Penetration testing and Red Teaming – Blueprints for your fortress

Theodor Adam, Florin Andrei, Larisa Gabudeanu, Victor Rotaru

In the first section of this article, we are discussing the stages an organization should go through in order to prepare for a penetration testing activity, together with practical approaches to be taken, especially in terms of prioritizing the applications to be tested based on risk assessment and mechanisms for addressing change management in terms of recurrence of testing.

In the second section, we have included the main differences between red teaming and penetration testing in order for the organizations to identify the best suited actions to be taken at a certain point in time or on specific IT assets. Further, we have outlined the main principles to be taken into account in red teaming exercise and, in the third section, we have outlined the main tools to be used in both red teaming and penetration testing exercises.

1. Penetration testing – the organization's perspective

In this section, we are starting with the description of the penetration testing concept with emphasis on the situations in which it is useful. We continue with making recommendations on the establishment of roles and responsibilities in your organization for this type of exercises and outlining the strategy for prioritizing IT assets for which penetration testing to be performed based on risk assessment results and CIA (confidentiality, integrity, availability) rating.

Following the risk assessment and CIA rating exercise, the organization can establish scheduling of penetration testing and create a mechanism for addressing change management scenarios. Practical recommendations in this respect are also included in this section.

1.1 What is penetration testing?

In this article we will be focusing on one of the important processes that are at our disposal to help secure our applications and infrastructure ensuring smooth and secure operations and, hopefully, peace of mind. With the growing adoption of web and web technologies there has been an increasing discussion around this topic and, sure enough, an increasing demand in professionals with a working knowledge of the process.

So, what is, exactly, penetration testing? For those of you more familiarized with the broader IT concepts, you could think of it as the functional testing of security features of an application,

infrastructure component, etc. Just as with the typical application being developed or infrastructure component being deployed when in the design phase the project team gathers the functional requirements (i.e. what the application is intended for and what are its functions), so is true for the security requirements of the very same new developments. The project team needs to identify what are, at least at a high level, the security requirements that the new application needs to fulfill.

At the end of the development/implementation, these will need to be tested for, to ensure they have been implemented and working correctly. Penetration testing is the process that ensures security measures have been implemented according to specifications and data and your whole network, ultimately, are secured in the face of malicious activity.

To note here that this is not a one-off process. One cannot assume that if done once and passed with no notable issues (as there is no application 100% secured just as there is no such thing as 0 risk) the application will be as secure in 3 years' time. The threat landscape is constantly evolving, new techniques of attack are constantly developed, not to mention that new vulnerabilities are discovered every day in all applications, frameworks and even in security products.

Thus, in order to maintain a sound level of security for your data, testing will need to be undertaken regularly. Penetration testing should be a continuous process in which not only new but also existing applications are included. Penetration testing should follow the application throughout its life-cycle.

