TestRail

Best of the TestRail Quality Hub

Security and Penetration Testing

TestRail

Ensuring security is an essential component of any testing plan. In this ebook, we've gathered several of the best recent articles on this subject from the TestRail Quality Hub. This ebook starts with a tester's first experience in security testing, followed by in-depth looks at vulnerability assessments and penetration testing. Finally, because it's critical to build-in security from the start, this ebook concludes with strategies for secure software development in an agile environment.

Contents

Security and Penetration Testing

Tester's Diary: A Crash Course in Security

Carol Brands, Software Tester at DNV GL Software

One of my goals for this year is to establish a curriculum to help my team learn more about security testing. We're developing our first cloud-hosted solution, and there are many elements of browser-based security testing that we've never experienced, so I've been hunting for articles that are easy to read and offer introductory material on security design.

A few days ago, a security article called "The Beer Drinker's Guide to SAML" came up in a chat room I participate in. The topic looked fun and easy enough, so I read through it. The article explained that SAML — Security Assertion Markup Language — is the go-between for authentication providers and authentication consumers, much like a wristband used in a beer garden. The authentication provider identifies that you are eligible to receive services from a vendor (i.e., checks your ID and gives you a wristband to access the beer garden). The vendor, who is the authentication consumer, then provides services based on that SAML assertion from the provider, similar to how a beer vendor allows you to buy beer based on the presence of your wristband.

Excited by this introductory material, I decided to test my mastery. I immediately messaged our security guru to ask him whether our new project uses SAML. I wasn't even sure I'd used the words correctly, but I was hoping the question I asked made sense. His response was, "Good question. I'm going over the security stack with another developer right now. You should really be in here. Come on over to the small conference room."

My jaw dropped. I was only being cheeky with a newly discovered term, and now suddenly I was about to be immersed in language I wouldn't understand in a room with two expert developers. There would probably be code involved. At first, I tried to back out with, "I only knew that term because I just read an article." Luckily, the developer knew me well enough to push me a little, and I joined them.

A Token System

When I walked into the room, there were already a number of drawings on the whiteboard, but the one that remained up throughout the discussion looked a lot like the drawings from the article I had just read. But instead of two boxes representing the authentication provider and the authentication consumer, there were now three boxes: the authentication provider, the authentication consumer and the browser itself.

I soon discovered that we do not use SAML, but we use an alternative with similar enough behavior that most of what I read still applies. Ultimately, it's all about having a token that says who you are and what you can do, having an authentication provider that provides that token, and having a product that consumes the token.

As we walked through the technical stack, we did end up looking at code. They showed me the algorithm where we start interpreting the token that was received in order to decide what permissions a user has in our program. The developer pointed out something interesting. Right now, we only have two types of user: basic or admin. In the future, we will probably have many types of user, and we plan to introduce new modules that will be restricted to specific subsets of user. That means that every time a new role is introduced, this code will need to be updated to make sure each new type of user has access to what they need.

Looking at the code, I could see how it would be easy to neglect adding a certain role to a certain set of permissions. I mentioned it, and they responded that we might alter the algorithm once we have to deal with the more complex set of permissions rules. Even so, I made a note of how we're handling permissions now, so I can write a charter about checking a set of user types against each set of permissions. You never know if that code is going to stick around or not, and this way we'll be ready if the old code is what we have to test against.

Swimming in the Deep End

Even though I was fearful, I'm glad I overcame my fears and joined the developers in their review of the security stack. Reviewing the code wasn't as scary as I thought. Even though a few things they discussed went over my head, I was surprised by how much I understood. I came out of the meeting with new respect for how difficult it is to implement a security system, as well as some self-confidence in my ability to understand technical information. Sometimes you need to take a surprise jump in the deep end to discover how well you can swim.

Security and Penetration Testing

Vulnerability Assessments or Penetration Testing?

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

The terms "vulnerability assessment" and "penetration testing" are often confusing. In many conversations, the terms are either used interchangeably or perceived that one is a subset of the other. Although these two activities can partially overlap, they are distinct activities, and each has its own specific set of goals.

Both vulnerability assessments and penetration tests attempt to identify vulnerabilities that could allow an attacker to compromise your systems. Organizations use the resulting reports to implement controls that help secure those systems. The primary difference between these two activities is that of focus. A vulnerability assessment attempts to identify as many vulnerabilities that exist in a system or environment as possible. A penetration test, on the other hand, is more focused on finding a way to access some protected resource. In other words, a penetration test doesn't involve finding all the vulnerabilities, just the ones that allow a system to be compromised.

Here's some guidance on how vulnerability assessments and penetration tests can be best used to provide the greatest value to organizations of any size.

Vulnerability Assessments

Many organizations that become interested in being more secure start with some level of vulnerability assessment. This assessment considers a specific environment and attempts to identify its security vulnerabilities. The goal is to be as broad as possible and determine which, if any, of the many known vulnerabilities exist in the assessment's scope.

It is common for organizations to have their internal personnel conduct the assessments. For smaller organizations, internal assessments may be all that is needed, but larger, more complex environments may need a more extensive assessment that requires external expertise.

Many vulnerability assessment steps can be automated. There are various software tools for this, including OpenVAS, Nessus Professional, Nexpose and Microsoft Baseline Security Analyzer. There are many more general and scope-specific assessment tools, but these are a few of the more common ones.

The typical approach for conducting a vulnerability assessment is to use a collection of tools that scan a group of systems to see if they can detect any known vulnerabilities. Any previously unknown vulnerabilities that security personnel discover are submitted to general repositories of vulnerabilities.

The most well-known vulnerability repository is the Common Vulnerability and Exposures (CVE) database, which has been maintained by the MITRE corporation for nearly 20 years. Some researchers have proposed an alternative to the CVE database called the Distributed Weakness Filing (DWF) repository. These two repositories contain extensive collections of identified vulnerabilities.

The general vulnerability assessment process is to periodically scan a computing environment to identify as many vulnerabilities found in the CVE or DWF repositories as possible. There are other more detailed steps in the process to ensure each system in scope complies with industry best practices. The team conducting the assessment then creates a report that contains all identified vulnerabilities, ranked by severity, and remediation recommendations for each one. It is recommended to conduct such an assessment quarterly, as new vulnerabilities are discovered daily.

Penetration Tests

A penetration test, commonly called a pen test, is more focused than a vulnerability assessment. The purpose of a pen test is to attempt to compromise a protected resource. Pen testers aren't interested in identifying and documenting all the vulnerabilities in an environment; they just want to find a way to break in. The focus of a pen test is on depth, or how far they can "reach into" an environment, as opposed to the broad focus of a vulnerability assessment.

Pen testers may initially use some of the same tools that vulnerability assessors use, but the goal is just to identify the easiest access path. While a vulnerability assessment may rely on automated tools, pen testing normally extends far beyond software tools. An effective pen test is quite labor-intensive and relies more on skill and experience than most vulnerability assessments. The main reason for this is that pen testers must effectively assess an environment and capitalize on any security weaknesses. These weaknesses don't have to be technical gaps, either: People are often the weakest link in any strategy.

One of the biggest differences between vulnerability assessments and pen tests is the use of social engineering. Pen testers can essentially use any means, including technical and nontechnical tactics, to compromise existing security controls. Carrying out social engineering attacks can be fun and productive. Regardless of how sophisticated an organization's technical security controls are, good pen testers know that it only takes one careless or distracted user to provide an opening that can be exploited. People are hard to assess for their ability to resist social engineering attacks, and they generally want to be helpful, so many pen testers leverage people in their attacks.

While a vulnerability assessment report will contain details of many vulnerabilities, the pen test report simply details in what ways the attacks were successful. While the pen test report may contain a narrative of attack paths that were not successful, the bulk of the report focuses on what vulnerabilities allowed the compromise to succeed and how the attack could have been stopped. The content of the report helps the organization determine how to deter real attackers from using the same tactics to break in.

Pen tests can be carried out less frequently than vulnerability assessments: Annual pen testing should be sufficient.

Which Should You Choose?

Vulnerability assessments and pen tests each have value and can help make organizations more secure. The right choice for a specific organization and circumstance depends on several factors, but the most important is probably how mature the organization's security policy is.

If your organization is just starting to focus on security, a solid vulnerability assessment is a great place to start. You'll likely uncover more vulnerabilities than you thought were present. If this is your first time conducting a vulnerability assessment, try using a straightforward software tool like Nexpose or Microsoft Baseline Security Analyzer. These tools are fairly easy to learn and use and will provide a good list of discovered vulnerabilities. After you've carried out a few assessments, take a look at OpenVAS or Nessus Professional. These two are a little more complex but provide far greater coverage of environmental vulnerabilities.

