Alma Mater Studiorum · Università di Bologna

CAMPUS DI CESENA
SCUOLA DI ACHITETTURA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria Informatica

ICT Security: Testing methodology for Targeted Attack defence tools

Tesi di Laurea in Progetto di Reti di Telecomunicazioni LM

Supervisor:
Prof. Franco Callegati
Co-supervisor:
Dott. Ing. Marco Ramilli

Student:
Luca Mella

Session III
Academic Year 2012/2013

. . . to my Mother

Abstract

La tesi di laurea presentata si inserisce nell'ampio contesto della Sicurezza Informatica, in particolare tratta il problema del testing dei sistemi di sicurezza concepiti per contrapporsi alle odierne minacce: gli attacchi mirati (*Targeted Attacks*) ed in generale le minacce avanzate persistenti (*Advanced Persistent Threats*).
Il principale obiettivo del lavoro svolto è lo sviluppo e la discussione di una metodologia di test per sistemi di sicurezza focalizzati su questo genere di problemi. Le linee guida proposte hanno lo scopo di aiutare a colmare il divario tra quello che viene testato e quello che in realtà deve essere affrontato realmente.
Le attività svolte durante la preparazione della tesi sono state sia di tipo teorico, per quanto concerne lo sviluppo di una metodologia per affrontare al meglio il testing di sistemi di sicurezza a fronte di attacchi mirati, che ne di tipo sperimentale in quanto si sono utilizzati tali concetti per lo svolgimento di test su più strumenti di difesa in uno scenario d'interesse reale.

The submitted thesis treats topics of Computer Security, in particular it addresses the problem of testing security systems against nowadays advanced threats: *Targeted Attacks* (TA) and *Advanced Persistent Threats* (APT) in general.
The main objective of this thesis is the design of a meaningful test method for modern threat defence tools and systems. The proposed methodology aims to fill the gap between what is tested and what is actually tackled.
Activities performed during the course of the thesis are both theoretical, regarding the development of a methodology to better address the problem of testing security tools against TAs, and experimental because the proposed approach have been adopted in a real-world test session.

Introduction

The modern IT world is literally under the continuous assault of hidden attackers, malicious softwares are built and diffused every day all around the globe, malicious infrastructures are growing every day and more sophisticate intrusion technologies are constantly emerging. But this is only the less worrying part of the story.

Attackers evolved their strategies by blending them with military fashioned modus operandi, the results are *Targeted Attacks* and *Advances Persistent Threats*, threats highly focused on stealing intellectual propriety, intercept communications and even disrupt physical assets. This represents the nowadays security challenge for organizations all over the word.

Security industry started investing in products and technologies for fight back modern advanced threats even cooperating with academic research groups, a new generation of more sophisticated, intelligent, aware intrusion detection systems and tools has born. Signature based era is over.

In few years, IT Security panorama has radically changed its aspect and important issues have been raised, for example: *"How to test modern security products with respect to advanced threats?"*[32]. Mayor industry players and part of the *InfoSec* community are aware of the ***gap*** between what the consolidated testing methodologies are able to test and the latest real-word advanced threats to tackle. Old fashioned tests are no longer meaningful when dealing with the nowadays threat landscape.

This work aims to investigate and propose a testing methodology for advanced threat prevention products and systems, always with Targeted Attack in mind. *Lower the gap* between what is tested and what is actually tackled is the main goal of this thesis.

Following chapters will firstly introduce the wide topic of cyder-attacks 1, showing principal characteristics of modern threats, then will move on detection systems 2 by reviewing taxonomies and by proposing a custom categorization that could provide a useful information base for designing tests adopting the proposed methodology.

After this more theoretical review of the conceptual space of modern attacks and IDSs, the journey will proceed with an analysis of what is needed in order to tackle modern threats 3 and a brief review of some cutting edge technologies[1] is reported. A quick summary of major test methodologies is then performed and particular emphasis is posed on literature about security systems testing 4.

Finally, I will introduce a testing methodology for targeted attacks prevention products and systems, providing indications about how to deal with every test phase 5. Then a real-world test using this methodology has been described with emphasis on previously introduced workflow 6.

[1]At time of writing.

Contents

Introduction **iii**

1 Cyber Attacks **1**

1.1 Towards Cyber Attacks 1

1.2 Opportunistic Attacks 8

1.2.1 Attacks using internet vulnerabilities 10

1.2.2 Attacks using malwares 10

1.3 Targeted Attacks 10

1.3.1 Attack phases model 11

1.3.2 Information Gathering 12

1.3.3 Infiltration 13

1.4 Differences between OA e TA 14

2 IDS and Taxonomies **15**

2.1 IDS in Literature 15

2.2 Adopted Taxonomy 21

3 Defence Tools **25**

3.1 What does not work 25

3.2 Defence Approaches 27

3.3 Technologies 29

3.3.1 Traditional Defence Tools 29

3.3.2 Network/Host Probes 29

3.3.3 Sandbox 30

3.3.4 SIEM 31

3.3.5 Computer Security Incident Response Team 32

3.4 Open Source Tools 33

3.4.1 Anubis Sandbox 33

3.4.2 Cuckoo Sandbox 34

3.4.3 Malheur 34

3.4.4 OSSIM 35

3.5 Commercial Tools . . . 36
3.5.1 FireEye Platform . . . 36
3.5.2 LastLine Previct . . . 37
3.5.3 FortiGate . . . 38
3.5.4 SourceFire FireAMP . . . 39
3.5.5 WebSense Web Security Gateway . . . 40
3.5.6 Damballa Failsafe . . . 41
3.5.7 Symantec Endpoint Protection . . . 42

4 Testing Methodologies 43
4.1 Software Systems Testing . . . 43
4.1.1 RedTeam Exercise . . . 44
4.1.2 War Gaming . . . 44
4.1.3 Black Box test . . . 44
4.1.4 Crystal Box . . . 44
4.1.5 Vulnerability Test . . . 45
4.1.6 Double Gray Box . . . 45
4.2 Security Systems Testing . . . 45
4.2.1 Anti-Virus and Anti-Malware Products Testing . . . 45
4.2.2 Cloud-Based Products Testing . . . 47
4.3 Malicious Samples . . . 48
4.3.1 Sample Selection and Validation . . . 49
4.3.2 The Sample Creation Debate . . . 50

5 TA Detection System Testing 53
5.1 Testing the kill-chain . . . 54
5.1.1 Information Gathering . . . 54
5.1.2 Weapons Delivery and C2 . . . 54
5.1.3 Actions . . . 56
5.2 Testing Apples to Apples . . . 57
5.3 Testing Systems . . . 57
5.4 Benchmarking . . . 58
5.5 The Whole Picture . . . 59

6 Case Study: Testing Two TA Detection Systems 63
6.1 Preliminary Analysis . . . 63
6.2 Test-Bed . . . 64
6.3 Samples . . . 65
6.3.1 Collected Samples . . . 65
6.3.2 Created Samples . . . 66
6.4 Running Test: Malware Delivery and C2 . . . 67

6.5 Results 68

7 Conclusions **69**

A Clarifications **71**

A.1 Developed Tools 71

A.2 Products and Test Results 71

Chapter 1

Cyber Attacks

Facing the non trivial task of testing security systems for detecting targeted attack require a proper background about what a cyber attack is meant to be, and about the reason this particular kind of attacks is qualified as *"targeted"*.

In this chapter I will first provide a general background about cyber attacks by describing how academy and industry define them through a review of several taxonomies evolved toward computer era 1.1, once addressed this preliminary issue I will focus on what an *opportunistic attack* and *targeted attacks* are meant to be, with some notes about the origin of this class of attack strategies1.21.3. Later, I will recap the differences between these attack strategies1.4.

A brief overview on these topics is not only a matter of elegance and completeness, it also plays a role during the analysis and the categorization of detection systems, which is concerned with the proposed methodology.

1.1 Towards Cyber Attacks

The notion of cyber attack has assumed different flavors and several levels of details along time, each definition is not a priori right or wrong, simply they are placed at different abstraction levels and focused on distinct aspects.

Most of them specialize in aspects like attack results, attack process and methods of operation[51], defenders point of view[50], another good part of them show some kind of incompleteness when facing to more sophisticated and modern attacks[1]. This simple fact is a clear indication of the non triviality of this topic.

[1]Taxonomies should be exhaustive, comprehensible, unambiguous, mutually exclusive, useful

In the following paragraphs I will focus on some key taxonomies which may help to understand how the cyber attack topic has been treated since the beginning and how it has evolved during the years. Also, I will take a look at some taxonomies adopted in non academic environments.

Cyber Attacks in literature

Several taxonomies have been proposed since beginning of the information age, most of them have a particular focus on certain aspects like attack results, attack process, methods of operation [51] , defenders point of view [50], and more, but part of them show some kind of incompleteness when facing to more sophisticated and modern attacks[2]. This simple fact is a clear indication of the non-triviality of this topic.

In [21, 3.3] was proposed a quite general classification based on the results of the attacks. Three main classes are introduced:

Corruption which is any unauthorized modification of any kind of information is the system, with or without the presence of a clear intent.

Leakage is meant to be any situation where information flows to places it's not supposed to go. This include both an error selecting email recipient and information stealing performed by a *malicious software*(*malware*).

Denial is the failure to provide any kind of service which should be provided accordingly to policies.

Obviously this kind of taxonomy offers only a primitive way to classify attacks: it can only suggest some macro types of malicious actions that can be useful in some limited contexts. For example, if we are dealing with a *generic malware* this categorization could also be complete - in fact a malware may deny some services, can corrupt host configuration for ensuring its persistence, and could leak information from the target - but it surely is not mutually exclusive. Again, when thinking of a more complex and realistic scenario, things quickly collapse: how do the information gathering phase (eg. network scans) maps into this taxonomy? The answer is completely subjective.

Years later a process oriented point of view [40, 6.4] based on the *"means, ways, and ends"* operational sequence has been introduced. Here five orthogonal dimensions of this categorization:

[2]Taxonomies should be exhaustive, comprehensible, unambiguous, mutually exclusive, useful

Attacker try to model *who* is performing the malicious actions. But [36, 1.5] reports that the proposed values for this dimension are not mutually exclusive.