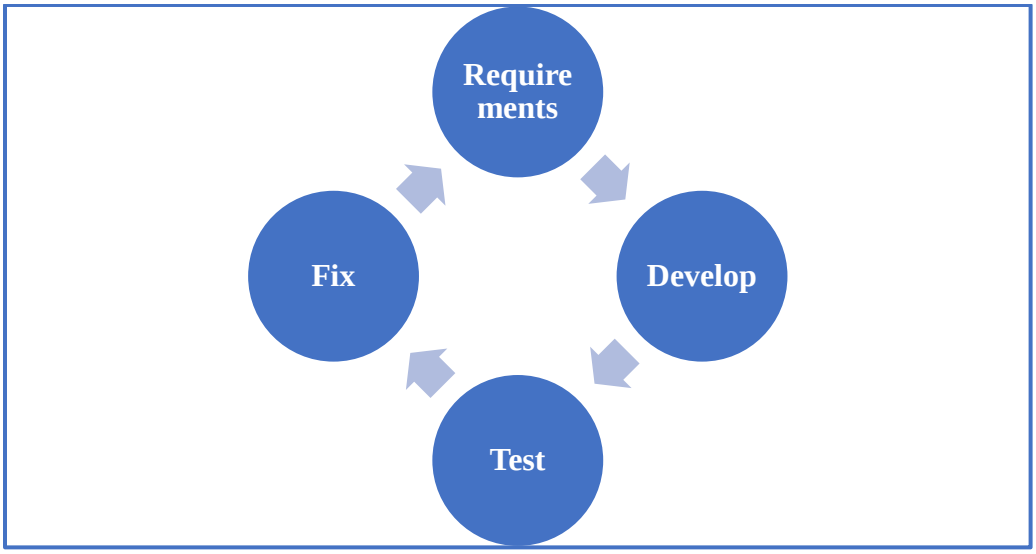

Figure 1 – Penetration Test process – high level overview

There is quite a lot of information already in these few lines already. What applications to test, what kind of testing and when to test them are, probably, just a few of the questions that pop-up. We will try in the following sections to address each of them, along with many others and, hopefully, by the end of the article our readers would have at least a baseline understanding of the concept and how it could be implemented within an organization and embedded within its processes.

It needs to be noted that, even though we refer to penetration testing, a broader description of the whole process of ensuring application security is needed – from requirements generation to the actual testing. We see these an interlinked and consider they need to be treated together in a single framework, rather than discussing just about the actual process of testing – which, in itself, is rather a technical discussion and is not the purpose of this article, although we will try to cover that as well, at least at a high level.

1.2 Roles and Responsibilities

Just as with pretty much anything else, implementing a penetration testing process should start off with the definition of roles and responsibilities. These, most probably, already exist in some form in the organization, as they would be part of a much broader implementation of an IT management framework, Information Security Management Framework or IT Risk management framework. The readers are probably familiarized with them. Nevertheless, a review of these roles is beneficial.

CISO – overall responsible (and accountable) with defining the security strategy for the organization. Depending on the size of your organization, the CISO might have multiple teams under his command.

Security Manager – or, depending on size, Security Managers. The Security Manager is responsible with implementation of the security strategy – developing and implementing projects aligned with the goals as set forth in the security strategy. He/She might lead one or multiple teams of security professionals with different areas of expertise.

Application Owner – The person from the organization that is accountable for the use and maintenance as well as security of a particular asset (application, server, database, etc.).

Penetration Test engineer – technical person, part of the internal security team or a vendor's team, with knowledge and skills to perform security testing for various assets.

It is very important that these roles are defined within the organization. Moreover, the need for the application owner in this context cannot be stressed enough. The application owner is that

person from the business that holds accountability of the application good functioning as well as security.

As with all other aspects of security, getting the buy-in from business is crucial in testing but even more so in remediation of found vulnerabilities as most of the time this will mean extra hours/man days – either internal or external – for development/configuration and retesting. Which, of course, translates into dollars – or whatever your currency is. The owner needs to be educated on security matters and understand the security gaps as well as the risks they are faced with, risks associated with not addressing these gaps, and support the remediation efforts.

A risk-based approach is needed for the overall penetration testing program as you will not always choose to address all identified issues nor will you test all applications with same frequency. But more on that later in the article.

1.3 Defining the strategy

Probably, at this point it should be mentioned that having a risk management framework in place within the organization would help the effort of implementing a penetration testing program. It is not absolutely mandatory, but it would definitely help as you could map the pen test process to the framework and use the same risk rating system as with the rest of risk assessments being performed. This way, it would make much more sense and be more relevant to the business especially when discussing about remediation efforts and costs versus the risks.

1.4 Risk Management – a quick overview

There are plenty of risk management frameworks out there to choose from and implement. Just to note a few, we have the NIST SP 800-37 - Risk Management Framework, ISO27005 Risk Management Framework or the ISACA Risk IT framework. Any of these would make a good choice for implementing risk management into your organization.

What is risk?

Risk is defined, in simple terms, as the possibility of something bad happening. Even though there are definitions out there that say that the existence of risk involves opportunities as well, we will focus on the former meaning – possibility of something bad happening. As such, in the IT risk world, risk is defined as below:

Risk = P x I where

- P = probability of something happening
- I = impact of that particular event.

