Penetration testing process - Hunting for vulnerabilities

Theodor Adam, Florin Andrei, Larisa Gabudeanu, Victor Rotaru

The penetration testing process, at a high level, is quite straightforward: the organization tests its assets by simulating different attack scenarios as an attacker would, in an effort to identify weaknesses in the way they are set up or developed. The details of how to do this are, however, more complicated, as penetration testing requires a unique set of skills and knowledge that not everybody possesses. Not to mention a rather special frame of mind – one needs to think like an attacker. While it is not our purpose to be extremely technical, we will present the penetration process, how it looks like, what it entails, what are its steps and what are some of the tools available out there to get the job done.

You have probably noticed that we have used the word "assets" when defining the penetration testing process, and not "applications". The reason for this is quite simple: attackers do not necessarily focus only on applications and attacks from nefarious parties might come in different forms and target various components of the organization: applications, network components, physical locations and even people. Some of these attacks are purely technical, targeting perhaps applications or network components, while other represent a combination of tools and techniques leveraging both the technical aspect but also the human factor/behavior.

The pen testing process consists of a few stages with each stage giving input to the next one. However, before going into details of the actual testing process it is best to outline some of the types of tests that are usually employed. The grouping of tests may be done based on knowledge of the tester on the target or, in the case of applications more specifically, based on the types of methods being used to test. Let's take them one by one and understand what the defining characteristics are for each of them.

1. Knowledge of the target

When doing penetration testing, one has a few options of how to do it. Each approach has its benefits and drawbacks, of course. Here are the main types of testing based on knowledge of the targets:

- Black box testing
- Grey box testing
- White box testing

1.1 Black box testing

It may be rather intuitive, but black box testing means the person testing has minimal knowledge of the target details. He/she will have no knowledge of how their targets have been developed, configured and deployed. Nor does he/she have any knowledge of any internal processes or procedures. For example, in the case of a web application testing, there will be no knowledge on how the application was developed, no knowledge of the code, nor how it was deployed in the network. Or, perhaps, in the case of testing physical location, the tester will have no knowledge of the access mechanism employed, what controls are in place, nor of any of the processes supporting the physical access to buildings.

The biggest benefit in black box testing is that it basically simulates an attack when the attacker has no knowledge of the target. It is probably the closest thing we have to a real attack and gives the opportunity to identify vulnerabilities just as an attacker would. You would get the view that they would when targeting the organization. Depending on the skills of the tester, of course. The drawback is that the results might not necessarily reveal all weaknesses.

1.2 Grey box testing

In grey box testing, some information related to the target is revealed to the tester. Partial knowledge of the internal workings of the target are known. To use the previous examples, in the case of a web application, parts of the code (or how it is deployed in the infrastructure) might be revealed to the tester. In the case of a physical access testing, the tester might be shown the details of the physical controls deployed – what is being used, how it is being used. To sum it up, the idea of grey box testing is that when performing the test, you have partial knowledge of the target.

In grey box testing one could potentially reveal more vulnerabilities than in black box testing. As it is usually the case, pen testing endeavors have a limited time frame in which they are conducted – typically 1-2 weeks. This is not enough time for a black box type of approach to identify all weaknesses. However, more information on the target would significantly help concentrate on areas of interest and speed up the process of vulnerability identification.

1.3 White box testing

White box testing is essentially the complete opposite of black box. While in black box testing there is no knowledge of the targets being testing, in white box testing you have complete knowledge of how the target operates, how it has been deployed but also complete knowledge of processes and procedures supporting the operation of it. In web application pen tests, there is full knowledge of the code and of the infrastructure supporting the application. In physical pen-tests there is full knowledge of physical controls used and processes and procedures supporting the physical access control in locations – what systems are used to monitor and grant access, what is the process of obtaining access, etc.

Based on all the information at hand, the tester can develop a much focused attack strategy. Not only that, but they may identify a whole plethora of vulnerabilities that would have otherwise been overlooked. Just to give some more examples, in vulnerability scanning this might mean an authenticated scan is used as opposed to unauthenticated scans used in black box testing, which usually leads to identification of much more vulnerabilities. Or, in case of web app scanning, besides knowing the code, the tester might also be granted access to the application to be able to test all the menus and pages of the application.