Tools one of the first attempts to distinguish between direct commands entered by attacker, programs to automatically exploit vulnerabilities, usage of autonomous agents (**bots** or malware in general), and distributed attacks.

Access describe what kind of vulnerability (design, implementation, configuration) have been exploited for gaining unauthorized access to processes, files or data in transit.

Results this dimension is quite similar to that one introduced in the previous paragraph[21, 3.3]. **Theft of service** concept is introduced: unauthorized use of a host or service without any kind of degradation.

Objectives some kind of main meta-goal of the attack. For instance political gain, financial gain, damage or challenge.

However, despite the more complete approach, some key aspects of modern attacks will remain out of the model. For example, no sufficient emphasis is posed types of malware used in the attack, or also there is no mention about non-computer vulnerabilities (eg. social and organizational ones). Also, there isn't any reference to the **attack phase** detected. See section 1.3 for details.

A more modern re-visitation of [40] is proposed in [36], where the process based approach is enhanced considering realistic case studies. The taxonomy meta-model is composed of four dimensions sub-divisible in more levels:

Attack vector better care about tools used during attack. A multi level approach ensure to better distinguish between different kind of viruses, worms, or Trojan horses. Also the information gathering phase has been introduced.

Target focuses on hardware components (disk, network equipment, keyboard, etc...) and fine detail about involved software (eg. type of application and version). But still *missing is the presence of human target.*

Vulnerability focuses only on implementation, design, and configuration issues.

Payload or effects mainly the same of previous taxonomies (corruption, disclosure, theft of service, denial) with better detail capabilities.

With this taxonomy some modern attacks (maybe the opportunistic ones) can be effectively categorized, but is also clear that this model is not exhaustive.

Recently, in [68], the approach on attack classification has been changed quite radically. For example, in the `AVOIDIT` (Attack Vector, Operational Impact, Defense, Information Impact, and Target) taxonomy a more flexible tree structure has been introduced despite the previous *list oriented* ones.

The changes don't involve only the structure of the taxonomy, also the main point of view has been revisited, in fact the focus has moved on **blended attacks** which may involve different techniques and steps within a single attack attempt[19]. For instance an attack that involves both Trojan infection, denial of service, and also the exploitation as a vector of infection.

As suggested by the acronym its main dimensions are:

Attack Vector which also include *social engineering.*

Operational Impact that also include a sub-tree about *malware installation.*

Defense distinguishing between mitigation or more complete remediations.

Informational Impact with in different flavors like:

- *Distort*, as information modification.
- *Disrupt*, as more fine grained form of Denial of Service.
- *Destruct*, information destruction like file removal.
- *Disclosure*, with emphasis on stolen informations.
- *Discovery*, with emphasis on scanning and fingerprinting.

Target Not only hardware and software target, but also users personal information.

A graphic representation of this more effective taxonomy is available in fig.1.1.

Of course, this review is not exhaustive, in fact only some key-taxonomies have been reported, further material and other reviews can be also found in [57, 81, 37, 51, 50].

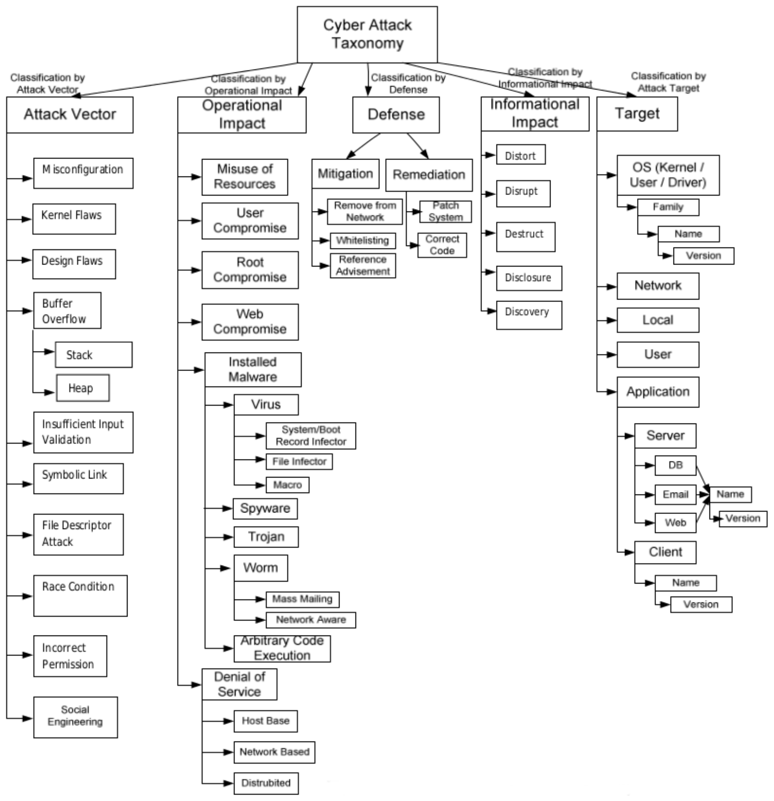


Figure 1.1: AVOIDIT Taxonomy (from [68])

Cyber Attacks in industry

Academic research is not the only player in the IT Security word, it's also important to take a look at which cyber attack taxonomies are in use in the more pragmatic *Info-Sec* industry and in the military environments.

A notable case is represented by ENISA (European Union Agency for Network and Information Security) which suggest to use the taxonomy described by Howard and Longstaff in the '98 [41, 61]. In this taxonomy, there are three main entities:

Event composed by two dimensions:

- *Action* Low level (malicious) action like scan, flood, steal, and so on.
- *Target* HW or SW entity targeted (eg. user accounts, network, process, etc. . .)

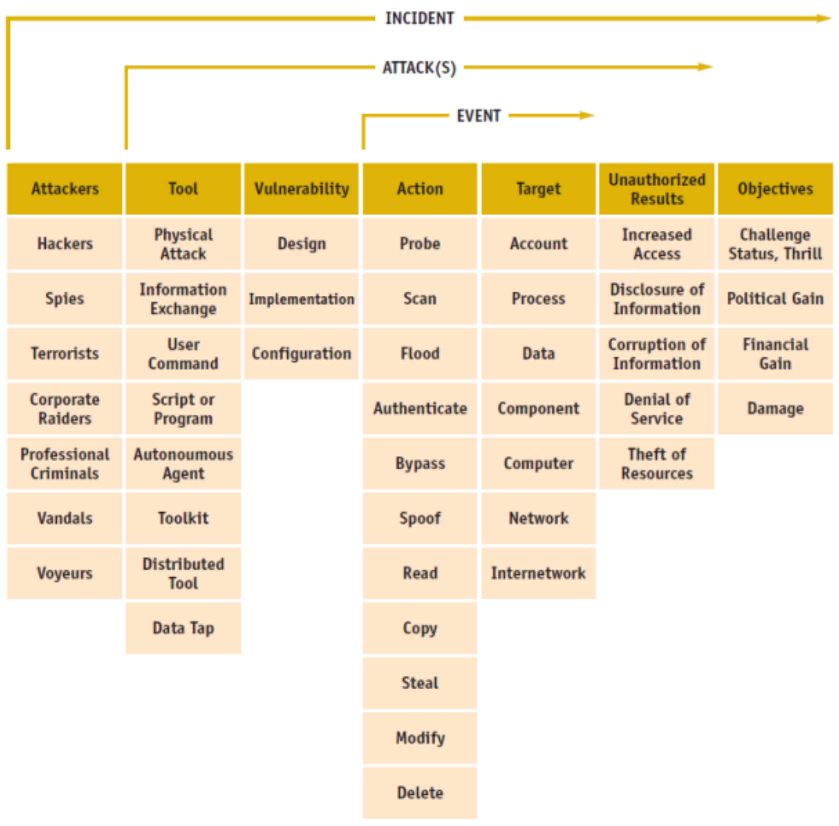
Attack composed by **events** and three extra dimensions:

- *Tool*, like in [36, 1.5] there are user commands, scripts, tool-kits, autonomous agents, *information exchange* tools, and others.
- *Vulnerability* implementation, configuration, design.
- *Results* same as [21, 3.3] plus *theft of resource* and *increased access* (aka privilege escalation).

Incident composed by one or more **attacks** with two supplementary dimensions:

- *Attackers*, as the type of attacker (hacker, spy, corporate raiders, criminals, etc. . .)
- *Objectives*, like political gain, financial gain, damage or challenge.

As the reader can see, this taxonomy is based on the process oriented one introduced in [40] and it's surely capable to better fit the conceptual space of a cyber attack, but still miss some key concepts of modern attacks, like the nature of the tools used, and the possible human-nature of the targets. In fig.1.1 shows a graphic representation of the taxonomy.


INCIDENT
ATTACK(S)
EVENT

Attackers	Tool	Vulnerability	Action	Target	Unauthorized Results	Objectives
Hackers	Physical Attack	Design	Probe	Account	Increased Access	Challenge Status, Thrill
Spies	Information Exchange	Implementation	Scan	Process	Disclosure of Information	Political Gain
Terrorists	User Command	Configuration	Flood	Data	Corruption of Information	Financial Gain
Corporate Raiders	Script or Program		Authenticate	Component	Denial of Service	Damage
Professional Criminals	Autonomous Agent		Bypass	Computer	Theft of Resources	
Vandals	Toolkit		Spoof	Network		
Voyeurs	Distributed Tool		Read	Internetwork		
	Data Tap		Copy			
			Steal			
			Modify			
			Delete			

Figure 1.2: Taxonomy suggested by ENISA (form [61])

Another non academic approach is one that emerged from military environments, which surely are first class actors in the *Info-Sec* community.

Reading articles published since early 2k's, is possible to discover really interesting facts about ***cyber-warfare***: *strong similarities with modern cyber attacks are observable.*

For example in [47] - published in 2001 - a cyber-warfare taxonomy is introduced with emphasis on the attack process:

Cyber warfare (CyW) act intended to compromise other nation will, executed against software infrastructure.

Cyber infiltration (CyI) defence penetration.