If we were to have a graphical representation of the risk, it would look something like this:

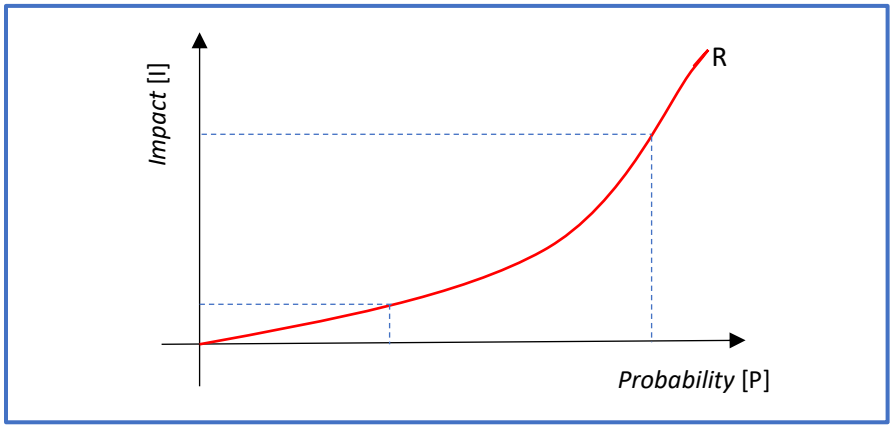

Figure 2 – Definition of risk

When we are discussing IT risks, we are referring to any event that could affect one or more of the three main characteristics of any IT system from a security perspective: confidentiality, integrity and availability. Just to remind ourselves, confidentiality is the property of something being secret, integrity refers to correct and unaltered data while availability represents a system's ability to operate uninterrupted, thus offering uninterrupted access to data.

Although it is not the purpose of this article to go into the details of risk management, nor to present any framework in detail, a quick review of general concepts is useful both for those more acquainted with it and for those rather new to the field. If we were to consult the frameworks mentioned a few paragraphs above, overall, the risk management framework has 4 main components: risk identification, risk response, addressing the risk and monitoring the risk.

Figure 3 – Risk management process

For the purpose of identifying and evaluating the risk levels, usually, a form of risk definition exists at organizational level. Probably, the most prevalent one is the use of a risk matrix that will define the organization's view on risk. It could look like the one below:

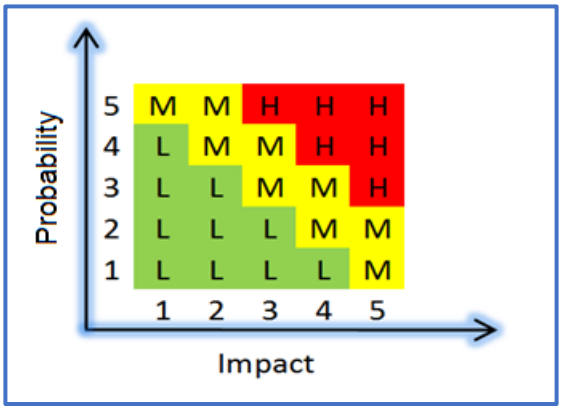

Figure 4 – Risk Matrix

1.5 Rating the applications

To summarize, until this point, we have covered the following – what is penetration testing, what are some of the roles and responsibilities within the organization that could be involved in the process as well as a quick overview of the risk management framework to use to identify, evaluate and respond to risk.

Now it comes down to what exactly to test. Not all applications have the same function, process the same kind of data nor are they accessible to the same audience. Some applications might be internal, some external. Some of them might be making use of confidential data, PII (personally identifiable data) or health data while others might contain just some marketing data or even data that would be readily available publicly as well. This complicates things and might generate a lot of confusion as to what to test exactly, when to test or how often to do it. Here, the rating of applications comes into play.

Rating the applications could very well be done as part of the overall IT risk management program – applications could be rated, for example, in terms of:

- Confidentiality – meaning what is the level of secrecy of that data and, subsequently, what would be the impact to the business if data stored in those applications would be lost
- Integrity – meaning what would be the impact to the business if data is stored would be altered in any way
- Availability – what are the availability levels of these applications and, of course, what would be the impact if connection to these applications would be lost

Rating the confidentiality, integrity and availability levels of an application could be done using simple notations indicating the level of importance of that data or application. For example, using levels from 1 to 4 or from 1 to 10 if you want to be more granular, to indicate importance of the application when it comes to any of the three characteristics of the C-I-A triad. These levels could very well match the data classification scheme at organizational level, as defined within the risk framework, or you could use a stand-alone rating system specifically for this purpose.