On the other hand, if you have already conducted several vulnerability assessments and want to see how well you've secured your system, then a pen test may be in order. Pen tests may be conducted by internal personnel, but it is a little more common to engage external resources that excel at conducting pen tests. Carrying out effective pen testing requires a particular set of skills, and those skills take time and experience to acquire. A good pen tester can efficiently devise an attack plan that follows the most likely weak controls. Those are the ones you most need to mitigate.

As your security readiness matures, you'll probably use both vulnerability assessments and pen tests. They work well together to make your environment difficult to compromise.

Security and Penetration Testing

5 Types of Software Security Tests

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

Companies large and small need comprehensive software security testing, but there is a lot of confusion about what that actually means. How do organizations go about conducting software security tests? What types of test should you run? And how do you know if you've done enough?

The Open Source Security Testing Methodology Manual is a great resource that explains what security testing is all about. The OSSTMM covers far more scope than just testing software, and it should be in any security tester's library.

There are many types of security tests. Knowing when to conduct each type of security test (and when not to) can make your application more secure and reduce the resources spent to get there. In this article we'll look at five main types of security test and recommend when to use them most effectively.

Types of Software Security Tests

01
Risk Assessment
02
Security Auditing
03
Vulnerability Scanning
04
Security Scanning
05
Penetration Testing

Because we're focusing on software security testing, let's align test recommendations with the phases of the software development lifecycle (SDLC):

Planning: The project management activities that help organize software development tasks

Requirements and analysis: The process of collecting business requirements and translating them into technical requirements

Design and prototyping: This phase consists of defining the way technical requirements will be implemented, producing software specifications

Coding and development: Application developers write code to implement the specifications created in the previous phase

Testing: Once the development phase produces software that can be executed, this phase consists of running specified tests to ensure the software satisfies the specifications

Deployment: Once software passes the testing phase, the software is placed on production platforms and made available to users

Maintenance: This final phase includes monitoring and updating configuration settings to optimize the software's usability and responding to any identified weaknesses in the software

Integrating security into software must start at the beginning of the SDLC process. In the past, software development organizations attempted to defer security implementation until the deployment phase. The thought was that forcing security into the process too early would waste time and resources. But they were wrong. Designing security into software from the very beginning is far more efficient than trying to add it later.

Corresponding with the phases of the SDLC, here are five types of security tests every initiative should include:

1 Risk Assessment

A structured analysis of known vulnerabilities, risks, threats, and probabilities of threats being realized. All of the information collected allows assessors to rank threats and determine which require the most attention

2 Security Auditing

The process of comparing observed artifacts with policies or requirements. In the context of software development, this often includes reviewing source code for compliance with development standards

3 Vulnerability Scanning

A series of tests to determine if any known vulnerabilities are present in the tested environment. Many vulnerability scans are automated and can identify the presence of a wide variety of vulnerabilities in software or the environment in which software operates

4 Security Scanning

While this appears to be similar to vulnerability scanning, security scanning expands the scope of tests to include all aspects of a computing environment that supports application software, including network and physical computing components

5 Penetration Testing

Activities in which security professionals attempt to "break into" software to identify existing vulnerabilities. This type of testing is also called "ethical hacking"

Each type of security test exercises a different aspect of your software environment. These tests sometimes overlap one another, but they work well together to identify security gaps at multiple levels — none of them can provide complete protection by itself.

Choosing the Right Software Security Test

While there is no wrong time to run software security tests, you will get the best results by conducting security tests that map to your current phase in the SDLC. As many software development organizations use agile or another rapid development methodology, development phases may not be very clear.

Here is how the five types of tests outlined above map to SDLC phases:

Software Security Test	SDLC Phase(s)	Comments
Risk assessment	Planning, Requirements, Design	Conduct a risk assessment as early as possible. You must have an idea of the risks you are facing in order to do a good job of producing software that is resistant to the most important risks.
Security auditing	Coding, Unit Testing	As you write components of an application, one of the best ways to ensure security is to adhere to secure coding standards. Code audits can help reveal gaps and violations of coding standards.
Vulnerability scanning	Integration Testing, System Testing	Once you have a working application, you can see how well it resists known vulnerabilities — before your users start using it.
Security scanning	Deployment	After deploying software, it is important to step back and assess the overall security of the entire environment.
Penetration testing	Maintenance	Regardless of how aggressively you pursue secure software, some vulnerabilities don't materialize until software is in a live environment. A skilled penetration tester can often find ways to compromise software that are difficult to foresee prior to deployment.

Knowing your environment and development methods may result in conducting tests in a different order. For example, you may need to run a vulnerability assessment initially to provide input to your risk assessment. That's fine. Your testing regimen should fit your organization.

The point is to make your software more secure, not to just say that you ran security tests. Every test should provide results that you can use. If it doesn't, your time would be better invested doing something else.

Plan for Security

The main takeaway is that security testing for software is far more than just checking authentication in unit testing. While testing security during unit testing is important, it is only one small part of testing security in a software development project.

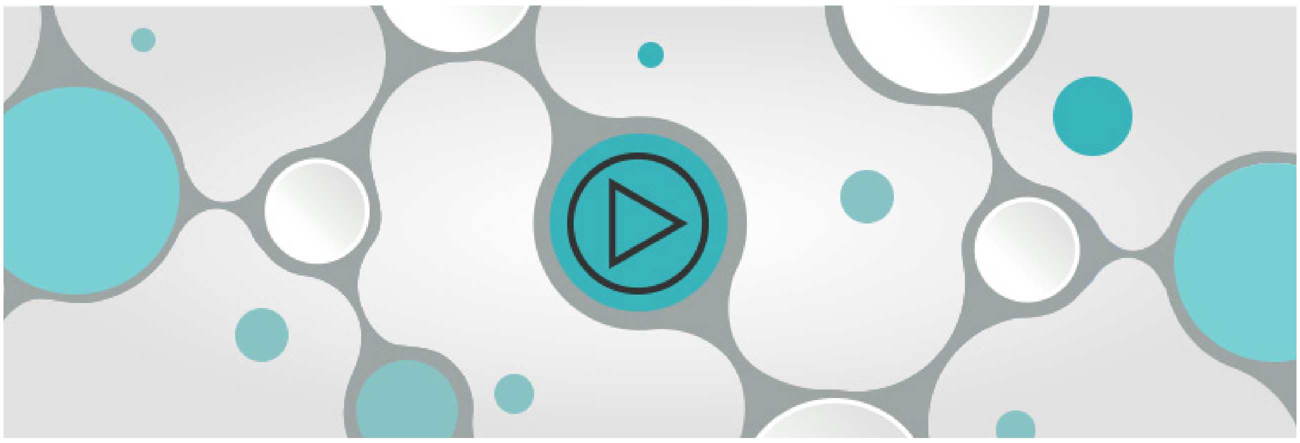
Plan ahead and build security testing into every software development project early in the process. You'll end up saving time and money, and you will produce software with fewer security vulnerabilities.

Penetration Testing 101: How to Get Started

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

You'll find Penetration Testing, often just called Pen Testing, explained differently depending on where you get your input. Most sources agree that Pen Testing is essentially an ethical hacker carrying out attacks, like a malicious attacker, to help expose environmental vulnerabilities. While accurate, that's a bit of a convoluted definition. Let's pull it apart a bit.

First and foremost, Pen Testing is an activity intended to help an organization – not to hurt it. The idea is to have security professionals, (i.e. the "good actors" or "white hat hackers"), act like attackers (i.e. the "bad actors" or "black hat hackers".) Pen Testers do this to expose weaknesses, or vulnerabilities in systems, networks, and devices, before real attackers find them. Pen Testing can be a valuable part of a solid security if it is implemented well and the sponsoring organization responds to the findings.



Being a Pen Tester can be fun. You get to act like an attacker without actually being bad. So, how does one get started in having fun and making environments more secure at the same time? A quick Internet search will return many articles on how to do just that. The problem is, many of them jump into the process too far in. Its easy to find books and articles on "Pen Testing using ______ (insert your favorite tool or scripting language here.) Common titles include Pen Testing using Kali Linux, Python, Bash Shell, Powershell, Perl, and the list goes on and on. The point is that many of the resources you'll find focus on tools, as opposed to the actual process. A much better place to start is at the beginning.

Understanding Pen Testing

Pen Testing is far more than just running software. Pen Testing software tools are just that – tools. If you don't really know why you need each one, you aren't adding much value to the process. I think of Pen Testing like the process of flying an airplane. If you know what you're doing, flying an airplane isn't really that hard. But you just don't jump into an airplane and go. You need some training and experience on things like weather awareness, airplane preparation, airport and airspace navigation, and interacting with air traffic control. And all that is in addition to knowing how to actually fly an airplane. Successful pen testing, like flying, is all about being prepared and planning well.