Cyber manipulation (CyM) after infiltration, take control of the system.

Cyber assault (CyA) after infiltration, destruction of software and data, damage system.

Cyber raid (`CyR`) after infiltration, could result in data alteration, destruction or acquisition, leaves the system intact.

Cyber attack (`CyA`) CyI, CyM, CyA, or CyR.

Cyber crime (`CyC`) CyA without the intent to affect national security.

Looking at the description of modern cyber attacks in section 1.3, the contribution of this kind of categorization results pretty evident.

Additionally, *Info-Sec* industry has also some de-facto standard like the `VERIS Framework` [31]. This taxonomy is currently adopted by Verizon in its annual *Data Breach Investigations Report*[3], which represent an excellent resource for reasoning about future trends of this sector. Over 19 of the major *Info-Sec* companies cooperate with Verizon in order to provide information about security breaches using a shared framework, easing further intelligence and data analysis. This model is composed by six main dimensions:

- *Demographics information*
- *Incident classifications*, like type of attacker agent, action performed on assets, assets, and assets attributes.
- *Discovery and Mitigation*, specifying time spans of major attack phases, discovery method, evidences , and control measures.
- *Impact classification*, as direct impact (eg. lost assets), indirect impact (eg. stakeholders reactions), impact estimation and qualification.

1.2 Opportunistic Attacks

Not every exploited systems have been attacked with a *targeted strategy*. Often an attacker needs its own (illegal) infrastructure for further attacks and search the Internet for vulnerable systems to take control of. Once a vulnerability is discovered and exploited, the attacker may be interested only in using the machine as a proxy for Spam, phishing or BotNet control. In this scenario, the economic loss and recovery cost for the target is minimal [17].

OAs attack vectors (means by which the attack is performed) may lack of advanced social engineering, because the attacker doesn't have nor is interested in specific information. A list of commonly used vector is the following [22]:

[3]Published at http://www.verizonenterprise.com/DBIR/ at time of writing

Email: in OAs, mails usually contain phishing links of commonly used websites. Those links may stole password, download binaries or simply validate the email address.

Message: similar to email, but using some chat-based protocols.

File Sharing: malicious software may be distributed by means of a sharing system protocol.

Trojan: in addition to perform the expected functionality, softwares may carry malicious code, such as a back-door.

Vulnerabilities: Internet services, anti-viruses, browsers, OSs and other program that expose themselves to the Internet, may have vulnerabilities capable of granting access to an attacker.

Passwords: although this can be considered a vulnerability itself, a weak password is a standalone and easy-to-use vector. A weak password can be broken regardless of any update or security system a sys-admin could install.

OAs use these vectors in a fully automatized way. Phishing servers generate email, bots connect to chat services and scanners scan the network for vulnerabilities and weak passwords. This is why the number of OAs is enormous:

1. The Verizion data breach analysis[30] reports more than 70% of attacks as OAs.

2. A Verizon analysis [44, 45, 46] says that, on average, a system on the internet is scanned after 11 minutes from the connection.

3. Honeypots receive hundreds of thousands of new malicious samples every day.

What makes an attack an OA, is the "*non-targeted-target policy*". The attacker selects the most general social engineering approach, the most vulnerable target of the internet, maximizing a profit over risk ratio. Among millions of targets, a single secured system is not worth it, thus, the attacker is not interested in exposing itself by repeatedly scans a server port. In [45] it's said that "97% of opportunistic attackers try one port and 81% send a single packet".

1.2.1 Attacks using internet vulnerabilities

The pattern is simple: the attacker automatically scans known ports used by login services (such as RDP or SSH), web administration pages or vulnerable programs.

In [44], the author notes that the source port of the scans is not fully random, but it's often set to a value that may bypass some firewall. This confirms the nature of the OAs: assuming the attacker has no knowledge of the targeted system, the parameter allowed by the majority of generic defense systems is used.

1.2.2 Attacks using malwares

In order to exploit common clients, the attacker usually writes a trojan by inserting its malicious code inside a pseudo-working program (e.g. the newest *Super Video Converter from .donald to .goofy*). Such types of malware don't requires advanced exploiting techniques, because the user is willing to grant privileges of his own (for instance, by clicking *Allow* on the *UAC* prompt).

By generating new variants of the same malware and creating new trojans, an attacker could infect lots of clients, creating its own botnet with a minimal amount of work. Loosing a single bot is not a loss for the attacker, because the infected machine isn't the real target.

However, this approach is risky. Malicious software and phishing links are distributed wildly, exposing to the world the attacker, its C&C (Command and Control), the compromised domains and the method used to exploit the system. Knowing that, the attacker may periodically check if its domains or malware were discovered and act accordingly, by using a new compromised domain or writing a new malware variant [72].

1.3 Targeted Attacks

A *targeted attack strategy* drives the attack through specific objectives against a well defined target. In these scenarios, an attacker gathers public and private information and design attack vectors to hit the target without being detected.

1.3.1 Attack phases model

At the turn of 21th century, cyber crime became acknowledged as a real threat by governments. A new military doctrine was needed. In [47] a military taxonomy for cyber crime acts is proposed:

Infiltration: during this stage, the attacker identifies all the system that can be attacked, that is, every system that can accept external input. That include physical and remote infiltration.

Manipulation: following infiltration, an attacker can alter the operation of a system to do damage or to propagate the infection.

Assault: following infiltration, the attacker can destroy software and data or disrupt other systems functionality.

Raid: following infiltration, the attacker can manipulate or acquire data and vital information.

Despite the military terms, this taxonomy describes actions that occur during TAs, being the target a military organization or not. In [42], Lockheed Martin company proposes a military-inspired intrusion kill chain, with respect to computer network attack (CNA) or computer network espionage (CNE). The model describes the stages the attacker must follow as a chain: any deficiency will interrupt the entire process.

1. **Reconnaissance:** intelligence and information gathering phase.
2. **Weaponization:** create the malware and inject it in a *weaponized deliverable*.
3. **Delivery:** transmission of the weapon by means of an attack vector.
4. **Exploitation:** the malware acts, exploiting the system and compromising the machine.
5. **Installation:** the malware installs a RAT (remote access tool) or a Backdoor to allow the attacker an access to the environment.
6. **Command&Control:** the malware establishes a connection to the attacker C&C, getting ready to receive manual commands.
7. **Actions on Objectives:** the attacker takes actions to achieve his objective.

The model above (proposed in 2011) expands the cyber warfare taxonomy by focusing also on the necessary steps before infiltration and between infiltration and attack. These steps need to be followed and offer the target an opportunity to detect or disrupt the kill chain.

Once the infection is consolidated, an attacker can perform various actions to reach its objective. In [76] a typical modus operandi is proposed, basing on the result of various forensic investigations. Unlike the kill chain model, the following phases are presented from the target point of view:

Incursion: in this phase the attacker attempts to penetrate a network.

Discovery: once inside, the attacker analyzes the network to spread the infection and identify the real objective.

Capture: when a worthy computer is infected, the attacker *installs* an advanced RAT to improve the stealth capabilities.

Data Exfiltration: the attacker uses the RAT to stole documents, password, blueprints. The attacker can also use the advanced capabilities to spread deeper into the network.

1.3.2 Information Gathering

A TA against an organization requires as much information as possible. The intelligence is needed to create the right weapon, choose the correct attack vector and which exploitation to use. Recon is the very first phase and must not be detected by the target, since an early detection could warn the target and disrupt the entire attack.

A list of critical information needed by the attacker could be[28]:

Employee's personally identifiable information: phone numbers, addresses, biometric records and so forth can be used for the selection of the right target and attack vector.

Network layouts: web and mails servers, their location, software version and system fingerprints reveals necessary information for the exploitation, installation and C&C phases.

Company files: reports, databases and source code, for instance, could reveal software and useful data.

Company information: information such as partners, services, mergers and acquisitions, facilities plants could delineate social engineering approach and attack vectors via physical infiltration.

Organizational information: for a successful social engineering attack and in order to identify the correct target the attacker needs technical staff and C-team names, organizational charts and corporate structure details.

Work interaction: emails content templates, inter-communication protocols, security and authentication mechanisms are essential information for the attack design.

Tons of public information can be obtained by searching in the employee's social networks profiles, public calendar entries and via specific Google searches. Private information can be harder to obtain and often requires the use of social engineering and of search engine hacking[28], that is, using search engines to find vulnerable web pages.

1.3.3 Infiltration

Infiltration is the macro-step that includes phases between *recon* and *action on the objective.* In this phase, the attacker uses the available intelligence to design the correct method to reach the objective. The first step is defining the malware requirements, for each step of the reversed kill chain:

Actions: which action must be performed? Can they be automatized or require constant control?

C&C: is a C&C required? Must the malware exchange data with the attacker or not?

Installation: does the attack requires a backdoor or is it a one-shot attack?

Exploitation: which security system must the malware bypass? Are administration privileges required?

Delivery: which is the right attack vector? Can be performed from a remote location or requires physical contact?

Weaponization: which is the right deliverable to use?

If the intelligence is good enough, the result is malware able to exploit and bypass the target defenses.

1.4 Differences between OA e TA

Intent defines the attack strategy and threat[17].
An attacker could target a vulnerable resource to gain money and valuable information. In such case, the target is chosen by searching for a known security flaw without any a priori knowledge of the system infrastructure. We call this kind of attacks *opportunistic attacks* (OA).

On the other hand, an attacker having a specific target to compromise would behave in a totally different way. These attacks are designed to bypass the target security systems, knowing the architecture and the infrastructure. Moreover, the attacker may be authorized to access to a certain area, becoming an inside threat. These attacks are called *targeted attacks* (TA).

TAs and OAs strategies imply a different intent, knowledge and work from the attacker's point of view and, more importantly, they require different defence methods from the target side. TAs and OAs differ under multiple aspect, even for the target point of view. The TA exploitation, unlike in the OA scenario, is just the entry point of the attack on the target. The attacker steals secrets and intellectual property and can remain undetected for weeks[39]: the economic loss is much higher than having an infected machine that acts as a spam proxy.