However, making use of the company wide data classification scheme has its advantages as everyone will understand their meaning when they see the ratings. An example of a data classification scheme that takes into consideration the confidentiality, integrity and availability of data could look like this:

Area	Classification Level	Classification	Data property

Confidentiality	C-1	Public	Public data
	C-2	Internal	Data for internal use. Some damage if data lost/made public
	C-3	Confidential	Important internal data. Significant damage if lost/made public
	C-4	Secret	Data that, if made public, would significantly affect the business

Area	Classification Level	Classification	Data property
Integrity	I-1	Public	Data is publicly accessible. Alteration would not bring any damage or impact would be very limited
	I-2	Internal	Internal data. Alteration might have limited impact on people/departments using it.
	I-3	Restricted	Important data. Alteration of this data would cause damage to organization – either financial, reputational.
	I-4	Critical	Alteration of this data would bring serious damage to the organization.

Area	Classification Level	Classification	Data property
Availability	A-1	Non-critical	Low or very limited impact should the asset be unavailable
	A-2	Sensitive	Unavailability of this data would cause some impact to the organization
	A-3	Vital	Loss of availability would lead to significant damage for the organization

	A-4	Critical	Serious adverse effects for the organization should the data/asset be unavailable.

Applying this rating/classification system to all your applications through the process of impact assessment would lead to having a full view of what data you have and process as well as a better understanding of the overall risk the organization is exposed to (e.g. running a plethora of C4 applications makes it clear that the organization is handling critical data and should it be exposed or lost this would have significant adverse effects on the business. Compliance or legal implications, bad publicity or down right loss of business are some of the possible outcomes). Ideally, of course, the application ratings would be noted in a CMDB and kept track of.

As it is usually the case, as business evolves so will your applications. A recurrent review cycle is, thus, needed. A re-evaluation of the rating is deemed necessary once every 2-3 years. A reasonable interval needs to be chosen so that the re-evaluation interval is long enough that it does not put a significant burden on your staff while yielding the same results, which is not productive, but short enough to timely capture any significant changes. Usually, a re-evaluation once every 2-3 years would be an optimum interval for most organizations.

We do acknowledge that some organizations, depending on their nature or size might not have a fully developed risk management framework or a sound data classification scheme implemented across the board. In this case, one alternative option of rating your applications but also deciding how and when to test them is the OWASP ASVS scoring system. OWASP stands to Open Web Application Security Project. As per their website, OWASP "is a nonprofit foundation that works to improve the security of software. Through community-led open source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web."[1] The OWASP foundation has a range of projects covering application security: OWASP Top 10 – which covers the top web application security risks, Mobile and Web application testing security guides just to name a few.

The OWASP ASVS (Application Security Verification Standard) "provides a basis for testing web application technical security controls and also provides developers with a list of requirements for secure development."[2]

The OWASP ASVS is a standard intended to help developers build secure applications but also to "allow security service vendors, security tools vendors and consumers to align their requirements and offerings." The ASVS has multiple uses and may be looked at as a guide for

[1] https://owasp.org/
[2] https://owasp.org/www-project-application-security-verification-standard/

security architecture, replacement for off-the-shelf secure coding checklists but also as a guide for testing. We will not go into the full details of the standard, but we do encourage a deeper inspection of the whole standard. For the purpose of our topic, we will mention that the ASVS provides also for a rating system that may be applied for your applications. There are three security verification levels defined within the standard, each with varying degrees of complexity:

- ASVS Level 1 is for low assurance levels, and is completely penetration testable
- ASVS Level 2 is for applications that contain sensitive data, which requires protection and is the recommended level for most apps
- ASVS Level 3 is for the most critical applications - applications that perform high value transactions, contain sensitive medical data, or any application that requires the highest level of trust.

The standard provides a list of requirements but also gives guidance on testing and verification.