While this type of testing does give the pen tester all the information related to their targets and has the potential to reveal much more vulnerabilities, it does expose the organization to the risk of testers concentrating on obvious weaknesses. They would not behave like real attackers and would potentially miss vulnerabilities that an attacker with less knowledge of the system may identify.

These are the three main types of tests that would be encountered and they may be used for any type of testing – application testing, physical access tests or testing various infrastructure devices and controls such as firewalls, email gateways, proxies, VPN concentrators, etc. There are other types of tests as well but these are rather more focused on application penetration test. Nevertheless, we will cover them, as it is important to understand their specifics.

2. Testing approach

There are two major categories of testing in this area and they refer to how the application is being tested from a code analysis perspective – if the code is executed or not. The two categories are:

- Static testing
- Dynamic testing

2.1 Static testing

Static testing, also referred to as SAST, entails testing the application code. This means that the application is not being executed in a runtime environment, but rather testers are looking at how it was developed in an effort to identify any vulnerabilities in the code. This can be done either manually or may be achieved with the help of various tools that help analyze the code. Given the amount of code an application might have, the use of static test tools is usually employed. There are various code analyzers available online that can help with the testing.

2.2 Dynamic testing

Dynamic testing, or DAST, usually means testing the application in a run-time environment. Application is deployed and running and testers try to manipulate input to the application and analyze the responses in an effort to identify vulnerabilities. Dynamic application testing is,

basically, a simulation of an attack on the application. As is the case with SAST, dynamic testing of an application could be done either manually but also by using a variety of tools to generate and manipulate input for applications. We will cover some of these in the next chapter, when we discuss about the testing process.

In most testing scenarios, you will find that a combination of automated and manual testing is used. Usually, testers will use different tools to automate part of the testing and help them identify areas that are more prone to vulnerabilities. Some of these applications already have built in testing procedures that they simply run against the application. Once areas of interest are identified, then the testers hone in on those areas and try to exploit these vulnerabilities by manual means, which could mean writing bits of code/scripts to get the application to respond in a certain way.

3. The process

Fig. 1 – The penetration test process

There are a few well-known and usually employed frameworks or methodologies when developing the actual penetration testing process and how you would want to do things with this respect. These frameworks have been developed by independent organizations or standards bodies, each with its own perspective on things. We enumerate them below, together with the steps that they outline.

OWASP

The OWASP framework addresses mostly web application scanning and security. The Open Web Application Security Project is an open project, maintained by the community and aims at helping organizations to identify vulnerabilities in their applications as well as secure them. It addresses web application and mobile applications security. There are numerous resources that could be used, such as the top 10 web and mobile application vulnerabilities, lists of controls as well as a testing framework – the Application Security Verification Standard.

NIST

The National Institute of Standards and Technology is a great resource when it comes to information security and risk management practices. They have created a breadth of standards and guidelines covering topics such as risk management, security controls and testing methodologies. In particular for this topic, the NIST special publications (SP) 800-53A and 800-115 may be of interest as they touch upon the building of an assessment program (800-53A) but also lay out instructions/technical guideline son how to test.

OSSTM

The Open Source Security Testing Methodology Manual is another framework for penetration testing and testing the security controls. Not only does it provide a framework for testing but also has tools useful for analysis and measurement of the test results – especially when organizations choose to become OSSTMM certified.

PTES

Penetration Testing Execution Standard is another framework to look at when developing your penetration testing process. It consists of 7 main sections covering the whole process from inception – reasoning behind a test – to reporting and presenting results to different stakeholders, including suggestions of what to include in your test reports.

Lockheed Martin' Kills Chain

Originally used as a military concept and more recently adapted to information security, it is intended to describe the structure of an attack[1]. It describes 7 attack phases that begin with reconnaissance and end with the "actions on objectives" phase, in which the attacker carries out his intended goal. The framework may be used as guideline for penetration testing as well as for implementing controls to prevent successful attacks.

Of course, various frameworks and organizations will have their own take on the penetration test process, even though there is a number of common elements to all of the above presented frameworks. Overall, considering also the best practices in the industry, we present our own view on the process, based also on our own experiences.