So, the skill level required is different. In TA the attacker must model its strategy to bypass the target defenses. This implies extremely customized malwares, targeted scans, policies exploiting and so forth. OAs, on the other hand, is focused on easy, risk-less and vulnerable target using automatic tools. This process requires a lower skill than designing TAs.

Another difference is the *context* itself. While an OA is usually performed from a remote location on public server, a TA can exploit local information and policies to bypass the security perimeter. *BYOD* (bring your own device), for instance, can expose the network to heavy security risks. Years ago, workers carried their laptops able to connect to the enterprise network, but with BYOD, the perimeter, the circle of thrust, is not defined anymore[75]. Since every personal device can be part of the network, a TA can be performed by infecting an employee's personal smartphone and thus gaining an inside access.

Chapter 2

IDS and Taxonomies

The other piece of background needed regards IDSs. The proposed guideline treats this kind of objects, or at least a part of intrusion detection system which also address targeted attacks.

In this chapter I will firstly review IDSs through an evolution of taxonomies over the latest years 2.1, providing a general and well-defined background. Then I will resume all the useful key aspects for the purposes of the proposed guideline 2.2, specifying a custom taxonomy that will be used as reference during the testing.

2.1 IDS in Literature

Since the first cyber attack notice, *Intrusion Detection Systems* (IDSs) have started a competitive co-evolution: more sophisticated attack led to finer detection techniques and vice-versa. In the previous section I have reported the evolution of cyber-attack through models developed in several epochs and environments, now - for the sake of completeness - I am introducing reference models, taxonomies and typical approaches in Intrusion Detection Systems (IDSs) development.

One of the first IDS model proposed dates back in 1987[27], it contains a sufficient general description of the main component needed for this kind of entities and provide some sort of architectural reference for future systems. The model provide six main components:

Subjects as *who* interact with the system. Normally users.

Objects as resources handled by the target system. They can be files, devices, connections, etc…

Audit records records logged by the target system in response at some subject action.

Profiles structure that characterize the behavior of a subject w.r.t. objects.

Anomaly records generated by IDS when some abnormal behavior is detected.

Activity rules action to take in case of certain events, eg. anomaly detection, profile updated, rule update, and so on.

This general structure - or almost part of it - can be found in all current IDSs. After a quick glance to a general IDS architecture, I am going to illustrate a non-exhaustive review of the main IDS taxonomies proposed in the last decade. As for cyber attack, a clear taxonomy plays a really important role during comparisons and further intelligence on collected data.

A notable work in the field of IDSs taxonomies has been published in 1999[26], the authors provided a basic reference on what kind of measure perform on IDS in order to obtain soundness evaluation[26, 2.3]:

Accuracy the *True Positive* and *True Negative* rate.

Performance as the rate at which audit events are processed. Note that this parameter is crucial in case of real-time detection.

Completeness the *False Negative* rate. This measure is really hard to evaluate outside test-beds, because of the impossibility to have global knowledge about attacks.

Fault tolerance how a IDS is capable to resist to attacks and failures.

Timeliness this measure the time needed for propagation of the result of the analysis. Quicker propagation to security team lesser time for the attack.

Aside for evaluation metrics, [26, 3] has introduced a really powerful taxonomy based on IDS functionality:

Detection Method how detection is performed and which kind of model are used.

- *Knowledge based*
- *Behavior based*

Behaviour on Detection what the IDS does after a detection.

- *Passive*, reporting only.
- *(Pro)Active*, trying to stop attack.

Audit Source Location where IDS retrieves informations.

- *Host Log* events captured on host machine.
- *Network* using network sensors.

Usage Frequency how the IDS is meant to be used.

- *Continuous monitoring*
- *Periodic analysis*

One of the key aspects of this powerful taxonomy is the emphasis on *detection method*, for this reason I consider useful spending few more words about it:

Knowledge Based The main characteristic of this approach is to search for evidence of attacks based on current knowledge. This is **not** only the case of the classical, standard approach based on *signature matching* used by the whole host-based Anti-Virus solutions, but also systems that rely on some kind of *expert system*[26, 3.1.1.1], preconfigured rules, or pattern matching too. These detection methodologies are all based on previous knowledge, or previous known models (eg. Petri Nets[26, 3.1.1.3.]).
This family of detectors is known for its good accuracy and performance, but it lacks in detecting *non a-priori known* attacks or attack patterns.

Behaviour Based This kind of detectors starts with a completely different assumption: alerts are generated by deviation from *"normal"* model of behavior.
This means a behavioral approach requires a more sophisticated set-up, and at least a *non static*, *non a priori* definition of *"normal"* behavior. The conceptual gap between this approach w.r.t. knowledge based one is in the capability of the system to dynamically learn what *"normal"* behavior mean in the specific installation context.
However, "learn" does not mean it must only use machine learning approaches, in [26, 3.1.2.1,3.1.2.2] are cited examples of behavioral IDSs

based on statistics and even expert systems (using dynamic inference rules).

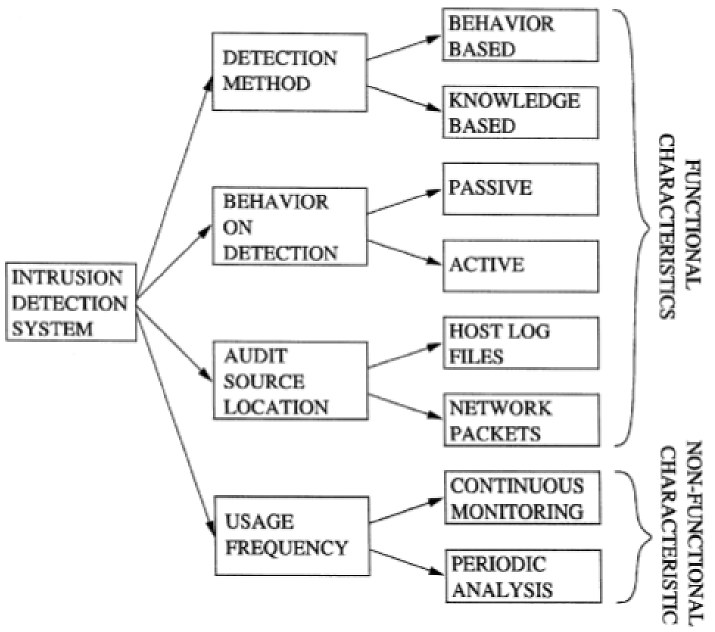


Figure 2.1: Functional taxonomy of an IDS (form [26])

Continuing on this line, an interesting work[70] has posed particular emphasis on the *detection properties* of the IDSs, in this article the authors have proposed a taxonomy for *detection principles* keeping a functional point of view, like in [26]. However, this taxonomy results interesting due to some important details I will briefly discuss. First of all, are present three main detection principles:

Anomaly based which involves some kind of behavioral analysis. A particular attention is posed on how the "normal" behavior model is generated, in fact two sub-categories are introduced:

- *self-learning*, citing approaches like artificial neural network (ANN), rule modeling and descriptive statistics.
- *programmed*, using techniques like thresholds, state series modeling, simple rules, etc...

Signature based based on the knowledge of a model of attack, this kind of IDSs can be programmed with a priori decision rules (eg. petri-net, state transition, string matching, simple rules, etc...)

Signature inspired a miscellaneous category, where attack model are stronger than normal behavior model. A *self-learning* approach is contemplated for this kind of IDSs, for example an automatic feature selection may change the weight of some detected events in order to increase accuracy on the specific installation.

This description may be particularly useful for getting aware of which kind of techniques can be used in order to develop a particular kind of IDS.

But the most interesting detail introduced is the *intrusion categorization* from the IDS point of view: **well known intrusions**, **generalisable intrusions**, and **unknown intrusions**. Where these types of intrusion also represent different levels of difficulty of the detection [70, 4.4.3]. This subdivision is critical during IDS evaluation and may have heavy side effect on the measured accuracy. For this reason mapping attacks in one of these class could be really valuable.

An alternative taxonomy, described in [77], has a different focus w.r.t. functional ones introduced above. The main idea of this work is to introduce the concept of *"intrusion matrix"* as the combination of output produced by the IDSs and the data scale used. The output produced by IDSs is mapped five categories:

Detection when the system outputs an indication of a state change within a network or host with. Note that no classification about the nature of the change is performed, there is only the *assumption* that the change is related to an attack.

Recognition capability of the IDS to declare the type of attack (eg. DDoS, privilege escalation, exploit, etc...)

Identification capability to identify the specific event. For example can determine when an exploit is a stack overflow from heap overflow in a particular application specific vulnerability.

Confirmation ability to react to the specific attack enforcing fine or coarse grained countermeasures.

Prosecution capability to identify the attacker.

And the contemplated data scale are:

File monitoring individual files for changes or access.

Host monitoring application running on hosts.

Network monitoring network packets between hosts, servers, and other network devices.

Enterprise monitoring traffic originated from trusted sources of an organization.

Also, this taxonomy may offer a direct visual representation of the *footprint* of real world IDSs. In figure 2.1 are shown several types of IDS which can be currently deployed on organizations networks. Obviously this taxonomy offers a completely different - *non-functional* - point of view for IDSs categorization.