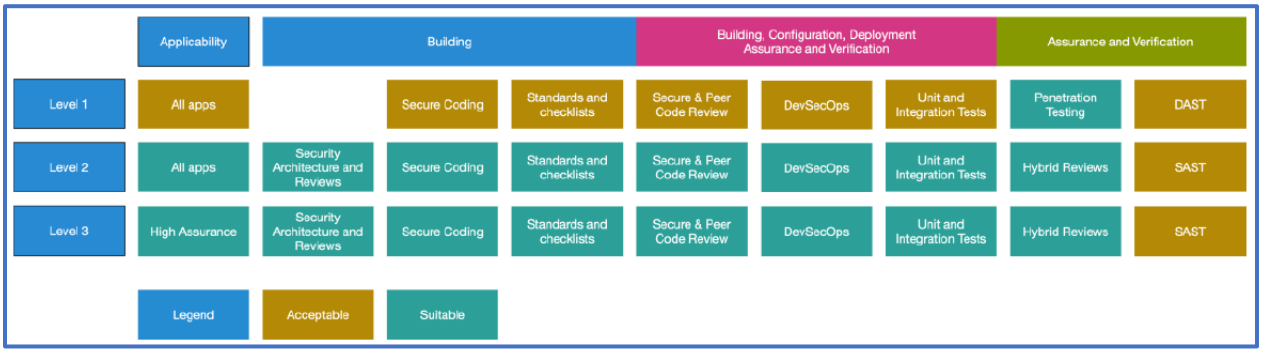

Figure 5 – OWASP ASVS Levels

Using the same approach, the applications may be given a rating/a level, based on the data they process and level of risk they pose to the organization. The ASVS rating system covers a broader set of requirements and controls but at a very minimum, one could use it as guidance for rating the applications.

1.6 Defining a testing schedule based on application rating

Having the applications evaluated and rated, regardless of the method used, is an important step in establishing the penetration testing program within your organization. Next, we need to identify how and when each of the applications should be tested. Some applications are more

important than others. Some process personal client data, while other applications may contain only internal data or marketing data. Not all have the same level of importance to the business and, as consequence, losing any of the C-I-A triad components for them will not have the same impact on the business.

Considering all the applications have been rated, either using the impact assessment method or ASVS scoring system presented or any other method, we can then just map these to a testing interval that is reasonable for the business and, just as in the case of the assessment interval, will be long enough to ensure there is no overburden on resources and there is no unnecessary testing being performed, but long enough to capture potential changes. It is also to be noted that new attack methods are continuously identified and published, as well as vulnerabilities for underlying operating systems, development frameworks, databases, etc. Given the complex nature of IT systems and the speed at which new vulnerabilities are identified the testing interval should not be too big. Compared with the assessment interval, where we have said that perhaps a reassessment once every 2-3 years would be ok to capture any significant/major changes, for the actual testing we should be considering 1-2 years, depending on the application, the data it processes and its rating.

Let's take the example of the impact assessment and scoring system we have presented. We will take each of the four possible levels (C1-C4) and map these to a testing interval. While creating the testing schedule, we need to take into consideration also what type of testing should be done for each level. Taking all of this into account, a potential result could look something like this:

C1 – application contains public data or data that is readily available from other sources as well. Losing this data would not have any impact on the business. For these applications, a testing interval once every 2-3 years could be applied. It should, however, be no longer than the assessment interval. By matching testing with the assessment interval, you can ensure you capture any significant changes and test them. For example, if an application changes rating at next assessment and becomes a C2, then you can also test it immediately in the next period as it would have been tested any way.

C2 – these applications might contain internal company data or any other data that is not readily available to the public. Impact is higher in case of security breaches, compromise and lost data. For these applications a tighter testing interval is needed. Perhaps one could consider 1-2 years for the testing cycle. In terms of testing methods, while for C1 applications you would go with just a dynamic black box testing, for C2 the option of dynamic testing and perhaps grey box or white box testing, where the testers have knowledge of the application design and can perform authenticated tests would be a choice. By performing authenticated tests and having knowledge of

the architecture, more vulnerabilities could be identified that would not have been visible in the case of a black box testing.

C3 – applications rated as C3 contain internal sensitive information, perhaps also some combination of personal data, financial data, etc. Testing these applications should be at least on an annual basis. At this point we would also introduce the concept of static testing, or code review. Static testing may be done by use of automated code review tools.

C4 – the most critical applications for your business in terms of data they process. Depending on the area of activity, they may contain sensitive personal data, health data or financial data. They may contain sensitive business logic or trade secrets. Needless to say, should these applications be compromised, the impact to the business would be major. To match their importance, the interval for testing should be, perhaps, once every 6 months but no more than 1 year. All testing methods should be used. White box testing, dynamic testing as well as static testing – both automated and manual reviews, where possible.