[1] https://en.wikipedia.org/wiki/Kill_chain

The penetration test process, in our view, has the following major stages: pre-engagement, preparation, scanning, exploitation, reporting and retesting. All these have a sub-set of steps that are to be fulfilled at each stage in the process. Each step produces results that are then fed into the next all the way up to final report preparation.

3.1 Pre-engagement

What to test?

The pre-engagement phase is not necessarily part of the actual testing process, but we consider it as strictly related to it and thus is worth being mentioned. And definitely should be documented. At this initial stage, the security team, together with stakeholders – business owners, risk management, application owners, etc. – need to decide what exactly should be tested. This stage could very well be a one-off effort at the beginning of the year when, ideally, a testing schedule would be created for the whole year. In order to be able to do that, the security team will need input from various teams and decide, together with them, what is to be tested and when. Input for this endeavor might be – risk ratings of systems, importance to the business, scheduled changes or new developments (presumably, we have a calendar for that as well). As it is most always the case, there will not be enough resources to test everything so a decision needs to be taken on what exactly will be tested. Just as an example, final list of assets to undergo a penetration test might be comprised of only high-risk applications, or maybe by externally facing applications and devices (firewalls, WAFs, etc.).

Why and how to test?

The rationale behind the testing will, ideally, be documented. This could take the form of a document signed by all stakeholders, or a testing schedule approved by all involved parties. The reasoning would be to identify and address vulnerabilities within the organization's infrastructure in an effort to improve the overall security posture.

How to test is equally important – will this be done by internal teams? By external third parties? What kind of testing will be required? Will it be a static or dynamic test? All these need to be decided before the actual test begins and, of course, should be documented.

From a tester's perspective, this stage is equally important. Knowing what they will test, when they will test will allow them to properly organize and plan for the testing period. Communication of the aforementioned test plan with the penetration testing team is mandatory. Having knowledge of the testing target, the complexity of the test and desired interval would allow them to properly assign resources and prepare the test.

As already mentioned, the options are either to go through this exercise with an internal team or with an external provider of such services. While things may be easier in the case of the internal team, in the sense that all that would be needed is the communication of the test plan and agreement on the schedule, for the external provider things are a bit more complicated. One will need to go through the vendor selection process, negotiating terms, contract signing, etc. – which, depending on company processes, might take quite a bit of time. The provider will have to ensure they meet all the requirements and are legally permitted to engage in such activities.

It is of paramount importance for the tester to be backed up by contracts and formal agreements before actually starting the test. From our perspective, a test should never be started until all these are completed.

3.2 Preparation

Once the pre-engagement stage is complete, everything has been decided and you have a tester or a testing team available, you're good to go for the actual penetration test. We describe in a bit more detailed manner the actual process and what goes on in each of the stages.

In the preparation phase, the penetration tester will gather as much information as possible regarding the asset being evaluated. Of course, depending on the test type chosen, some information might already be available for them. Nonetheless, they will undergo a few actions the gather information for themselves.

3.3 Footprinting

The initial stage in which information about a computer system is gathered is known as footprinting or reconnaissance. Depending on whether the testing is done on an internal or externally facing app, testers will have different tools at their disposal. Still, most of them may be used in both cases. Testing can make use of simple tools such as ping and trace route commands to gather more information on the assets but also deploy other tools and techniques as well.

Other techniques that may be employed during this phase are:

- Network enumeration
- Port scans
- DNS queries
- WHOIS queries
- Operating system identification
- Google searches – for externally facing applications
- Spidering – in case of web applications – internal or external

All these techniques will give the tester more information on the asset under testing. Port scans may reveal certain open ports that could be exploited – either because of vulnerable services that have known vulnerabilities and, possibly, even exploits or because there are services/protocols in place that are just plain simply not secure – or example Telnet. The same goes for Operation system identification – you might think it is not much, but actually knowing the OS running, coupled with other results from footprinting, could give you a very detailed "map" of the vulnerabilities that might be present on the host. Not to mention if the OS is an older one. Spidering, or the use of a web crawler, can reveal pages on a web application that are not usually accessed by typical users but could be useful in building up an attack against the website – thinking here of possible error pages that do not filter out information and give much more details than they should (such as what technologies are used in the backend), or just giving our information that just should not be public.