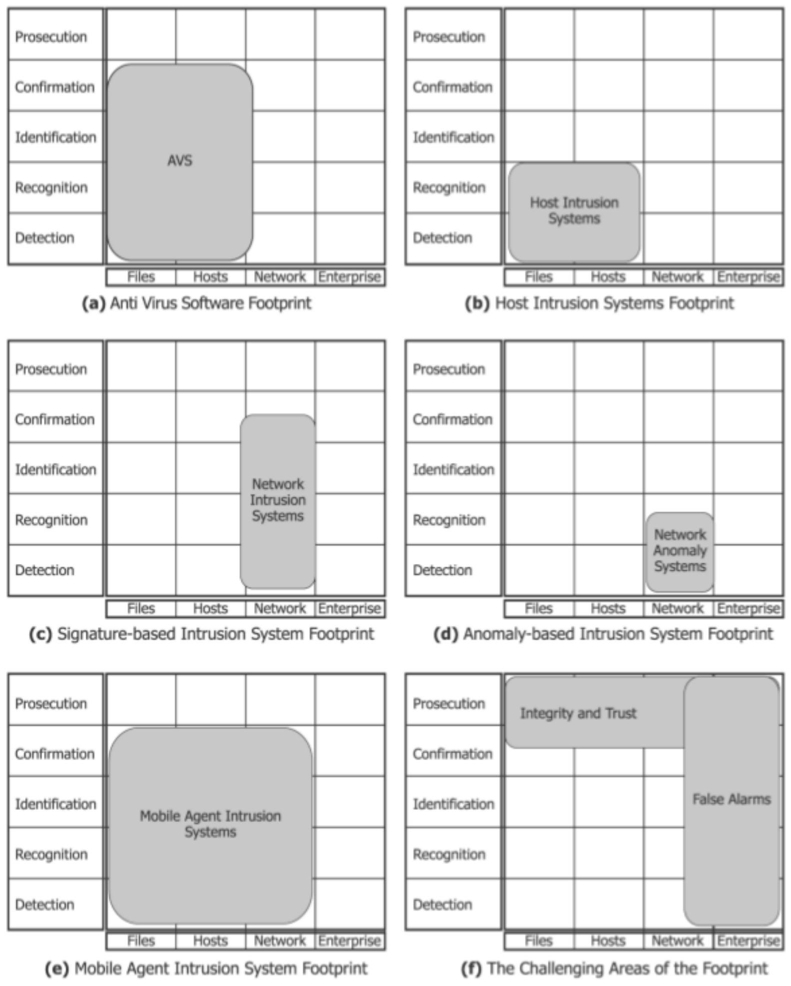

Prosecution
Confirmation
Identification
Recognition
Detection
Files
Hosts
Network
Enterprise
AVS
(a) Anti Virus Software Footprint
Host Intrusion Systems
(b) Host Intrusion Systems Footprint
Network Intrusion Systems
(c) Signature-based Intrusion System Footprint
Network Anomaly Systems
(d) Anomaly-based Intrusion System Footprint
Mobile Agent Intrusion Systems
(e) Mobile Agent Intrusion System Footprint
Integrity and Trust
False Alarms
(f) The Challenging Areas of the Footprint

Figure 2.2: Footprint of IDSs using IDS matrix taxonomy (form [77])

2.2 Adopted Taxonomy

Every taxonomy cited in section 2.1 cover interesting aspects needed in the proposed methodology[77, 70, 26], so I will not merely adopt one of them for my purposes, I am going to define an ad hoc taxonomy where to map targeted attack detection systems in order to:

- Properly classify the detection system under evaluation and identify weakness and strengths of the adopted approaches and technologies.
- Provide a valid background information to develop the right tools for testing the detection systems.

The proposed taxonomy will focus on aspects like **information sources**, **detection approach**, **recognition capabilities**, and **reaction capabilities**. All these aspects are quite important in order to evaluate detection systems that claims to be able to handle advanced, targeted, cyber attacks. In detail the reference framework will be structured as follows:

Information Sources aims to define where detection system gather its information, and also what kind of data it treats. A first level subdivision could be similar to that one described in [26, 3]: *host based* data sources and *network based* data sources. Here I consider several capabilities like:

- File system monitoring
- Registry and processes operation monitoring
- Host memory monitoring
- Access to network/host meta-data, like Network flows or SNMP informations.
- Access to network data, full traffic captures.
- Access to application level data and/or meta-data (eg. http server log or protocols understanding).

In other words, which kind data feeds the detection system.

Detection Approaches I firstly adopt a categorization similar to that one proposed in [70], which may help to summarize how gathered data are used in the detection process. Secondly technologies adopted in the implementation of the system are concerned. In fact the type of the technologies implemented may enhance detection capabilities despite the limitations of the general approach of the system. In detail the first level categories are:

- *Anomaly based*, when input data are processed in order to find deviation from a "normal" behavior model. Typically self-learning techniques and/or statistical models and rules are adopted during this kind of approach.
- *Knowledge based*, when the detection is driven by direct knowledge of the threat, by recognition of malicious patterns, or by knowledge of application/protocol states flow or also predefined rules. Typically this approach is focused on *knowledge of intrusive behavior*.

- *Hybrid*, when both intrusive behaviour and normal behaviour knowledge are used, hopefully outputting distinct alerts in order to properly weight the detection.

The list of technology types should not be defined in this phase due to the high number of possibilities encountered in a real world scenario. For this reason, here is reported a list of key features which should be investigated when defining a technology type:

- *Goal* of the technology, what kind of problem it should solve, or more specifically which kind of things it should detect.
- *Mainstays*, key features of the implementation and scenarios where it can be useful.
- *Limitations*, when it may fail and how it can be bypassed by an attacker.

So, in this part of the taxonomy detection approaches are addressed via two different level of abstraction, an higher coarse grained one - and also well discussed in literature - and another one much more fine grained based on technological specification.

Recognition Capabilities the ability of the system to detect which kind of threat has been detected, like *Drive by Downloads*, *DoS*, *Malware callbacks*, *Malware Downloads*, and so on. The recognized events could not be exhaustively listed because the type of anomaly detected could be not only at technical level like cited ones, but also I concern higher level recognitions like “infection”, “information gathering”, “data leak” ,or some other higher level states which may characterize attacks and more specifically targeted attacks.

Reaction Capabilities how the system can counter-strike to the recognized threat. Example of this kind of capabilities are *“blocking”*, *“reporting”*, *“providing evidence”*, ...

Chapter 3

Defence Tools

In this chapter, the kinds of bullets needed to face the TA detection challenge are briefly discussed. The main objectives of a defence system able to deal with TAs is to detect *unknown threats* in order to block current attacks and even block further intrusion attempts from the same attacker.

The following sections will show what kind of approaches are not optimal for the TA detection purposes 3.1 and will point out to a holistic approach which employ specific knowledge of the targeted attack kill chain 3.2.

Once discussed a possible defence approach, I will briefly describe which kind of components/entities are useful for setting up a TA detection system 3.3 and then I will review *some* of the latest solutions/products - both open-source and commercial - that claim to deal with TAs and APTs 3.4 3.5, mapping them in the taxonomy introduced in 2.2.

3.1 What does not work

In this section I'll have some thought about TAs with respect to intrusion detection systems, from both a theoretical/academic point of view and a more pragmatical industry oriented one.

Theoretical point of view

The first thing to do for understanding what kind of thread are we facing, is a modest abstraction work for finding a soundness mapping to one of the model introduced in chapter 1.

Thinking about the difference between TA and OA discussed in section 1.4 the **unknown** nature of a TA become explicit: it depends on target defence systems, policies, custom scans, and also involve some kind of ad-hoc malware. So, following the taxonomy proposed in [70, 4.4.3], we are facing the most difficult to detect type of intrusion possible. Having that mapping, then we can review some of the most recent literature about the intrusion detection problem for understanding which kind of IDSs may better deal with this kind of problems.

A recent IDS review [56], evaluate in depth the characteristics of a great number of IDSs proposed during the last decade, mapping all of them in a taxonomy really similar to those ones briefly discussed in section 2.1. In this work exist three IDS types: *signature based* (knowledge based), *anomaly based* (behaviour based), and also *state-full protocol analysis based*[1] (specification based). Main pros and cons of these intrusion detection types are the following[56, Table 1]:

	Knowledge Based	**Behaviour Based**	**Specification based**
Pros	Simpler, effective for **known** attacks	Effective for **unknown** attacks	Distinguish unexpected sequence of commands
Cons	Ineffective with **unknown** attacks, maintaining knowledge is time consuming	Profile accuracy, rebuilding profiles	Resource consuming protocol state tracing, can't inspect attacks looking like benign protocol behaviours

Table 3.1: Pros and cons of the three main types of IDS

Once accepted these two facts, is quite clear that the thread represented by TAs can't be faced with traditional countermeasures, which are commonly characterized by a signature, pattern based, state based, or in general knowledge based approach.

Industry point of view

Nowadays, *Info-Sec* industry players *seem* to be pretty aware of of the situation, in fact the term *"Advanced Persistent Threat"* is a live motif in security communities, a simple research using Google (or similar) can clearly

[1]An IDS which knows how to trace protocols states and use network protocol models for its detections.

demonstrate it. Also, Anti-Virus (AV) industries perfectly know that signature based technologies are not enough from long time, different commercial solutions plug “heuristic” based technologies, *“behavioral”* engines, or *“cloud intelligence”* support [49, 14] to classical pattern matching engines. However, even OAs have evolved during years by enhancing and automating evasion capabilities[16, 2.2] even using third party services for generating new version of the same executable [58], for this reason some interrogatives about what does the industry word intend when talking about “heuristics”, “behavioral”, and “intelligence” remain: taking a look to few AV company blogs[48, 71] suggest that their main focus remain in OAs, so all these detection techniques may be interpreted as oriented on **generalization of known attacks/intrusions**, which is *not* exactly an unknown attack.

AV industry apart, other technologies like Network Intrusion Detection Systems (NIDS)[66], Firewalls and Proxies are almost standard equipment in company networks for boundary definition, hardening and monitoring. However this kind of systems are typically based on preconfigured rules, pattern matching, or also protocol state inspection, which are all techniques based on knowledge (or almost all). Just for this reason these systems does not provide adequate detection capabilities for unknown attacks, in addition consider the BYOD issue reported in the previous section 1.4.

Having said that is pretty clear that further investigations should be performed in order to assess the efficacy of defence systems and technologies against TAs.

3.2 Defence Approaches

A general defence approach against TAs is the interaction between multiple security layer, called **Multi-Level Defence**. An individual security system cannot stop a targeted attack, but their holistic union can provide information useful for prevention and detection.

The effectiveness of the multi-level defence depends on the *ability of organize*, *analyze* and *use information* obtained by the various security layers, therefore dynamic analysis assumes a leading role during the defence process. Data sources are queried to extrapolate information that must be related to an attack. The analysis process has not only to distinguish benign events from malicious ones, but also to attribute information to the right attack. A wrong or imprecise analysis could afflict correct prevention and detection [42].