Pen Testing is all about looking at a computing environment to find the vulnerabilities an attacker wants to find and exploit. You can only do that if you thoroughly understand how your environment works, its architecture, and its attack surface. That's a LOT more than just downloading a Kali Linux virtual machine image and running some tools from a menu. But that doesn't really answer our question yet. Where do I start to become a Pen Tester?

Know Thy Environment

Before you start running Pen Testing tools, you need to know a good bit about your network, its connected systems, and its devices. In almost all cases, a good place to start is learning about TCP/IP and its administration. System administration skills are essential. Know how to add computers and devices to your network, and then configure each one. Can you add a new database server to your network and ensure that only the ports needed to operate are opened? Can you add a new user to your environment and permit that user to access only required resources? Those skills will help you to better understand how various components work together in an environment.

Here is a great self-test to see how well you understand your environment. Describe the process that starts with a user entering a URL in a web browser, and ends with active content being rendered in that user's browser. Can you describe precisely what traffic travels across the network, and where? Can you explain all of the devices and systems in your network involved in the process? If so, you have a good understanding of all of the pieces – and the opportunities for vulnerabilities to exist. A solid understanding of network protocols is needed to explain the round trip of a web request. You can bet that the attackers that launch successful attacks understand these protocols, implementations, and their weaknesses.

Scripting is Helpful

Another key skill a good Pen Tester possesses is the ability to write scripts. You don't have to be a scripting guru, but a good knowledge of at least a couple scripting languages will make your life easier and unlock lots of neat tools that others have written for their own use. If you are completely new to scripting, I'd suggest learning Python and either Powershell, if you primarily work in Windows, or bash shell scripting, if you primarily work in UNIX or Linux. Knowing these two scripting languages will enable you to build your own toolbox and leverage your time when conducting Pen Testing.

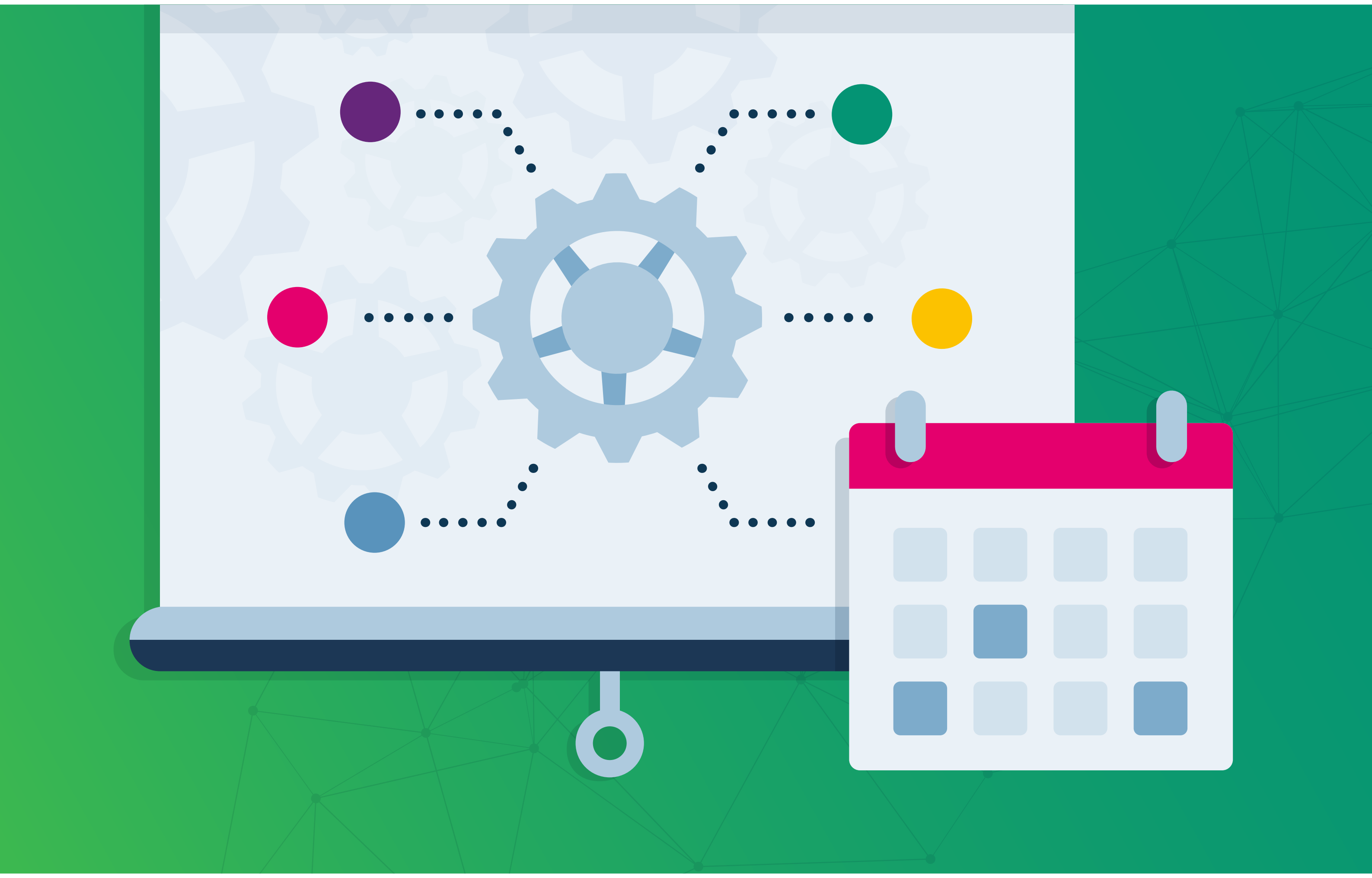
What about Permission?

Before you ever run the first Pen Test, be aware that you are simulating attack activity. Some of the activities you'll carry out in Pen Testing can be dangerous, or even outright harmful. NEVER conduct Pen Testing activities without explicit permission from the system owners. Pen Testing without appropriate permission can result in civil or criminal proceedings. In short, if you don't have permission (in writing), you could be sued or prosecuted for your activities. Don't risk it. Get explicit permission in writing first.

Permission includes the network you use as well. If you are operating within a single organization, make sure you have permission to access the network as well as computers and devices. Your tests may cause excessive or malicious traffic that could interrupt normal operations. If you are conducting tests remotely, be aware that your Internet Service Provider (ISP) may very well take a dim view of having their network used for attack purposes. You could see your home or business Internet service terminated. Understand your ISP policies.

Finally, Using the Tools?

Once you have the basics under your belt, its time to roll up your sleeves and get started. But that doesn't mean to start running Pen Testing tools (yet). There's still a lot left to do. In the next two articles, we'll talk about how to plan your Pen Testing activities as a project, and how to determine which tools you'll need. Planning is crucial to determine the scope of your tests and set expectations for your stakeholders. After that we'll talk about Kali Linux and a few alternatives to get started. But you'll have to wait until next time for those details. In the meantime, review the basics. Knowing how your environment really works is the most important requirement to becoming an effective Pen Tester.



Penetration Testing 102: Managing Pen Testing Like The Project It Is

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

In the previous article, we discussed the basics of pen testing, and focused on the activities you carry out during a pen pest. This time we're going to look at the whole process. In fact, we're going to focus on a strategy that can dramatically increase your probability of success and make the whole endeavor more enjoyable. I'm not going to let you in on any huge secrets – I'm just going to explain how treating pen testing like a project makes sense.

The Project Management Institute (PMI), the "world's leading not-for-profit professional membership association for the project, program, and portfolio management profession," has a lot to say about projects. Since 1969, PMI has maintained a central repository of project management knowledge, called the Project Management Body of Knowledge (PMBOK). The PMBOK provides invaluable guidance on how to manage projects of any size. PMI also administers the world's most popular project management certification, the Project Management Professional (PMP).