3.4 Scanning

In the scanning phase, you take it one step further and perform vulnerability scans on the identified systems. You use the information gathered in the reconnaissance phase, such as IP addresses, hostnames, etc. and launch scans against those targets. This is done automatically using vulnerability scanners, pointing them towards hosts, servers, network equipment, etc., depending on what the target is. This is also true for web applications. However, to some extent testing for vulnerabilities in web applications may be done manually as well – testing input fields in web pages, for example, by trying to pass input to the application that will determine it to process data in an unwanted way.

There are multiple types of scanners available online:

- Vulnerability scanners – these are usually dedicated tools that scan your target with the purpose of identifying known vulnerabilities present on it. They present their results in relation to the CVE identifier for the particular vulnerabilities they identify. These are complex tools that can actually do a plethora of tasks, such as scanning ports, identifying Operating systems and also test for known vulnerabilities.
- Port scanners – a somewhat simpler version of the vulnerability scanner, the port scanner does what its name implies – scans 1 or more ports to identify if these are open.
- Fuzzing – although not practically a scanner type of application, we consider fuzzing as part of the Scanning phase. Fuzzing is a technique of providing incorrect or unexpected input to an application/ a particular field and observe the responses the application returns to unexpected inputs. Usually, you are looking for error messages, crashes. Fuzzing may be done with dedicated tools, specially built for this or by manual

means. They are typically used to detect potential buffer overflows, memory leaks, undefined behaviors, errors, etc. – all of which pose a security risk for the applications.

3.5 Exploitation

The Scanning phase will yield various result to the tester – from operating system used to potential known vulnerabilities present, open ports and fields with issues and errors. Next phase of the process is to gather all this information and try to exploit the (potential) vulnerabilities identified. This may be done by using known attack techniques that have been proven to work in a particular case or the penetration testers can develop their own approach to exploitation.

Without going into the technical aspects, these are some of the techniques that might be employed:

- code execution in applications

Fields that have been identified as having issues will be further explored. The penetration tester will generate pieces of code and will pass them to the application to run them. If the code was poorly written, then this additional chunk of code might get executed by the application, thus performing unwanted operations. Running additional code may be very dangerous, especially if the test is done in production (which we do not recommend), so extreme care must be taken when choosing what to run. For purpose of proof of concept, a function that would display something on the screen rather than executing operations on the database would be enough to showcase the presence of vulnerabilities.

- malware delivery

The penetration tester might choose to deliver a malware either to people or to the system. Delivering malware to people might be done through phishing emails that could be distributed to all employees. This would test also the level of awareness within the organization. Another option of delivering malware would be through upload fields in applications. This would test if the input fields have been limited to specific expected types of files and also if the uploaded files may be run in the back. Basically, the test is to see if they could pass on malicious content to the organization and if that gets run on not. Of course, when we say “deliver malware” we are not suggesting to deliver actual malware that might wreak havoc in the organization. Rather, the penetration tester would make use of known test files, such as the EICAR malware test files, or craft their own that would perform a certain benign activity just to prove the existence of a vulnerability.

3.6 Privilege escalation

This step is the peak of the penetration testing endeavor. Privilege escalation is actually "the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user."[2] The purpose of the penetration test process is to identify and prove the existence of vulnerabilities and obtain privileged access to systems. While the proof of the existence of vulnerabilities is definitely useful, obtaining privileged access to systems is the king for a penetration testing endeavor as it showcases that it is actually possible for an ill intended party to do so (at least from my point of view).

The result of this stage is either an application that has more privileges than intended by the developers or a user with admin access that could perform unauthorized actions on systems/applications.

Privilege escalation could be split in two types/variants:

- Vertical privilege escalation – this would be the actual "privilege escalation". This involves a user/application with a lower level of access obtaining a higher level of access that was not intended for them. Such an example would be a normal user gaining administrative access on his workstation or a normal user gaining admin access on the server.
- Horizontal privilege escalation – this entails the access of content and functions that is normally reserved for other users of the same level. One such example would be a user accessing another user's account in a web application.

An important note should be made here. Should a penetration tester find a really serious vulnerability and through exploiting it would obtain access to important accounts (such as domain admin accounts, admin accounts in applications, etc.), this should be reported immediately to the organization, rather than wait to report it through the final penetration test report (which we will discuss shortly). The existence of such vulnerabilities with potential catastrophic outcomes for the organization need to be addressed immediately. Of course, the final report may contain details about this finding as well, but fixing the problem should benefit from immediate attention and remediation. The organization can address this finding while the penetration tester moves on to testing other areas.