The end result of how a testing schedule would look could be as shown in the below table:

Application rating	Assessment interval	Testing interval	Testing method
C1	2-3 years	2-3 years	Dynamic, black box
C2	2-3 years	1-2 years	Dynamic, white box
C3	2-3 years	1 year	Dynamic, Static
C4	2-3 years	6 months/1 year	All test methods

1.7 Special cases

Handling change management:

Up until now we have discussed assessing application importance and impact and testing based on these ratings. The testing schedule has been defined considering that the applications remain the same.

However, the business environment is in a constant change and, as a result, so are the applications. Throughout their lifetime, applications evolve: new functionalities are added, some are merged together in one bigger application or they may just collect additional data as part of the same type of processing – due to business needs or as demanded by regulatory requirements, etc. The important thing is that applications do change and this is an aspect that needs to be taken into consideration as well when evaluating their impact and establishing the testing interval.

Your framework would need to account for major changes as well. The change management process would need to account for major changes to applications and trigger either a reassessment of the application or a test, or even both before new functionalities are promoted into production. New functionalities such as processing additional personal data or adding a module for health data should definitely be evaluated and the rating of the application establish accordingly. Testing would need to be done to ensure that the new features do not have significant vulnerabilities that could be exploited.

Adjusting the testing schedule – balancing business needs and security requirements:

There are a lot of moving parts involved in the testing schedule. It is important that the testing schedule be aligned with the changes/new projects schedule. The teams overseeing the security testing schedule should be in constant collaboration with the teams overseeing new developments. A discussion on coming testing and new projects/functionalities would ideally take place once every 3 months for alignment. This way, the security team can reprioritize the testing based on the IT planning.

For example, one application is due to be tested in 2 months' time. But the development schedules estimates that significant new functionality to the application will be added and estimated to go in production in 3-4 months' time. The security team can use this information to postpone the testing of the application for 2 months to ensure they cover the new functionality as well. In this case, even though the application test would be delayed for 2 months, and perhaps surpass the testing interval established, it would cover more ground and would be more relevant for the new version of the application.

There is no established formula on how and when to do the testing. As also shown in the example above, testing may be delayed with good reasoning, or, in the same manner, done faster. Sound reasoning and constant collaboration between the security team and rest of business is, however, essential. As is almost always the case, there will be resource constraints, priorities will change, etc. As long as communication channels are open with the security team and decisions are governed by a risk-based approach that will ensure both efficient use of resources and security of your data and applications, the business has only to gain.

2. Red Teaming – methods and practices – the organization's perspective

Now let's also describe the concept of Red Team and review some of practices that should be taken into consideration during a red team engagement. The below outlines first the distinction between penetration testing and red teaming exercises. Further, we outline the main principles to be considered for the red teaming exercise from the perspective of the red team and from that of

the organization, together with the main technical tools and techniques to be considered in this case.

2.1 Differences between penetration testing and red teaming

The Red Team is, essentially, a group of people/a team that is brought in to assess the level of your security posture or the effectiveness of your security controls. The red team, basically, simulates an attack on your systems, in a controlled fashion (preferably), in order to identify potential gaps.

Although you might be tempted to say that penetration testing and red teaming are the same (they are often used interchangeably), there is a difference between the two.

Scope: Penetration testing is performed in order to identify potential cybersecurity vulnerabilities – application layer flaws, network or system level control gaps or even physical security vulnerabilities. It's looking at your environment through the eyes of an attacker. The pen test is planned, usually known about, and the target is agreed upon prior to the assessment. Red Teaming, on the other hand, while at a high level has the same aim, identifying and exploiting vulnerabilities in an effort to gain access to systems, it differs in approach. Red team operations do not entail agreeing necessarily on a target – their job is to assess the level of security and try to obtain access to systems in any way possible. The approach entails, of course, "attacking" the organization from multiple angles, simultaneously.

Team members: Further, the Red Teaming effort usually entails more people, time and resources as compared with a pen test which is usually done by 1-2 people in a limited time-frame.

Tools and techniques: Of course, the red team will be making use, probably, of the same tools and techniques you would see used by a pen tester. The difference lies mainly in the time, people and resources available, not knowing the actual target and, in general, in the approach.