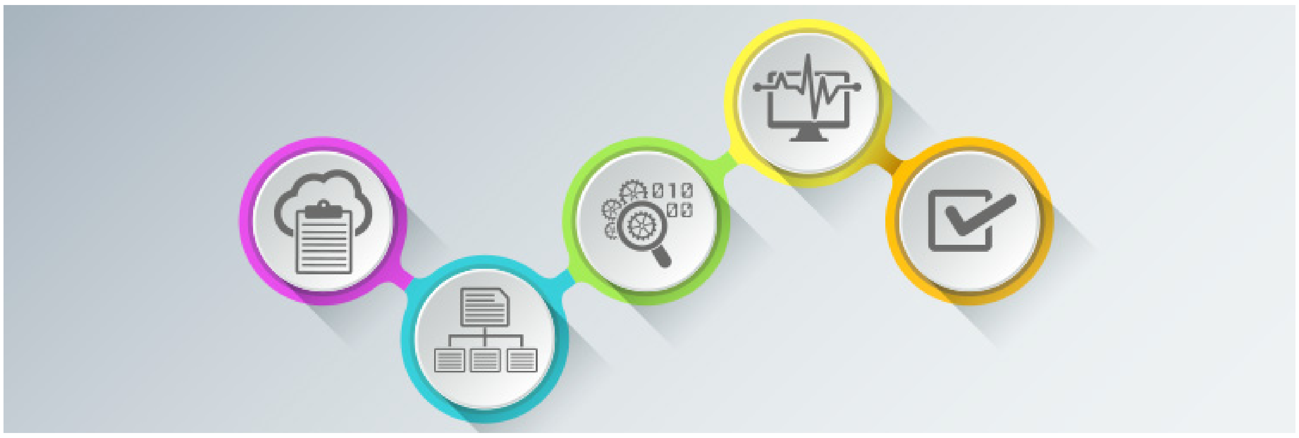
So that's the PMI pedigree. How does that apply to pen testing? PMI defines a project as:

... a temporary endeavor undertaken to create a unique product, service, or result.

You can, (and should), use the PMBOK to help manage pen testing activities. You don't have to use the whole thing, and you don't have to hold the PMP certification to use the great advice in the PMBOK. What you will get is an approach to pen testing that is repeatable, reliable, and can be refined over time to increase your quality. When using good project management practices, you'll be able to keep your pen testing activities on track, and know early on if things start to depart from what you expect. It keeps you from constantly running into the unknowns that result in budget overruns and late deliveries.

So, to answer the original question, a pen testing endeavor is a project. Let's look at how we can treat it as such.

Don’t Projects Make Everything Slower?



An organized pen testing project is one in which the tester doesn’t start out just running test. You have to start at the beginning. In fact, the more time that you spend up front, the better quality your results will be – and you’ll have an easier time repeating the process next time. And, in most cases, you’ll actually spend less time on the project since you won’t have as much re-work. Let’s take a high-level look at PMI’s view of how you should organize any project.

You can group project activities into five separate phases:

Initiating – This is where you develop the Project Charter, which contains a high-level statement of what your project will accomplish, along with the project sponsor’s written authorization to conduct the work.

Planning – In this phase you clearly define the scope of work, and then create a budget, schedule, and resource requirements. Here is where you can start avoiding overruns by putting it all out there in your plans. For pen testing, you’ll select the targets for your tests and the actual tests you’ll run in this phase.

Executing – Simply put, you follow your plans from the previous phase.

Performance and Monitoring – As you carry out your project activities, you have to keep an eye out to identify any issues early. Identifying issues early can dramatically reduce the effort needed to fix those issues. For instance, if a test that was supposed to take 45 minutes is taking nearly two hours so far, you'll know that something is wrong. It is far better to intervene now than wait until perhaps production operations are affected.

Closing – Many project managers either abbreviate or completely skip this part. But this phase can be one of the most important. In closing, you tie up all the loose ends and take the time to document what went well and what didn't – all while it is fresh. This gives you the ability to continually make your next project run more smoothly.

Why Should I Waste Time Making Plans?

Taking the time to plan before you act has many benefits. Perhaps the most important one is that you cut down on surprises. If you follow the PMI recommendations, you'll lay out the whole project for the sponsor, (you know, the person who signs the checks), to formally approve the actions you propose. Right there you can avoid misunderstandings by putting things in writing.

Another thing you MUST get up front is written permission to conduct pen testing activities. Such written authorization, (from the owner of affected systems, networks, and devices), is often called the "get out of jail free" card. If your tests cause any damage or trigger malicious activity actions, you really want to have this piece of paper with you.

Planning helps you think about all the things you have to do, sequence those activities, and ensure that you have everything and everyone you need to carry out the plan. And, it allows you to say "No." One of the nice things about having your sponsor officially approve your project scope statement, which is part of the planning phase, is that you can use that to resist adding "more tests" to the project.

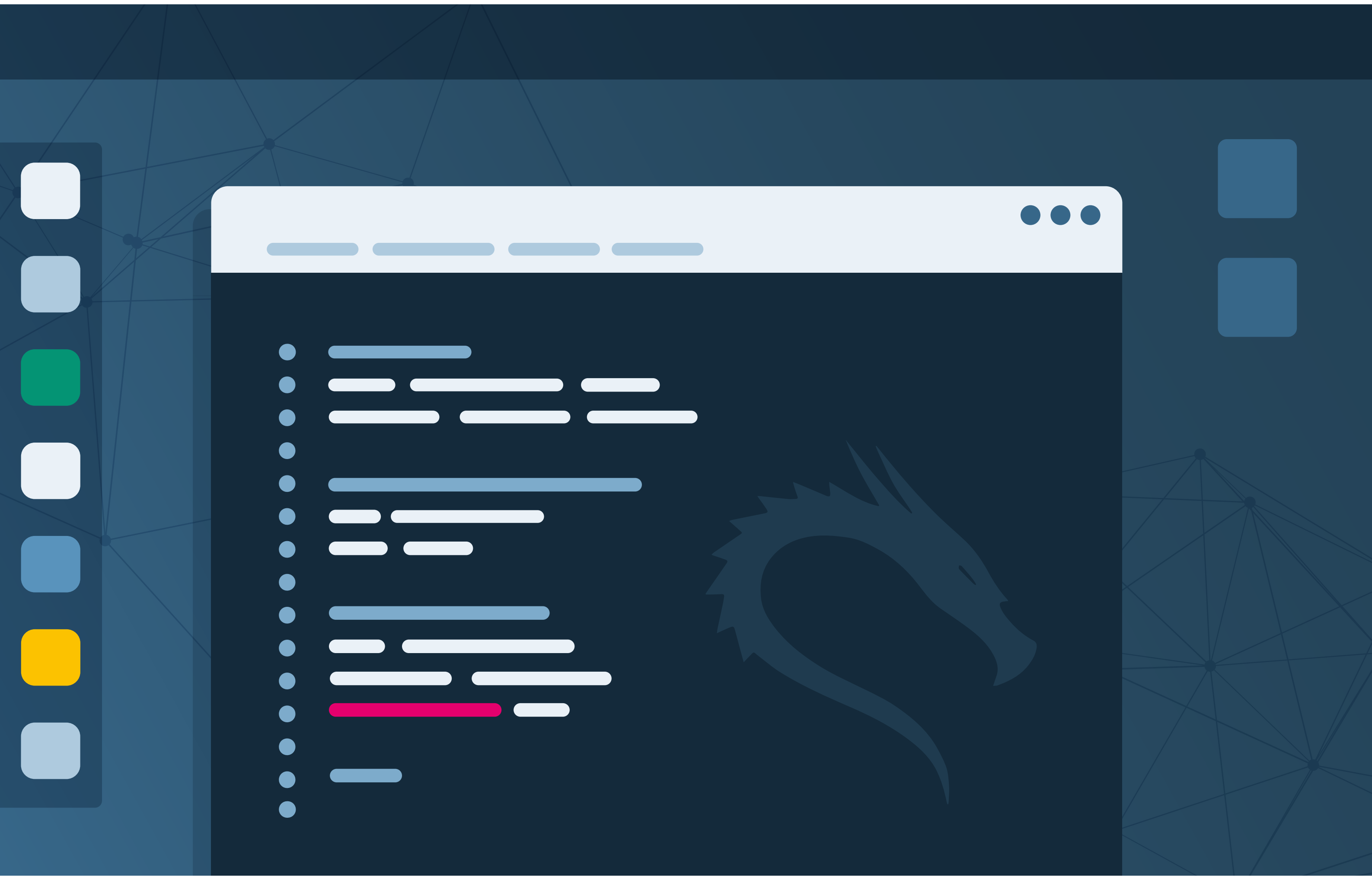
Sometimes you really do need to expand the scope of a project. That's OK. You just have to amend the scope statement, assess the impact (i.e. will it cost more or take longer), and have the sponsor approve the new scope. Without sponsor approval, you can say "No." Many poorly managed projects have failed because the scope grew uncontrolled. We call this "scope creep." Plan well and stick with your plan. You'll be able to approach your testing more efficiently.

The Pen Testing Project Checklist

So, here is a simple checklist that will get you started toward managing pen testing like a project. (For more detailed information, visit PMI's web site and look up the PMBOK.)