The multi-level defence is based on attack phase models (see section 1.3.1). Phases model is an attempt to bind an event to a particular context, situating an information inside a bigger pattern so that the analyst can identify the attack, the current phase and act accordingly.

Example: Intelligence-Driven Computer Network Defence

Lockheed Martin published a defense methodology[42] defining intelligence-driven CND as a risk management strategy that addresses the threat component of risk, incorporating analysis of adversaries, their capabilities, objectives, doctrine and limitations.

Briefly, the intelligence-driven CND uses the kill chain model (see section 1.3.1) to analyze past and current intrusions, in order to create patterns for every chain link of a specific intrusion campaign. By detecting and disrupting a single link, the kill chain cannot proceed and the attacker must change his strategies. If the defender evolves faster, the complexity and the cost for the attacker grows.

The analysis is based on *indicators*, pieces of information that objectively describe an intrusion. Indicators can be atomic, computed and behavioral. Atomic indicators retain their meaning in the intrusion context, computed indicators are derived from data involved in an incident and behavioral indicators are collection of atomic and computer indicators, possibly quantified and binded by combinatorial logical operator.

The analyst obtains indicators by analyzing tools logs and reports and he matures them to create a pattern that can be used to match future activities. The matched activity could reveal new information, that will be analyzed to create new indicators to be added to the knowledge base.

Assuming that the indicator attribution is correct, the analyst can analyze the intrusion campaign, determining the **TTP** (Tactics, Techniques and Procedures) of the intruders and even their intent. The difference with common intrusion response or detection systems, is that *the intelligence-driven CND tries to determinate how the intruders are performing the attacks* and not only what they are doing.

In February 2013, Lockheed Martin claimed to have successfully identified an intruder with valid credentials, using a multimillion-dollar framework based on the Kill Chain model[38].

In the `VERIZION DATA BREACH INVESTIGATION 2013` [30], the team recommends to focus on the kill-chain approach connected with the information provided by the `VERIS` framework[31].

3.3 Technologies

As discussed above, an effective defence solution could be based on a holistic approach which involves several types of security components and different technologies. For this reason I will summarize what kind of entities should be useful for the TA detection challenge.

3.3.1 Traditional Defence Tools

Of course a basic layer of classical tools is a must-have, in detail I refer as "traditional tools" the following pieces of software:

- Anti-Virus software (signature-based)
- Firewall (level 3/4) and Next Generation Firewall (NGF)[2]
- Application Proxy, at least HTTP/HTTPS proxy
- and also Application Firewalls (eg. WAF)

3.3.2 Network/Host Probes

In order to deal with advanced threats like TAs a good practice is the monitoring of crucial points of the enterprise network with probes, establishing useful data sources, which are basilar for any further analysis. Probes are an abstract concept which may be implemented in a huge amount of ways, an ad hoc piece software, a third party product with the ability to report events, network taps, application or operating system log daemons. The following non-exhaustive list summarize what kind of data should be monitored by a probe:

- Network Captures
- Contacted Hosts
- DNS request, network flows meta-data

[2]NGFs help to detect application-specific attacks, may integrate functionalities of classical signature based NIDS, could support SSH and SSL traffic inspection and provide basic malware filtering

- Received emails
- Web Application requests
- Artifacts accessed/modified by a host process (files, registry entries, other processes)
- USB device activities
- ...

Probes and sensors could not be raw data collectors only, higher level devices like (H/N)IDS[3][69, 3] can be included as *"sensors"* or *"data source"* with respect to the whole system.

3.3.3 Sandbox

Sandboxes are key technologies in the more general malware detection challenge, they can be informally defined as *"a way to separating running programs, contain their execution through a fully controllable environment"* and the good abstraction level of this definition may suggest wider applicative domain than the malware detection one. Sandboxing solution may be implemented via a rich set of techniques such as *jails*[64], *virtualization and emulation*[80, 12], *rule based execution*[13, 17.2.2], and so on.

In this section I will focus on virtual machine and emulator based sandboxing technologies, which have gained a central role in **malware dynamic analysis** automation and reached good maturity levels thanks to the interest of the *InfoSec* community.

Dynamic analysis is intended to be an investigation of the behaviour of a target binary at run-time, this analysis method is one of the principal techniques adopted in malware analysis due to the constantly growing cost of static analysis (ie. Reverse Engineering) caused by the increasing of the usage of anti-reverse and AV-evasion technologies such as *polymorphic* and *metamorphic* code, *packers* and *custom virtual machines.*

Stepping back to the more specific context of TA detection, dynamic analysis of unknown binaries plays a first class role because the kind of threat intended to block cannot be quickly detected via traditional approaches due to:

[3]Host based Intrusion Detection System and Network Intrusion Detection System

- possible *use of the technologies listed above for customizing known malware* (packer, VMs, metamorphism. . .)
- and possible *unknown nature* of the malware used in restricted attack campaign.

Having said that is pretty clear that having behavioral data about binaries and all executable artifacts in general may provide an useful information base for an attack detection.

3.3.4 SIEM

Another component adopted in many APT detection solutions is the *Security Information and Event Management* (SIEM)[73]. In the holistic approach all the data gathered by probes, sandbox, and sensors in general should be sent to this component which, if properly configured, has the capability to become *"the central nervous system of a network"*.

A SIEM technology may be configured and adopted by need, involving types of information relevant for the purposes of its installation, for this reason is quite common to find SIEM technologies both as stand-alone software and also integrated in specific defence tools. The main goal of a this technology is to handle information about *vulnerability management*, *intrusion detection*, *asset security assessment*, *asset management*, *patch management*, *configuration management*, and *incident response*, and *threat analysis.*
The amount of information types cited above should immediately stimulate a couple of issues in the reader mind, and consequentially should also suggest why several SIEM solution are integrated in defence tools:

- First issue merely regards the amount of data treated by this kind of systems which strongly brings to the fore the field of *Big-Data Analysis*[78] which rigorously treat this kind of problems.
- Second, the different quality of the information and the number of vendors and products capable to provide these kind of information is not negligible. In fact interoperability and the definition of a common ontology for SIEM technologies are some of the challenges addressed by security community and government institution[62](eg. MITRE).

B.t.w. typical SIEM technology should support the following capabilities:

- *Dashboards* tools
- *Alerting* and reporting support

- *Retention* if the data for a long term storage.
- *Data aggregation* and *correlation.*
- Support and *aid forensic analysis* by enabling smart navigation of the data.

3.3.5 Computer Security Incident Response Team

Despite the non software nature of the *CSIRT*, it still remains a key component of a holistic defence approach. The idea of an emergency response team is not new at all, in fact a first formalization of this concept was firstly developed and promoted through the *CERT/CC* (Computer Emergency Response Team Coordination Center)[18] after the *Morris Worm incident*[4] back in 1988, and just a year later a worldwide community of emergency response teams was established[34]. The acronym CSIRT has emerged due to a specialization of the IR-team in computer security incidents and for copyright issues too.

A good place to start understanding CSIRT is the *"CSIRT Handbook"*[59] which contains high level definitions and guidelines. It also provides general indications for setting up and maintain a CSIRT, in brief:

- CSIRT macro types and missions, for example a Corporate CSIRTs should focus threat minimization and intrusion containment missions.
- The relationship between CIRTs and the security team of the organization, by generally describing how integration should take place and also what kind of overlapping with IT security department is concerned.
- CSIRT services types:
  - *Reactive* such has reporting of compromised hosts or vulnerability assessment
  - *Proactive* like network hardening and threat minimization.
  - *Security quality management services* such as improvement of internal procedures or employee training.

However the handbook clearly warns the reader that definitions, services, policies and procedure valid for a CSIRT are not likely to be appropriate for another one[59, 2.1], so the authors do not recommend to blindly adopt their guidelines.

[4]The Morris worm was the first worm distributed over the Internet, which was capable to infect about 6000 computers, around the 10% of Internet population

3.4 Open Source Tools

The following paragraphs will provide some more concrete examples of software systems and solutions which can play a role in the TA detection challenge, characteristics of the tools listed below are summarized using the taxonomy introduced in 2.2. The list is not exhaustive.

3.4.1 Anubis Sandbox

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
Executable files	Anomaly based	Malw. Download	Report
Technologies	**Goal**	**Mainstays**	**Limitations**
Anubis Sandbox	Dynamic Analysis	different file formats and detailed behaviour log	sandbox detection techniques may be adopted (aka *red pills*)
Anubis Automated Clustering	Behaviour Classification	unsupervised approach	based on sandbox report

Table 3.2: Anubis[11, 12] (sandbox, malware detector) summary table

3.4.2 Cuckoo Sandbox

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
Executable files	Hybrid	Malw. Download	Report
Technologies	**Goal**	**Mainstays**	**Limitations**
YARA signatures	Static Analysis	ability to define ad hoc pattern	knowledge/pattern based
Virus Total integration	Static Analysis	exploit AV-industry knowledge	knowledge/pattern based

Table 3.3: Chuckoo Sandbox[67] (sandbox) summary table

3.4.3 Malheur

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
dynamic analysis report	Anomaly based	Malw. Download	Report
Technologies	**Goal**	**Mainstays**	**Limitations**
Automatic Analysis of Malware Behavior	Behaviour Classification	unsupervised techniques, clustering	rely on sandbox reports, that can be fooled

Table 3.4: Malheur [65](sandbox-report analyzer) summary table

3.4.4 OSSIM

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
traffic capt., alerts, events, info, logs, reputation data, . . .	Hybrid	-	Reporting, Alerts
Technologies	**Goal**	**Mainstays**	**Limitations**
Security sensors integration	Information Correlation	arpwatch, openVAS, ntop, NFDump, P0f (fingerprinting), Snort NIDS, Suricata IPS	-

Table 3.5: OSSIM [3](SIEM) summary table

3.5 Commercial Tools

The following list is a **non-exhaustive** list of commercial solutions which explicitly *claim to deal with APTs and TAs*, other software exists, but no review have been performed here because out of the purposes of this thesis.

