainly not for the weak at heart.

Another major area of up-front discussion and clarification is whether or not the client wants an external test, internal test, or both. External tests are sometime perceived to be the only valid test based on the erroneous notion that an attack, or real threat, will only come from an external source. Many target entities feel that way even though they have read the FBI and CSI reports about the majority of attacks coming from the inside. Understand the implications for you as an entity providing professional services. The client needs to realize that if you audit externally, the true level of overall risk associated with the given targets is not assessed.

## Rules of Engagement

Cyber criminals will do a lot of things that we, as "friendly hackers," will not and cannot legally do. The line you cannot cross, the so-called *Out of Bounds Behavior,* must be defined and adhered to. Assuming that the client understands all of the possibilities in an endeavor such as pen testing is a huge mistake. The bad guys will not balk at using destructive tactics because they have no boundaries. It is prudent to understand how far real cyber criminals might be willing to go, but it is just as important to establish professional boundaries for yourself — and stay within them.

People who do this for a living are typically brought in as the application and software experts within a full penetration testing team (that is, Tiger Team, and so on). As such we do not engage in many of the practices common to a full penetration test team. We are software experts and stick to that realm. The following table lists some general rules of engagement that have proven beneficial over time and throughout many different projects with a multitude of clients. These are best practices within the context of this professional endeavor.

| Attack Action | Ethical |
|---|---|
| Electronic Discovery — External | Yes* |
| Electronic Discovery — Internal | Yes* |
| Social Engineering by Telephone or Mail | No |
| Adopt Employee Identity | No |
| Engage in Psychological Warfare with Employee(s) | No |
| Break into Employee Workstations | Yes* |
| Take Known Destructive Action — Production | No |
| Take Known Destructive Action — Non Production | Yes |
| Attack with Actual Data Injections — Production | No |
| Attack with Actual Data Injections — Non Production | Yes |
| Target Production Systems | Yes (with great care) |
| Read Sensitive Material | No |
| Save Sensitive Client Data | No |
| Pretend to Be Someone Else | No |
| Dumpster Diving | No |
| Target Sensitive Corporate Resources | Yes |
| Employee Extortion, Blackmail, or Coercion | No |
| Audit Linked Systems not Part of Original Project | No |
| Audit Resources Linked To, But not Belonging To | No |

*Assumes client/target has granted permission.

## *Self Protection*

Never do anything against client target hosts without explicit consent, in writing, from someone in the client organization authorized to give such permission. Things will break and data will be exposed (to you) when you do this type of work. These things are inevitable, so to protect yourself, please read the following suggestions and use common sense:

- **Always request non-production systems.** Many times you will hit different systems for different tests. For example, the common thread I have encountered is to hit production systems for

any test areas that don't involve actual data injections or submissions. For testing the data-related areas I am provided with a mirrored, non-production environment of the same target. For obvious reasons, avoid production systems when you can. When you can't, be extremely careful.

- **Get the client to establish, agree upon, and when appropriate, announce clear time frames for the pen testing exercise.** You will find that your test is rarely ever approved as a business-day, peak-hour effort. Be aware of the fact that sometimes not all the internal folks know you are doing this. I have done remote pen tests where the target's security team had to spring into action in the middle of a weekend night to stop me. They didn't know I was doing attack simulations, and I didn't know that they were not aware of it. It was part of the overall results the client entity was interested in. Be clear on your accepted time frames for attack simulations and hard stop dates if there are any.
- **Get the client to clearly agree that you are not liable for something going wrong that may be triggered by your actions.** You just don't know what is happening behind the scenes sometimes. A good example is one target I worked on that had mail functionality that was triggered when data was submitted via a particular form. I was never told of this, and I forcefully attacked the form, many, many times. The target application basically ended up causing a Denial of Service condition on the entity SMTP infrastructure and people had to get out of bed to resolve this. Establish the fact that you are not legally liable for mishaps and get evidence of the fact that they understand, and agree to, the risk.
- **Find out up front if your target entity has any Non Disclosure Agreements (NDA) that have to be signed prior to doing this type of work.** Handle these requirements prior to even digging DNS for their information.
- **Make sure you have entity resources on call during the approved time frames for your work.** And obviously make sure you have all of the relevant entity contact information.

Take heed of one final note about the professional aspect of this type of work: Try your very best to appropriately set the expectations placed on you and the project. Expectations differ from requirements in that they are not hard, documented entities; they are subjective to the party making them. That is, requirements are concrete deliverables, yet there will be expectations about how you go about doing your pen testing work and there will be expectations about what the end-result deliverable is. There is an unfortunately high level of mystery and a lack of knowledge surrounding Web applications, Web services, and information security in general. It is best to spend extra time with your clients prior to kicking off the project in order to manage expectations and keep things realistic.

# Summary

You should now have a solid idea of where this field is going. Web application security testing and secure coding in general will dominate the software industry due to the new levels of awareness that are now pervasive in the industry. Couple this awareness with the legal ramifications of some of the new compliance regulations and you will see an industry ripe for change and enhancements. On top of this, the phenomenon of off-shoring so much software development work, which results in in-house personnel becoming detached from the source code, is creating enormous opportunity for an objective exercise in security scrutiny.

This chapter covered the following topics to commence your journey into pen-testing Web apps:

- ❑ The state of Web application security
- ❑ The case for pen testing
- ❑ The relevance of experience and methodology to a pen tester
- ❑ The mindset of a Web app pen tester
- ❑ The goals of a pen-testing project
- ❑ Aspects related to methodology
- ❑ How this book relates to certain aspects of pen testing
- ❑ Aspects related to the business on pen testing

With all of this high-level information and concepts about the profession of pen testing in hand, you're prepared to go on to Chapter 2 to go over some basic elements you need to understand. These basics will prove invaluable throughout the rest of this book, and throughout your career as a pen tester.