1. Develop a Project Charter (high-level description of what the project will do.) You should incorporate all lessons learned from previous projects here.
2. Get the sponsor to sign the Project Charter and all testing authorization forms.
3. Create a Project Schedule, Project Budget, and Project Resources Requirements Documents.
4. Get the sponsor to accept and sign each of these.
5. Execute the planned activities, according to the Project Schedule.
6. Monitor all activities, making changes as necessary to project documents (of course, always with the sponsor's approval.)
7. Create results report and presentations, along with "Lessons Learned" documentation.
8. Deliver reports and presentations.

That's a very simple list of pen testing project steps, but it will get you started. If you plan and manage your pen testing activities like a formal project, you will be able to provide more consistent results and avoid many unexpected surprises.



Security and Penetration Testing

Penetration Testing 103: Why Kali is a Great First Toolset

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

You don't have to visit many "how-to" resources on penetration testing before you read something about Kali Linux, often just called Kali. It is one of the most common and accessible suites of pen testing tools available today. In fact, Kali has pretty much become synonymous with pen testing and hacking.

Kali Linux is one of the many Linux distributions that are based on Debian. It is developed and managed by Offensive Security as the successor of the wildly popular Backtrack Linux distribution. In short, Kali is a pen testing toolbox.

Kali includes over 600 software tools and utilities that pen testers commonly use. The vast majority of these are free and open source. The Kali Tools page lists the tools included in the current distribution.

But Kali isn't just a static collection of tools. The Kali Git development source tree is open source as well. That means that anyone can see what goes into any Kali release and can customize any Kali packages. The list of authorized users who can modify the Kali source is very short and well-protected, but that provides a layer of control over the distribution.

Although Kali is derived from Debian, it does not include all the packages many general-purpose Linux distributions include. That's because Kali isn't for general-purpose users. It is built specifically for pen testers, and pen testers use it on a variety of hardware. You can download Kali to install on Intel PCs, a variety of ARM devices, as VMware, VirtualBox, or Hyper-V images, or simply as an ISO file to create bootable DVDs or USB devices. You can install Kali or just run it from the live boot device.

Why is Kali So Popular?

Try conducting an internet search for the phrase "pen testing tools." You'll find that many of the resources include Kali in their "best of" lists. Kali is immensely popular with both pen testers and hackers. The reason for the cross-cultural appeal is that the tools included in Kali can be used for beneficial or malicious purposes.

So, what is it that makes Kali the de facto toolset for so many pen testers and hackers when they're just starting out?

Here is a short list of the main reasons Kali has gained such a prominent position in the share of pen testing tools:

- **Kali is free:** The price is right
- **Kali is largely comprehensive:** Over 600 bundled tools means there is a good chance you'll find a way to do what you want
- **Installing Kali and accessing tools is easy:** Download the image you want and either run the simple install procedure or create bootable media
- **Kali runs on lots of computing devices:** Kali runs on Intel PCs, many ARM devices, or as a virtual machine in VMWare, VirtualBox, or Hyper-V.
- **Kali has a rich community of users:** Many Kali users interact in various forums across the world
- **Kali is used on "Mr. Robot"**

That last point needs a little more explanation. The USA Network produces and broadcasts a television series called "Mr. Robot." The show is about a cyber security engineer, Elliot, who moonlights as a hacker. Several episodes highlight the fact that Elliot uses Kali Linux in many of his hacking exploits. The show is quite accurate, at least as far as its depiction of hacking techniques and tools. Many wannabe hackers saw "Mr. Robot" and immediately downloaded Kali Linux to become a hacker.

This media awareness of Kali, and hackers in general, has created a growing group of entry-level hackers and pen testers. They generally want free tools that are bundled all in one place to get started. Kali provides exactly that product.

How to Get Started with Kali

While becoming a legitimate pen tester takes time and effort, getting started with Kali is easy. There are only a few steps to get up and running. However, just booting Kali will not make you a pen tester. It won't make you a hacker, either—Kali is a tool. Education and skill acquisition are mandatory paths to becoming good at either pen testing or hacking. But for now, here's how to get started with Kali:

1. **Decide which Kali download file format you need. This will be based on the type of device on which you plan to run Kali.**

   - Install image: This format allows you to burn DVDs or bootable USB sticks. You can boot and run Kali directly or boot and then install Kali.
   - Device-specific ARM image: This download type allows you to install Kali on a variety of ARM devices.
   - ARM build scripts: These are the scripts used to build the ARM images. You can build your own images for custom devices.
   - Virtual machine images: Kali is available in prebuilt images for VMware, VirtualBox and Hyper-V.

2. **Download the file type you need for how you'll use Kali.**

   - If you don't know what to select, just get the Kali Linux 64-bit or 32-bit image.
   - Burn the ISO file to a DVD or USB storage device to create bootable media.
   - Boot Kali from your newly burned boot media and install it, or just run it from the boot media. Running Kali from the boot media is a great way to get started with Kali without having to install it in a hard drive.

3 **Log in and explore Kali.**

- If you install Kali, you will use the root password you provided during the installation process.
- If you run Kali from the boot media, the root password is "toor."

Next Steps

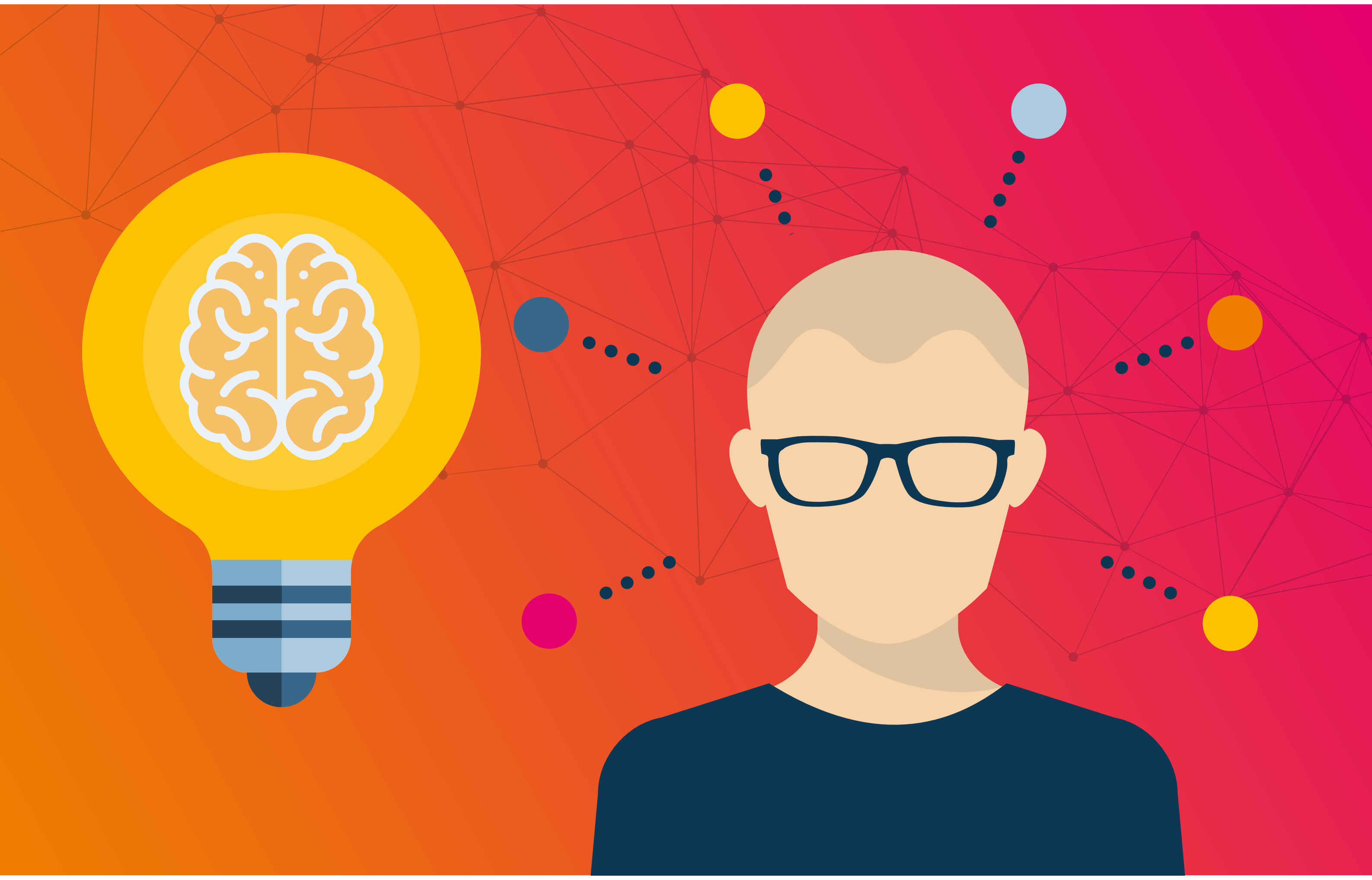
As you explore Kali, you'll notice that the default user interface has the tools organized into categories. Take some time to peruse the menus to see what tools Kali includes.