3.5.1 FireEye Platform

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
Traffic capt., Web objects, Shares	Hybrid	DriveBy, Callback, Infection, Malw.Download, 0-Day	Host blocking, Alert, Reporting
Technologies	**Goal**	**Mainstays**	**Limitations**
Dynamic Threat Intelligence	Information Sharing	global threat meta-data sharing	-
Multi-Vector Virtual Execution	Dynamic Analysis	Multi-Path technology[60] for triggering hidden behaviours	detection techniques may still exists (eg. red-pills)
YARA rules customization	Static Analysis	fine grained and customizable signature detection	knowledge/pattern based

Table 3.6: FireEye Platform[33] (APT detection, SIEM, Sandbox, Network Sensor) summary table

3.5.2 LastLine Previct

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
Traffic capt., Web objects, mail, Shares, reputation data	Hybrid	DriveBy, Call-back, Infection, Malw.Download, 0-Day	Host blocking, Alert, Reporting, Evidence
Technologies	**Goal**	**Mainstays**	**Limitations**
Enhanced Anubis Sandbox	Dynamic Analysis, Behaviour Classification	sandboxing and unsupervised approach for behaviour classification	sandbox detection techniques may be adopted (aka *red pills*)
Bot Traffic Analyzer[55, 43]	Botnet Detection	supervised approach	performed during sandboxed execution, malicious behaviour may not be triggered
Shellcode analyzers[54]	Exploit detection	analysis and instrumentation of several file types	-
DriveBy analyzers[53, 52]	Exploit detection	sandboxing of web resources	-

Table 3.7: LastLine Previct (APT detection, SIEM, Sandbox, Network Sensor) summary table

3.5.3 FortiGate

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
traffic captures, web objects, application protocols	Hybrid	Malware Downloads, 0-Day	Drop malware, Alert, Reporting
Technologies	**Goal**	**Mainstays**	**Limitations**
FortiSandBox	Dynamic analysis	behavioural analysis and logging, able to retrieve objects from SSL connections	sandbox detection techniques may be adopted (aka *red pills*)
FortiGuard	Information Sharing	data gathered worldwide	knowledge based

Table 3.8: FortiGate[35] (NGF, sandbox, network sensor, APT detection) summary table

3.5.4 SourceFire FireAMP

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
traffic captures, documents, reputation data	Hybrid	Malwr.Download	Block host, Alert, Report
Technologies	**Goal**	**Mainstays**	**Limitations**
Cloud-based Sandbox	Dynamic analysis	behavioural analysis and logging, able to retrieve objects from SSL connections	sandbox detection techniques may be adopted (aka *red pills*)
Sourcefire VRT (Vulnerability Research Team)	Information Sharing, CSIRT	Data gathered worldwide, human analysts are contempt	-

Table 3.9: SourceFire FireAMP [20](APT detection, SIEM, sensors, sandbox) summary table

3.5.5 WebSense Web Security Gateway

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
network capt., web objects, mail, reputation data	Knowledge based	Malwr.Download, Dangerous Web Page	Blocking traffic, Reporting
Technologies	**Goal**	**Mainstays**	**Limitations**
Websense ACE (Advanced Classification Engine)	Threat Detection	classify binaries, web pages and mail, optionally support sandboxing, ssl inspection	based on knowledge and sandbox outcomes

Table 3.10: WebSense Web Security Gateway[79] (Advanced Web Proxy, sandbox) summary table

3.5.6 Damballa Failsafe

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
network capt., web objects, mail, reputation data	Hybrid	Malw.Download, Callback, Infection	Alert, Reporting
Technologies	**Goal**	**Mainstays**	**Limitations**
Behaviour-basad Profilers	Botnet Detection	detect anomaly in communications, frequency anomaly, p2p traffic, anomalies after downloads	-
Content-basad Profilers	Botnet Detection	Sandboxing, AV scanning, HTTP request structure analysis[63]	encrypted traffic may not be analyzed
Threat Intelligence	Information Sharing, Botnet Detection	reputation analysis of destinations, correlation with behaviours detected in sandbox logs	knowledge based

Table 3.11: Damballa Failsafe[24] (APT detection, SIEM, Sandbox, Network Sensor) summary table

3.5.7 Symantec Endpoint Protection

Information src.	**Detection appr.**	**Recognition cap.**	**Reaction cap.**
File and registry activities, process activities, web objects	Hybrid	Process blocking, Alert	
Technologies	**Goal**	**Mainstays**	**Limitations**
Symantec Insight	Dynamic Analysis, Behaviour Classification	sandboxing new binaries, classification based on malicious patterns	Knowledge/pattern based
Symantec SONAR	Dynamic Analysis, Behaviour Classification	use reputation information, runtime behaviour monitoring via process instrumentation	Knowledge/pattern based, may also inject code into monitored process user-space memory.

Table 3.12: Symantec Endpoint Protection (APT detection, behavioural AV) summary table

Chapter 4

Testing Methodologies

Before introducing the guideline to test and compare APT and TA detection solutions, I am going to discuss the methodological approaches to software system testing. First of all I'll review common testing methodologies in order to provide a clear conceptual testing framework introducing best-practice and standard testing approaches which have been consolidated during the years 4.1, then I will focus on testing of security systems, with particular emphasis to the Anti-Virus solution testing.

Testing this kind of system is not trivial at all and both AV industry and security community have spent lot of time and resources arguing about *"how to properly test AV solutions"*: several independent third party testers exists nowadays[23, 74, 29, 15] and also a standardization organization AMTSO (Anti-Malware Testing Standard Organization) have been created. In section 4.2 I will introduce principles, best practice, and also some of the latest issues raised by security communities.

4.1 Software Systems Testing

Software security testing is not a trivial task at all, difficulties are not strictly related to technicalities and *"hackish-stuff"*, sure they cannot be liquidated as details or trivialities, but there are other less visible issues to consider. Issues which may have heavy side effects on test results or even damage system under test itself. For this reason a software system test should be well-defined[1, 2.1] in several important aspects such as:

- *Asset involved*, *limitations*, and *rules of engagement*.
- *Engagement zone*, as the area around the asset involved, processes, security mechanisms or services built around them.

- Define *vectors* as the interaction between groups of assets (eg. between department A and department B) and which *channels* are used in interactions (Human, Physical, Wireless, Data Network, . . .)

- Determine what you want to learn from test and choosing proper *test type.*

The following paragraphs will briefly describe several common test types, where the different level of awareness of both target and tester will lead to the measure of distinct qualities of the target (and also of the tester too)[1, 2.3].

4.1.1 RedTeam Exercise

The tester engages the target with *full knowledge* of its processes and security, but the *target knows nothing of what, how, or when* the test will be performed. The true nature of this test is to *audit the preparedness of the target to unknown* variables and vectors of agitation.

4.1.2 War Gaming

The tester engages the target *with no prior knowledge* of its defences, assets, or channels. The *target is prepared* for the audit, *knowing in advance all the details* of the audit. This kind of blind audit primarily tests the skills of the tester. This is often referred as Ethical Hacking.

4.1.3 Black Box test

The tester engages the target with *no prior knowledge* of its defences, assets, or channels. The *target is not notified in advance* of the scope of the audit, the channels tested, or the test vectors. This audit (aka Double Blind) *tests the skills of the tester* and the *preparedness of the target to unknown variables* of agitation. Also known as Penetration test.

4.1.4 Crystal Box

The *tester and the target are prepared* for the audit, *both knowing* in advance all the details of the audit. This tandem-audit *tests the protection* and controls of the target. However, it *cannot test the preparedness of the target to unknown* variables of agitation.

4.1.5 Vulnerability Test

The tester engages the target with *limited knowledge* of its defences and assets and *full knowledge of channels.* The *target is prepared* for the audit, knowing in advance all the details of the audit. This audit (aka Gray Box) tests the skills of the tester. The nature of the test is efficiency.

4.1.6 Double Gray Box

The tester engages the target with *limited knowledge* of its defences and assets and *full knowledge of channels.* The *target is notified in advance* of the scope and time frame of the audit but **not** the channels tested or the test vectors. A double gray box audit *tests the skills of the tester* and the *target's preparedness to unknown* variables of agitation.

4.2 Security Systems Testing

After a brief review of the testing methodologies commonly adopted in real-world software systems, I am going to focus on a more specific area of the testing discipline: Security Systems Testing.

Typically, this area is populated with a various number of security related software and solutions, from the most classical Anti-Virus systems, to Advanced Thread protection systems, through Firewalls, NIDSs, and IPSs. The purposes of the tests performed over the years are several: *performance* evaluation, especially in network based products, *static* file *detection* capabilities, and also *heuristic and behavioural detection* capabilities (aka dynamic tests)[23]. In the following paragraphs I will show how the industry-world deal with this kind of test by reviewing guidelines, recommendation, and also some of the most challenging issues recently raised by both companies and standard organizations[32, 8]. Hopefully this may help the reader to place the concepts introduced in following chapters in a more general context.

4.2.1 Anti-Virus and Anti-Malware Products Testing

One of the first kind of security systems which have been tested are Anti-Virus solutions, this kind of test have been primarily driven by the AV industry, where numerous companies compete since the Internet boom. Traditionally AV products where tested in order to compare their performance and their capability to statically detect malicious files and one of the most typical standard sample used during the years is the `EICAR TEST FILE` [29]:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

This particular file is detected by almost all AV solutions and should trigger an ad hoc message. EICAR test file has been firstly created for enabling different kind of people to test AVs, finding valid malware samples in not trivial at all and may lead to several issues discussed in 4.3.1, then to enable users to check if their AV solution has been deployed correctly, and also for avoiding legal restrictions which can be encountered in some countries for the distribution of virus samples.

However, recent AV product tests have raised important issues regard the use of EICAR test file for malware detection capabilities evaluation[25], in fact the use of modified, obfuscated, packed version of the EICAR file led to flawed test conclusion due to the wrong presumption that general detection methodology can be deduced from the single case of EICAR detection: at the EICAR conference 2010 disclosed test results shown that the only cases where the test file have been correctly detected were when the file remained intact. Such changes constitute a seriously unreliable guide to detection performance and even worse, changes that *"resemble"* a malicious program has no value for testing purposes unless they really are - in any meaningful sense - malicious. For these reasons AMTSO (Anti-Malware Testing Standard Organization) suggest to consider the EICAR test file as *"EICAR installation check file"*[10].