[2] Privilege Escalation Wikipedia article - https://en.wikipedia.org/wiki/Privilege_escalation

3.7 Report

Each penetration testing endeavor should end with a detailed report. The report, ideally, would be split into two parts:

- Management summary
- Detailed technical analysis

Management summary

The management summary part should a 1 – 2 pages maximum report outlining in general, the findings, presenting the overall risk rating and a breakdown of vulnerabilities and their associated risk ratings together with a general opinion on the security stance of the organization based on the scope of testing and findings. Management is interested on the impact to the business and dollar figures, so leave the technical details out of this part.

Detailed technical analysis

This part of the report is presenting in depth the whole testing process. For each vulnerability identified, it should present the details of what tests were performed as well as the steps performed to identify it. Moreover, print screens showcasing the finding should also be included. Give as much details as possible to that the technical teams can actually reproduce your findings for confirmation. Besides the technical details, each vulnerability should also have its associated risk rating mentioned. Ideally, each vulnerability would also have recommendations for remediation made by the penetration tester.

Determining the risk level for each vulnerability might be quite difficult, especially if the penetration tester is a third party and might not be accustomed with the risk framework and risk rating system used internally. However, an initial estimation based on known best practices can be done. The penetration tester may make use of several resources when rating the vulnerabilities identified:

- the CVE scoring system
- OWASP Top Ten web application vulnerabilities
- MITRE ATT&CK framework
- Own experience

3.8 Recommendations

As already briefly mentioned, the report should (or better said must) contain recommendations made by the penetration tester or team on how to remediate each of the identified vulnerabilities. Recommendations may span from the more general ones, such as “patch the

operating system" or "apply patch x for the application" to more specific items such as details on how the configuration should be modified or what areas of the applications should be modified such that the vulnerability is addressed. Recommendations for remediation, from our perspective, should be as specific as possible. This will depend, of course, on the knowledge and expertise of the tester/testing team.

4. When to test

Up to this point we have discussed the why and the how of the penetration test process. This section will cover the when. It is important to note that the penetration test process is a cyclical one. The applications evolve over time, technology used changes at a rapid pace and new vulnerabilities are discovered every day. We add to this the ever-evolving attack tools, tactics and techniques the bad guys use and it becomes clear that penetration testing is not just a one-off effort, but rather is cyclical and repetitive. Applications and the organization's infrastructure should undergo initial tests, retests as well as constant testing at specific intervals.

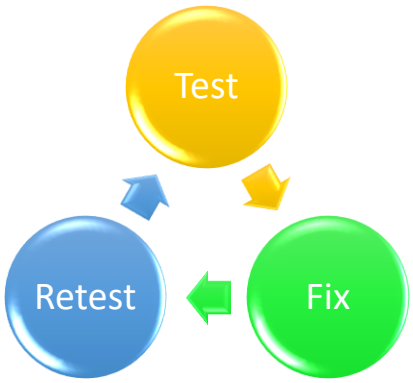

Fig. 2 – Penetration test process

4.1 Initial testing

The organization wants to implement a penetration test process. Where to start off? To begin with, a decision needs to be taken on what to test. Moreover, the retesting interval should also be defined. We have described this in more depth here https://cert.ro/vezi/document/pentesting-and-redteaming but suffice it to say that testing and prioritization of what needs to be tested may be done based on risk scores of applications and elements of its infrastructure. The testing process, prioritization and timeline can be discussed by you with the organization, based on your experience.

For initial testing, there are two main areas of focus:

- Penetration tests for existing applications
- Penetration tests for new applications

Penetration tests for existing applications

In the case of existing applications, we can also further split this into two types of testing:

- Test the applications itself
- Test the application in operational context

Testing the application itself is pretty much straightforward. Based on the risk rating (or any other method of prioritization the organization chooses), the organization selects the application that needs to be tested. The organization selects who will be testing, the type of test that the organization wants to perform and the penetration test process will be the one described in this article.