Once you are a little familiar with Kali, the next step is to really start learning what the tools do and when to use them. There are many Kali and pen testing tutorials freely available online. Find one that you like and work through it.

The best way to learn Kali is to use it. Actually employing the tools in different ways gives you a good feel for how they work. If you take the time to learn about pen testing, you'll gain a good understanding of how Kali fits into the process.

Although Kali is a great collection of tools, it isn't the only pen testing tool suite. There are many more freely available tools and collections, and fully featured commercial options, as well. As you gain experience as a pen tester, you'll find that you need more tools that provide specific capabilities.

But Kali is a very good starting point. It is free, and there is lots of information about it out there. That's why Kali is one of the first tool collections that is mentioned when talking about pen testing. (And, of course, it's on TV.)



6 Habits of Effective Penetration Testers

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

Being a penetration tester is fun, but it can be quite challenging. In some ways, it's kind of like playing a sport. With both types of endeavors, you start off not being very good, and then you spend a lot of time and effort trying to get better. You make mistakes, learn from them (hopefully) and then try it again.

Really good pen testers make it look easy, but they weren't always that good. They had to pay their dues, learn the basics and slowly develop their skills over time.

There really isn't a shortcut to learning how to conduct pen tests well. However, you can avoid some of the missteps and wasted time by focusing on developing some common traits that many effective pen testers possess. As you acquire the necessary skills and experience, paying attention to cultivating these six habits will help you to become a solid pen tester.

Habits Common Among Experienced Pen Testers

If you get the chance to observe several people who make their livings as pen testers, you'll likely notice a few ways in which they are similar. These shared habits can reveal a lot to lesser experienced pen testers, and emulating these common habits can help accelerate the learning process.

These basic habits tend to help internalize much of the foundational knowledge that pen testers need to be effective, but this isn't an exhaustive list. They won't replace the need to acquire lots of knowledge and skills, but they can help organize the many requirements of pen testing into manageable areas. Although there are many habits that can help make you an effective pen tester, here are the six most common ones.

1. **Hanging out at the command prompt:** Although GUIs often look very nice, most pen testers agree that the real power and art of pen testing is at the command line. You'll see most experienced pen testers drop to the command line, regardless of operating system, and spend a good bit of time there. That means for Windows testing, you need to learn the command prompt, batch file scripting and PowerShell. For Linux, become familiar with at least one shell, such as Bash, and learn to write shell scripts. Command line interface (CLI) utilities give you the flexibility to poke around systems in many ways.

2. **Talking their way out of problems:** Pen testers need good soft skills in addition to technical skills. The ability to use social engineering is one of those skills that some people are just born with. They seem to be able to talk themselves out of any predicament. Developing this habit makes you better able to leverage those around you to participate in your pen test. In other words, perfecting the ability to smooth-talk your way through a social engineering exploit is like working on sleight of hand for a magician.

3. **Solving puzzles and playing games**: Good puzzles and games that make you think are great training grounds for pen testers. Those who love the challenge of figuring out the solution tend to map that desire to pen testing. That's because pen testing is really all about solving a few puzzles.

4. **Strongly preferring closure**: This is a nice way to say that pen testers tend to detest open-ended sequences of events. People who enjoy pen testing generally find leaving things undone to be uncomfortable. That means that they generally stick with a thread of activity until it is finished. This habit helps pen testers stick with a process that can become tedious until the conclusion of a test sequence is reached. Closure is important in pen testing to ensure that the tests have provided sufficient coverage of the test domain.

5. **Scripting everything**: It isn't necessary to write a shell script or batch file for every activity, but it does often help. People who script most activities tend to know their operating systems quite well and are comfortable using the available tools to get work done. Additionally, scripting many tasks leverages automation and can increase the amount of work accomplished over manual command entry.

6. **Getting the gadgets:** Because technology changes so rapidly, good pen testers often spend time and money to stay up to date with the latest techno-gadgets. They are often some of the first to acquire new technology, and they often gain an edge by playing with new gadgets first. This natural curiosity leads to a desire to discover how things work — and that leads to a good pen tester.

Why Are These Habits Important?

Remember that there is no easy path to becoming a good pen tester. Becoming good takes time and lots of practice. But developing the habits listed above can help.

These habits aren't a prescriptive approach to getting good, but they do seem to be common habits among effective pen testers. That means there appears to be some value to them. One way to get good at anything is to observe those who are already good and emulate them.

Pen testing is all about carefully searching for vulnerabilities in infrastructure and software. To do that, you need to have a very good understanding of what you're examining. Medical doctors study the human body for years before they ever study what can go wrong with it. Without a deep understanding of how the human body is put together and how it works, it would be nearly impossible to understand the impact of a heart attack. That one event is important in the context of the whole body.

Pen testing is somewhat similar. You must have a solid understand of operating systems, networks, hardware and software before you can effectively search for security vulnerabilities. Most of the habits we discussed above all depend on a good understanding of the technology that makes up computing infrastructure.

On the other hand, some are just interesting personality traits. But if you practice these habits, you'll find that you will have to learn more about your computing environment, and that includes social engineering — remember that users are part of the computing environment.

How You Can Develop These Habits

Let's assume that you currently possess none of these habits. That's OK. You don't have to habitually do all the things on the list. But if you want to improve your pen testing skills, it would be a good idea to try to incorporate as many as possible into your normal activities.

Try to cultivate these habits whenever you can. Do you like to solve puzzles or play games? Then do those activities more. Do you tend to type commands over and over? Write a script instead. Learn how to use parameters and input/output, and you'll both learn something useful and lighten your workload.

The point is to think about what makes an effective pen tester. It is more than just knowing some tools. A good pen tester really understands the environment and leverages the actors, both human and technological, to explore and find vulnerabilities. Happy hunting!

Security and Penetration Testing

Are Blockchain Apps Really Secure?

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

Blockchain technology allows multiple people or devices that don't trust one another to share trusted data. The way blockchain works makes it nearly impossible for a malicious party to tamper with the data without being caught, so it has been called "unhackable technology."

But is blockchain really unhackable? Let's look at the security and privacy that blockchain really provides.

Blockchain is widely predicted to disrupt business practices as we know them. Most expectations are that blockchain technology, and the applications that run on the technology, are more secure than legacy alternatives. But is that truth or hype? The answer, as with nearly all technology questions, is "It depends."

Blockchain app security depends on doing these three things well:

- Applying blockchain in appropriate use cases
- Designing security into the application from the beginning
- Extensively testing for secure operation before deploying the application

These three principles may sound pretty obvious, but they are often overlooked or ignored when schedules and budgets get tight. But because blockchain is fairly new, it's essential to be purposeful when the goal is designing secure applications on this technology.

The Basic Tenets of Security: The CIA Triad

Before getting into whether blockchain apps are really secure, let's look at what security means: Blockchain technology is all about sharing a ledger of data among un-trusting nodes. The primary focus is on the security of the data.

At its core, data security is based on ensuring three properties:

- Confidentiality: Only authorized subjects can access protected data
- Integrity: Only authorized subjects can modify protected data
- Availability: Authorized subjects can access protected data on demand

These three tenets of security are so foundational, they are often referred to as the CIA triad.

Most legacy approaches to data security depend on some trusted authority to limit access to data and enforce access restrictions. The key to this approach is that all users must trust the authority. Blockchain makes it possible to remove the trusted authority and still provide security guarantees.

Confidentiality

Every authorized user can access the whole blockchain. For public, or permissionless, blockchains, that means everyone can see all the blocks. For private, or permissioned, blockchains, the blockchain owner enforces access restrictions as to who can access the blocks. Because private blockchains reintroduce the trusted authority, we'll just focus on public blockchain security in this article.

Public blockchains provide confidentiality through encryption. Instead of preventing access to the data, access is controlled by whoever has the decryption key. Even though everyone else can see your data stored in the blocks, only you (and anyone else you give your key to) can decrypt the block contents. Of course, this leads to other key management issues, but we're focusing on the basics of security guarantees here.

Integrity

This is an area in which the blockchain approach is novel. Blockchain data integrity relies on several complementary techniques

Firstly, all new blocks must be approved by more than half of the participating nodes in order to be added to the blockchain. That means that while anyone can add data, everyone must agree that the data is valid for the blockchain. In blockchain, the majority rules. All nodes agree to accept the majority view of correct data.