One interesting way I have found describing this is the following: "pen testers are pirates — ready to rampage and pillage wherever and whenever they can. In this analogy, red teamers would be more like ninjas, stealthily planning multi-faceted, controlled, focused attacks"[3].

2.2 Main principles in red teaming

In terms of desirable practices for the Red Teaming effort, without getting into the technical aspects of testing, there are a few worth mentioning:

[3] https://www.redteamsecure.com/blog/penetration-testing-vs-red-teaming

a) Plan in detail

The team should take the time to detail the approach as much as possible, outline the approach, define roles in the team and create a general attack plan. Generally, best practices are to have common discussions with the red team and relevant persons from the organization that are aware of the assessment and take into consideration their valuable input.

The discussions at this stage, of course, will not go into details about the red team strategy and targets, but focus on the perimeter of the organization and specific limitations to the scope of the red teaming exercise.

b) The red team should document its approach

The red team documents the target, the roles assigned to the team members and their general approach (basically, document the results of the planning phase). Even though the target might not be specifically defined, it is a good idea to document the approach and the discussions with the organization about the attack methods used, the depth of the test and how far the red team would be allowed to go (e.g. prove that access may be obtained to a database versus actually extract/delete data from that database).

This documentation part is useful for the lessons learnt part, in order for the organization to take away areas and manner of improving its security.

c) Diversify

The red team should consider using a variety of tools and techniques in their assessment and not limit themselves to just one attack method or tool. Use a variety of vulnerability scanners, pentesting tools, web application attacks, frameworks, social engineering techniques, etc. Basically, the red team tries as much as possible to simulate a real-life attack. An attacker would usually exploit the same vulnerability in multiple ways and would try out as many techniques as possible to gain access. The same needs to happen with the red team.

This is aimed at testing the reaction of the blue team of the organization under usual attack circumstances.

d) Document findings

At the end of the engagement, the red team should document all findings in a detailed report.

What are the findings?

What were the vulnerabilities identified and how have they been exploited?

What is the risk level for the organization, given the deep understating the red team has gained of the organization and their environment?

What are some of the actions they need to take to remediate the findings and address the risks?

The red team's report should contain all of this information so that the organization can take an educated decision on the next steps.

This step is closely tied to the continuous feedback given by the red team to the organization throughout the red teaming exercise. On the side of the organization, a similar report is usually prepared to identify the improvements that can be made in terms of detection, assessment and response to potential attacks.

3. Tools and platforms to use

Now let's look at some of the tools that might be used during a pentest or a red team engagement. Generally, there is a combination of automated tools and expert analysis in each of these cases.

a) Vulnerability scanners

These are tools that scan your systems (network devices, servers, workstations, dedicated appliances) and provide you with details related to vulnerabilities present on these systems. They often list their findings/the vulnerabilities using CVE identifies that provide information on known weaknesses.

b) Proxies

These are tools that can be installed locally and intercept traffic between the host and destination. These may be used while communicating with web applications to modify responses to server requests or alter the inputs or requests from client to server. They can be very useful in testing for a variety of web application vulnerabilities.

c) Web application scanning tools

These are tools that have been built for the sole purpose of testing web applications for vulnerabilities. They usually come with test templates out of the box but they may also be configured to perform specific tests.

d) Dedicated pen test platforms/frameworks

These are systems or collections of tools dedicated to scanning, identifying and exploiting vulnerabilities. These may be in the form of full operating systems containing a suite of tools (e.g. Kali Linux) or dedicated tools (e.g. Metasploit)

e) Packet sniffers

Also known as packet analyzer or network analyzers, these are pieces of hardware or software that are used to monitor network traffic. These may be used for traffic sniffing and analysis.

f) Password crackers

Tools dedicated to cracking passwords. May come in use when, during the test, encrypted passwords are found.

g) Scripting and programming languages

Generally, the testing would not be based solely on automated test performed by dedicated tools. Manual testing is an important part of the process and team members would be acquainted with scripting languages – perl, bash, powershell or programming languages – HTML, javascript, Java, SQL, etc. Testers can make use of these to craft malicious input to applications, alter responses sent to server requests or build custom scripts to perform certain tasks, as needed.