For existing applications, however, there is another type of testing that should be performed – the application in operational context. This can very well be part of the same testing endeavor but must not be overlooked. The tests should cover the application's uses of various departments, any integrations it may have with other applications, essentially covering all the data flows to and especially from the application. Is the application sending data to another application (integration) or are there any processes that heavily rely on its outputs? These would also need to be tests to assess the real impact on the organization. Testing just the application itself would reveal important information, but it's just one part of the problem.

Penetration tests for new applications

In the case of new applications, the rule of thumb is always test before going into production with the application. The process usually follows the same steps outlined in this article: selection of tester/testing team, type of test, etc. The penetration test will need to happen after development has been finished and before the application goes into production to ensure that potential vulnerabilities are identified and addressed prior to it becoming operational.

4.2 Retesting

One important aspect of the whole process is the retesting of vulnerabilities. As previously mentioned, new applications need to have vulnerabilities addressed before going live. The process

is simple: penetration tester issues initial report outlining the findings, risk ratings and recommendations, as already covered. The report is shared with the development team and internal teams. They will need to address the vulnerabilities before declaring the application production-ready.

Once vulnerabilities are fixed/addressed, a retest of the initial vulnerability presented in the report will need to be done to validate it has been addressed. This process of develop and retest, ideally, should be done until there are no vulnerabilities left open – or at least the ones that remain open are low in which case the owner of the application may choose to accept those temporarily. At the very least, critical, high and medium vulnerabilities identified should be addressed.

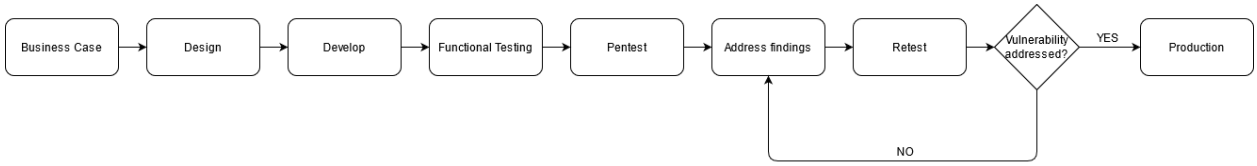

Fig.3 – Retesting of vulnerabilities

In the case of existing applications, which are already in use, the resting of findings usually takes the same route – although there is a significant difference here: the application is already in use and in production, so there is no dependency in the retest to validate the fixing of vulnerabilities before it being available to users. Once the penetration test is performed and findings presented, the addressing of findings will usually take the form of an action list/projects list for the internal teams (developers, infrastructure, network, etc.). They will work on creating fixes/applying patches, etc. However, once each point is addressed, a retest will be needed to confirm that the initial vulnerability is no longer valid. While in the case of new applications fixing the vulnerabilities would be done immediately as there is also pressure to promote the app to production, for existing applications the plan may span multiple months, depending on existing operational issues and tasks, resources available and importance of findings. Given the constraints, the list of findings should be prioritized and the most critical vulnerabilities addressed first and as soon as possible.

5. Where to test

We have covered the types of testing, the details of the penetration test process as well as why and when to test. In the end, there is one important aspect to discuss and that is the testing environment – or where you test.

There is one rule: ideally, because of the potential disruptive nature of a penetration test engagement, these should never take place in the production environment as daily operations may be impacted which, in turn, may translate to loss of money or reputation for the organization (at the very least). Testing should be conducted in a test environment that is configured to replicate the live environment as much as possible. Ideally it would be a 1-to-1 copy of the production environment but since this might prove to be extremely costly to set up and maintain, a close enough replica that would allow you to deploy the application being developed (if it is new) or that would already host the existing applications and simulate all connections it may have will work. The important thing is that you have a test environment where all functional and non-functional tests, including penetration testing, would be performed.

Just to mention a few of the characteristics for test environments:

- Replicates production environment – including operating systems used, development frameworks, databases
- Is isolated from production environment
- Typically, should not contain live, production data

Conclusions

This article describes what the penetration process entails, its steps and some of the tools available for the specific activities, from the angle of the person or organization that is responsible for execution. The knowledge about the target makes a difference in choosing the right approach for the testing. The reason for the pen testing mission has to be understood and the actual process must be properly prepared. Once executed, writing the reporting (including analyzing the findings and making recommendations) needs to deliver actionable information, addressing the vulnerabilities found, offering iterative retesting after remediation. The outcome should be an improved state of security.