Once a block is added to the blockchain, it cannot change. Blockchain implementations use strong cryptographic hashes to ensure that each block remains in its original state once added to the chain. Hashes make it fast and easy to verify the integrity of the blockchain, and any small change to any block is immediately evident.

Availability

The last basic security guarantee is that data is available on demand. Most legacy application data storage techniques rely on central storage, which implies a single point of failure. By its very design, blockchain technology ensures that a complete copy of the entire blockchain is shared among many nodes, and the integrity of the blockchain is maintained among all the nodes.

In other words, every full node stores an identical copy of the whole blockchain. If any node fails or goes offline, the blockchain is still available on the other nodes. This design feature removes the single point of failure and strengthens availability.

Blockchain, Apps and Security

Data security is at the heart of all security concerns. Even when we talk about application security, we're really talking about how well the application protects data security. So, are blockchain apps really secure?

While blockchain doesn't automatically provide security, its features do lend themselves well to some aspects of security. You'll get the best results by understanding what blockchain can do for your organizations, and then applying the technology to the right use cases.

However, to complicate matters, all the hype surrounding blockchain technology has introduced some perceptions and beliefs that just aren't correct. Let's look at three major blockchain security myths.

Myth 1: Blockchain encryption makes it secure

This is one comment I see repeated over and over. Transactions that are stored in blocks are generally encrypted, but that isn't required. And encrypting data can provide confidentiality, but only if the encryption keys are managed securely. Managing keys is a challenge that extends far beyond blockchain.

Blockchain isn't built on encryption, but cryptographic hashes do play an integral part. Each block is connected, or linked, to the previous block in the chain by recording the cryptographic hash of the previous block in the current block header. Because the previous block's hash is part of the current block's header, part of that data is used to calculate the hash of the current block (which is then stored in the block header of the next block).

The use of a cryptographic hash provides a guarantee of integrity only. If any block is modified, the hash value of the modified data will not match the hash value that is stored in the next block. That situation would essentially break the chain and data would be easy to detect. While this feature does provide integrity, it does not provide confidentiality or availability.

Myth 2: Blockchain is unhackable

Blockchains are said to be immutable. That means that once a block is added to the chain, it can never be modified. That is generally true, but not as an ironclad guarantee. A better way to phrase it is that blockchains are tamper-resistant — or at least tamper-evident. Hashes provide the ability to easily detect any meddling with the blocks.

However, it is theoretically possible to modify a blockchain and escape detection. The most popular consensus mechanism in blockchains, proof of work, requires that a majority of nodes validate and accept new blocks. Once a majority consensus has been achieved, remaining nodes accept the majority decision and allow the new block. As long as the nodes are free from malicious collusion, this approach virtually guarantees blockchain integrity.

On the other hand, if 51 percent or more of the nodes on a blockchain network agree to falsify a new block validation, an unauthorized block could be added to the blockchain and minority nodes would be forced to accept it. However, the mutual distrust among nodes and the overhead required to carry out such collusion make this type of attack expensive and unlikely.

Myth 3: Smart contracts are error-free

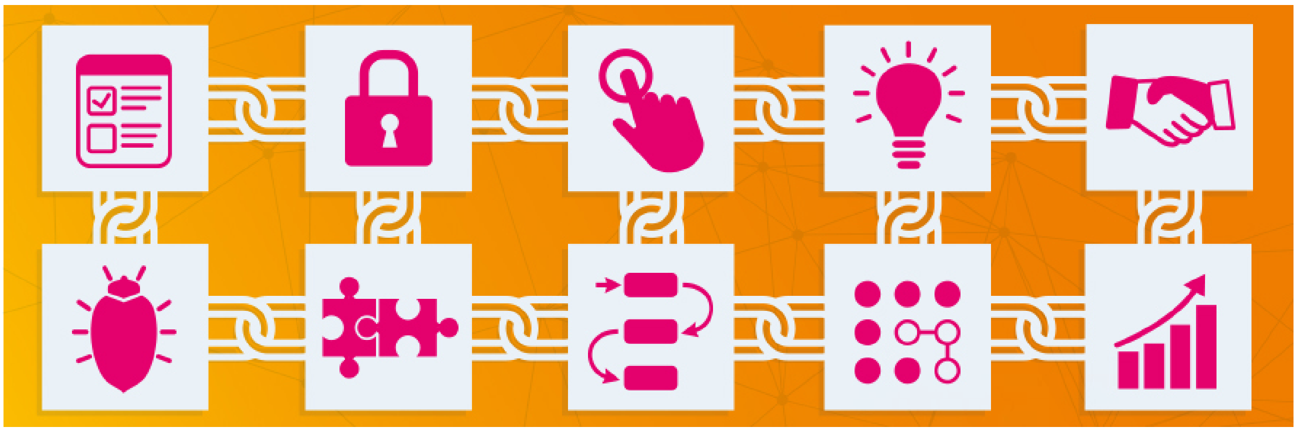
Blockchains store much more than just data. Smart contracts are procedures, or even small applications, that are stored in blocks on the blockchain and run on every node. The purpose of smart contracts is to provide a consistent set of rules for interacting with blockchain data. Smart contracts are crucial for verifying transactions before they are allowed to be recorded in a block.

For example, a smart contract may provide very basic financial transaction validation. Suppose you are buying a cappuccino using bitcoin. Before you are allowed to transfer your bitcoin to the coffee shop, a smart contract would check your bitcoin balance to ensure that you have sufficient funds to continue. This is a simple example of how smart contracts govern blockchain data interaction.

Smart contracts are stored in blockchain blocks (which means they are immutable and can't be changed) and run automatically and consistently. You can't turn off smart contracts, and you can't change them once they've been deployed. They are just blocks of code that developers wrote to carry out some tasks.

Just like with any other software, smart contracts may not always produce the desired results. They may be buggy.

Designing and Testing for Security



Just as with all other types of software, blockchain apps can be secure, or they can lack good security. It is important to consider data security from the very beginning of the design process. And because smart contracts are immutable upon deployment, getting it right the first time is of upmost importance.

If you're wondering how bugs are handled in smart contracts, it is possible to address program flaws. If you do find a bug in your smart contract code, you can simply invalidate the old code and deploy a new version. But you have to be careful that all references to the old code address are updated and that any blockchain data that was added using the old smart contract is valid using the new smart contract. In some cases, the data format has changed. This can cause some headaches when trying to fix bugs in smart contracts and trying to juggle multiple data formats. These issues make it that much more important to get it right the first time.

There is no substitute for fully testing software applications to identify flaws. Blockchain apps are no different from legacy apps in this regard. Comprehensive testing for proper functionality and data security will take more time, but it will always result in cleaner software with less rework required.

At the end of the day, whether blockchain apps are reliably secure depends on the quality of your design and testing.

Secure Software Development in an Agile Environment

Michael Solomon PhD CISSP PMP CISM, Professor of Information Systems Security and Information Technology at University of the Cumberlands

Agile software development has gained tremendous popularity since its introduction as an approach that emphasizes efficiency and timely responsiveness to customers' needs. VersionOne publishes an annual survey of organizations using agile that explores their goals and how well agile helps to meet these goals. The latest is the 12th Annual State of Agile Report, which shows that agile consistently provides customers what they want, when they want it, while requiring less time and cost commitment from the software producers.

Although agile is wildly popular and generally accepted as being effective, it doesn't help software developers address most security concerns. The problem of insecure software isn't a weakness of agile; it is a result of how agile is used.

In agile, user stories provide the specifications for software developers to know what to do. Approved user stories are added to the backlog, which is simply a list of user stories that haven't yet been addressed. Many software developers view security qualifications as nonfunctional requirements, and nonfunctional requirements are depicted as constraints on the backlog, as opposed to separate user stories.

Because security concerns aren't expressed as user stories, the perception is that adding security features slows things down. So as organizations move to agile software development, security often gets left behind.

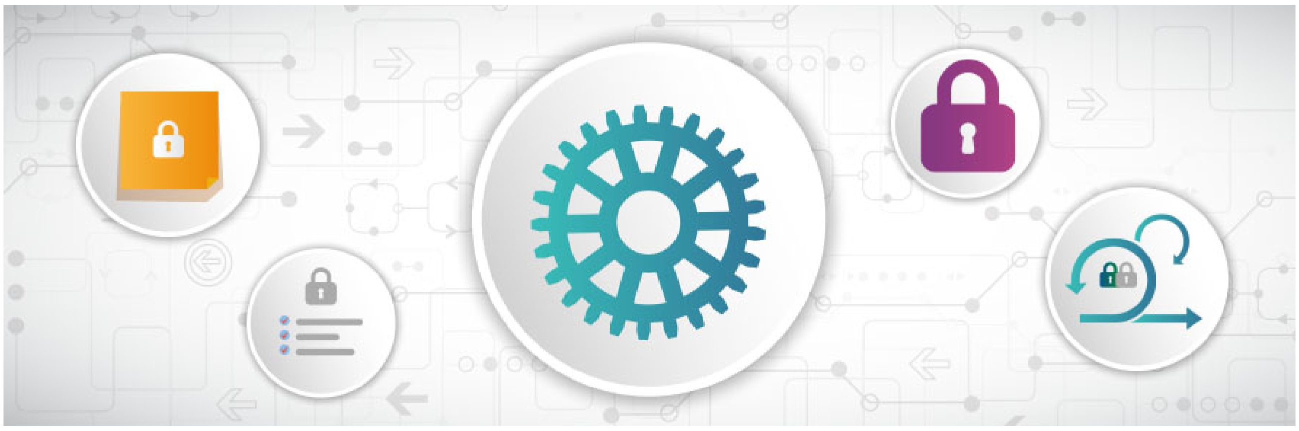
Why Isn't Agile Software Development Already Secure?

In the waterfall method, security planning generally takes place during the initial design phase (or at least, it should). Security is baked in up front and is part of all activities that occur after the design phase. Then, the post-development activities include security testing in the overall functional testing phases.

Security testing validates that the security design goals have been met, along with functional design goals. This approach has the advantage of enforcing application or enterprise-level security policy across all developed software. Further, the post-development testing can validate that security is implemented uniformly across the application.

Agile uses a different approach for software development. Software products are decomposed into multiple builds, and activity to create these builds are called sprints. Sprints are focused on producing narrowly scoped deliverables in a short time frame, commonly two weeks. That means the emphasis is placed on simply satisfying user stories. In other words, success depends on producing a deliverable that meets user story requirements in two weeks or less. Satisfying extraneous constraints is a nice goal, but it's not always a "front of mind" task for software developers. That's why security generally isn't a primary design goal in agile.

Developing Secure Applications with the Agile Methodology



You can develop secure software and use agile. It just takes a slightly different approach to the overall development process, especially how backlogs get built.

Early on in agile history, researchers realized that new techniques were needed to ensure secure software when using agile. In 2005, Mike Siponen, Richard Baskerville, and Tapio Kuivalainen published an academic paper, "Integrating Security into Agile Development Methods." The authors describe the problems with using traditional security approaches in agile development and then propose techniques for integrating specific security features into the development process.

A more recent paper shows that the problem is still one to consider. In 2013, Davoud Mougouei, Nor Fazlida Mohd Sani, and Mohammad Moein Almasi published the paper "S-Scrum: a Secure Methodology for Agile Development of Web Services." The authors define a security-focused implementation of Scrum, called S-Scrum, that incorporates security best practices into agile. These papers point out that security problems are real, but viable solutions to them exist.

There is also a growing number of industry resources that focus on secure agile application development. For starters, Microsoft documents the Software Development Lifecycle (SDL) for Agile. Microsoft has decomposed the most important tasks related to software development and mapped each one to a specific phase in the agile method. Tasks, or best practices, are mapped to one of three categories:

- **Every-sprint practices:** Tasks that should be carried out for each sprint
- **Bucket practices:** Tasks that should be carried out on a regular basis but need not occur in every sprint
- **One-time practices:** Basic tasks that must be carried out once for each project

Their guide should be a mandatory document for any agile shop. You don't have to implement everything, but you should at least compare Microsoft's approach to your own. It is a great checklist to ensure that you aren't omitting any foundational best practices.

Another great resource for secure agile development best practices is the Open Web Application Security Project (OWASP). OWASP debuted in 2001, the same year the Agile Manifesto was published. The site has a very good resource that addresses agile and security: The "Agile and Secure: Can we do both?" presentation covers best practices for developing secure web applications in an agile environment. Instead of categorizing security task like Microsoft does, the OWASP approach is to describe how the agile method should be extended to include security.

Most approaches to developing secure applications in agile focus on a single foundational aspect: creating security-based user stories. Considering user stories are the drivers for sprint activities, it makes the most sense to include user stories that meet security goals. By adding comprehensive security-based user stories to the backlog, the agile process drives the inclusion of security in each sprint.

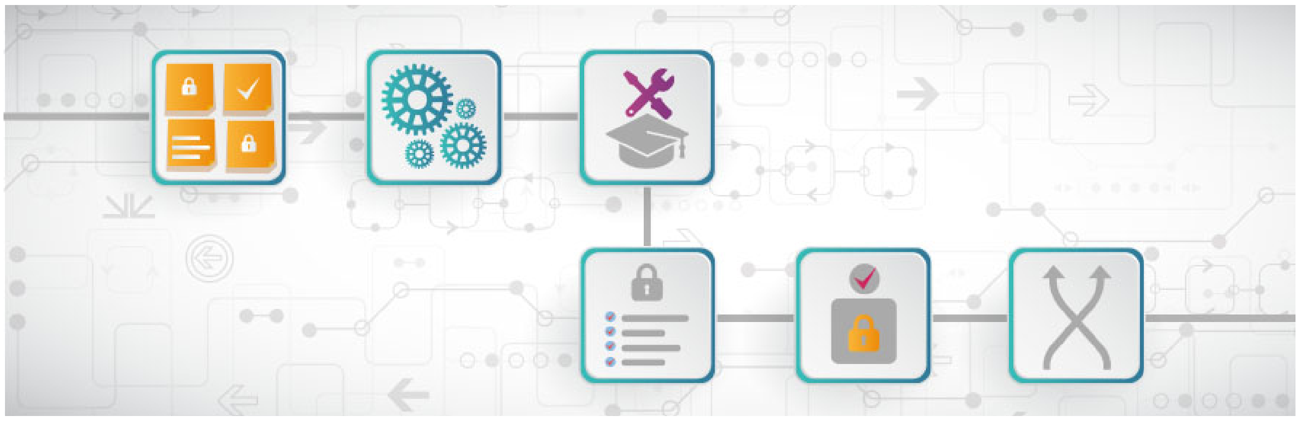
For example, in addition to functional user stories in the form of "As a, I want so that," it is imperative to include user stories that address security-related roles. They could include user stories such as:

- As a hacker, I can input data that is too long and cause unexpected data to be returned
- As a hacker, I can send input that terminates a SQL query and adds additional SQL queries to return unauthorized data
- As an architect, I want to ensure all output is properly encoded

There are many security-related user stories you could add to each sprint. The OWASP site contains an article about evil user stories, and the software assurance nonprofit SAFECode published a paper detailing many more types of security user stories and tasks. These are both great resources to get you started with adding security-centric user stories.

The key takeaway is to realize that just by adding security to user stories, you can make a dramatic impact on the security of your software development process in agile.

Developing Secure Applications with the Agile Methodology



Once you have decided to pursue secure software development in agile, you should get started right away to make it happen. Here are a few of the most important steps to take next:

- Add security-related user stories to your backlog
- Implement automation whenever possible to carry out repetitive tasks (especially testing)
- Educate developers on how to write secure applications
- Develop and publish your own organization's secure best practices
- Empower developers to take responsibility of application security
- Be flexible and continuously build a culture of security

These are rather general goals, but they provide the overall direction for making your software more secure.

Developer education is of critical importance. All software developers must be well-versed in developing secure application software — and given the ability to take responsibility for doing so. Testing should primarily exist for the purpose of functional and security validation.

If you commit to making inclusion of security a priority, agile can provide a framework that results in the most responsive, functional and secure applications your organization can produce.

About TestRail

We build popular software testing tools for QA and development teams. Many of the world's best teams and thousands of testers and developers use our products to build rock-solid software every day. We are proud that TestRail – our web-based test management tool – has become one of the leading tools to help software teams improve their testing efforts.

Gurock Software was founded in 2004 and we now have offices in Frankfurt (our HQ), Dublin, Austin & Houston. Our world-wide distributed team focuses on building and supporting powerful tools with beautiful interfaces to help software teams around the world ship reliable software.

Gurock part of the Idera, Inc. family of testing tools, which includes Ranorex, Kiuwan, Travis CI. Idera, Inc. is the parent company of global B2B software productivity brands whose solutions enable technical users to do more with less, faster. Idera, Inc. brands span three divisions – Database Tools, Developer Tools, and Test Management Tools – with products that are evangelized by millions of community members and more than 50,000 customers worldwide, including some of the world's largest healthcare, financial services, retail, and technology companies.

TestRail