At this point is pretty clear that modern Anti-Virus and Anti-Malware solution testing cannot be tackled with simple test files, a malicious program *sample-set* should be adopted in this kind of tests. AMTSO suggest a best-practice guideline[4] to inspire meaningful and valid test. It also explicates that the *static* malicious file detection tests are no more the best approach for testing security systems against modern threats and introduces/formalizes the concept of **dynamic test**, where samples are actually executed. This choice has drawbacks that should be handled carefully in order to avoid bias in test results:

- The execution of malicious code should be controlled carefully.
- The performance of the product is strongly *determined by the behaviour of the sample.*
- Sample behaviour is strongly determined by the *execution environment.* Virtualization technologies adopted, network connectivity and malware launch mode (eg. manual launch, drive by download, infected USB disks, ...) may introduce bias.

- Tester loose part of the control on the test due to possible interaction with external - non controllable - resources.
- Test may loose part of its reproducibility

Also the nature of the *"measures of success"* of the test is discussed by distinguishing between **detection** of the sample, **removal** of the threat, **persistence** of the malware after reboot, and **damage** performed during the test (eg. stolen or altered data). Some indications on dynamic test styles are also provided, which are valuable methodological contribution even in testing not only AV products:

One at time machines set up with a single product, the sample is introduced in the environment and then executed. The state of the system is monitored and analyzed in order to ascertain whether the malware was detected (and eventually removed). After the test the machines are restored to previous state and the test is repeated with next sample.

Many at time machines set up with a single product, but this time multiple malware samples are executed at the same time. This kind of test is more time-effective but also less precise.

4.2.2 Cloud-Based Products Testing

The review of defence tool discussed in 3.5 clearly shows an important trend of the defence systems: the great part of the detection solutions contempt a *cloud based* component on their architecture. This architectural detail cannot be ignored in modern security solution testing due to the heavy side effects it introduces in a testing session: first of all the strong dependency of the systems under test to external resources lead to a severe lack of reproducibility of the test itself.

Continuous updates, the reputation data, black-list, white-lists and threat data correlation cannot be frozen during a test session, the tester cannot simply unplug the network cable because these features are required in order to evaluate the security system at the maximum of its capabilities, for this reason the tester should accept the *loss of reproducibility* introduced and tune-up the testing methodology for ensuring a fair, unbiased test. Pragmatical clues come from ATMSO[7], which suggest to:

- Perform tests of cloud-based products *parallelly* in order to minimize the risk of a bias in favour of the last system under test.

- *Split test* in several test-sessions for minimizing advant ages related to the receiving of fresh updates of the slower detection system, in other worlds the tester should try to ensure the similar condition to all the evaluated products. Of course, this recommendation does not worth in case of speed of detection or throughput performance evaluations.

- Carefully implement the *"first do no harm"* principle, in fact a too restrictive filtering of network traffic could debar both the functionalities of the system under test and also the behaviour of the malware samples producing biased test-results.

- Assess the performance of the internet connection and consider it as a part of the the testing environment.

Network Based Products Testing

Some consideration could be performed even for network based products like IDS, IPS and Next Generation Firewalls too. Tests on these categories of system should be performed by considering both software functionalities like the quality of the application protocols supported (HTTP, CIFS, SMB, mail protocols, SSL, SSH, ...) and also hardware characteristics, in fact deployment of these systems may affect the performance of entire parts of the network, for these reason *"stress tests"* are welcome in order to assess the efficacy of the solution under heavy loads.

4.3 Malicious Samples

A Few paragraphs above I discussed about the huge amount of details that a tester should consider during a security product testing session, I deliberately skipped the issues related to the *sample-set* adopted because I think it's a central point in the testing process, especially when dealing with modern security solutions which claim to be able to counterstrike the advanced attacks that characterize the modern threat landscape.

In previous chapters I have shown the main trends in underground malware development (continuous repacking, mutations, exploits, ...) and also introduced issues regarding non-repeatability of a meaningful test, now I am going to discuss the role of the sample selection in this delicate testing process.

4.3.1 Sample Selection and Validation

In [5, 9] the sample issue is treated with a particular emphasis on modern system testing, these documents provide valuable guidelines regarding the quality of samples, which is gaining a central role and sometimes is also considered more important than the quantity. Quality of samples could be reached by a proper sample selection process that can be summarized in three macro phases:

Collection is the first problem that a tester will have to solve, in fact the source of samples should be carefully chosen considering the designed context of the test (corporate oriented or also consumer oriented). Also is valuable to categorize samples from two different points of view:

- How and where the sample were collected, like *honey-pots*, *crawlers*, *ISP*, *malware collections*, ...
- Where the samples have been collected from, for instance it's interesting to know if a malware sample has been gathered from a internet URI, a phishing email, file sharing platforms, social networks, intranet, ...

However, the ideal source of samples should offer **fresh**, **prevalent** and **diverse** real-world samples. The diversity of the samples should minimize the bias introduced by a hypothetical specialization of a AV company in dealing with some particular malware family, freshness will help to better simulate a real-world scenario, where newly distributed malware may be unforeseen, and prevalence should ensure to focus on meaningful samples by controlling the amount of gray-ware samples, that could trigger false positives.

Validation consists in a series of test performed on the samples. The goal of this phase is to ensure the malicious nature of the samples which is not trivial because a simple AV scanning test may not be reliable due to the risk of erroneous detection produced by some too much aggressive detection rule. For this reason different tools and approaches could be adopted:

- Dynamic analysis of the sample by running it in sandboxes
- Monitoring execution using system tools, logs, rootkit and network monitors.

- Executing malware in a passive environment, then turn off and investigate difference between pre-execution system image and post-execution image, searching for file system difference, boot sector and registry modification.
- Reverse engineering the sample
- During 0-Day detection product testing ensure that exploit sample works in target machines.
- Check the validity of the contacted URL at time of test.

Also the sample should be verified to be working in the testing environment, a corrupted sample or a not loadable sample may spoil test results.

Classification of the collected samples by labelling them as *good*, *malicious*, or also *gray* when the maliciousness may depend on the intent of the author or by the understanding of the target user.

Gray-Ware sample issues

Spyware, adware, toolbars and other unwanted software are commonly referred as gray-ware due to the annoying nature of their purposes. Performance of detection of this kind of software may vary from security solution to security solution and is in general driven by the reputation assigned by the security product developer rather than from detection capabilities.

For this reason evaluation of grayware detection performance should be done with particular care to the target user perception of this kind of software and should also consider *geo-location* of test scenario, for instance a false positive detection on a chinese toolbar may not be critical meanwhile the same false positive in a chinese testing scenario may be considered critical.

4.3.2 The Sample Creation Debate

From long time members of the anti malware community debate about the creation of new sample for testing purposes. Argumentations does not only regard drawbacks of the adoption of this kind of samples in a test session, but also regards the nature of the concept of *"sample creation"*[8].

This issue could be addressed in a very philosophical way by firstly discussing about what kind of elements are required for identifying a new version of a malicious program, it may seem trivial but the first impression could be faulty because different point of view and abstraction levels can be legitimately adopted to distinguish malicious programs. For example considering the inference *"Executable bytes and binary instruction are changed, this implies the existence of a new variant of the same malware"* in the context of a packed malware is correct, but is also correct to say *"Malware is more than a sequence of instruction"*. It depends on the features you want to describe.

Creation may be achieved in different ways, even the use of compressed archives, self-extracting archives and installers could be considered a sort of sample creation, in fact these kind of technologies may add functionalities to the malware itself. For example the use of a silent installer software may help to keep infection persistent despite the lack of this feature inside the real malware payload, or also a self extracting archive could be able to copy malware in specific locations and create links to the extracted file. All these features can be added without a real modification of the malware payload itself. Analogous consideration could be performed when dealing with packers and virtual machines: they typically add anti-reverse, anti-debug, anti-dump features to the malware. Creation also happens when the malware sample is generated using a *malware generation kit*, extremely common tool in real-world attacks. This enables testers to generate arbitrary malware variants. Of course, writing new samples using previously *known* and *unknown* techniques is also a sample creation case.

No official position about the sample creation practice has been taken by the anti-malware community, for this reason I will report here some of the key discussion about the motivations of this particular testing practice:

Testing Heuristic/Proactive capabilities without signatures is an important aspect claimed by several anti-malware solution and its testing is obviously valuable. In this context, sample creation may be used for guaranteeing there is no signature involved in the detection, permitting the tester to focus on other detection capabilities. Nevertheless, part of the anti-malware community advice to use fresh samples from the field because they better represent the real-world trends in malware development.

Test support for specific packers tester may decide to pack known malware payloads with a particular kind of packers in order to assess how the anti-malware product handle this threat. Again part of the community continue suggesting to use fresh samples packed with packers beyond question, but this time the gathering of specifically packed samples may not be easy.

Keep test focused Focusing a test on certain type of threat may be eased by the adoption of created sampled, especially when dealing with systems which claims to deal with a particular kind of threats due to possible difficulties in gathering different samples of the same type.

Improving independence Samples used during tests may be *known* to some vendors even without any maliciousness. If the source of the samples is shared between both testers and one of the anti-malware companies involved in the test a bias on result could be introduced. In this context, creation of samples may help tester to gain independence with respect to tested companies. The part of the anti-malware community which disagree with sample creation suggest to obtain samples from independent third-parties, but this may not solve the problem because same third parties shall be used by anti-malware companies too.

Chapter 5

TA Detection System Testing

In this chapter I will introduce a guideline for testing security systems with particular attention to their capabilities to detect targeted attacks, typically characterized by the *unknown nature* of the intrusion. The nature of the threat suggests to keep the inspiration from *Double Gray Box* testing methodology in order to sense the preparedness to unknown variables of agitation.

Advanced malware detection does not exhaust the conceptual domain of the TA detection, despite it represents an important part of this type of attacks, other aspects should be evaluated in order to perform a complete test of a TA defence system as the *reconnaissance detection*, or also the *command and control communication detection.*

For this reason the *TA kill-chain* introduced in 1.3 is considered as main track to follow in this methodology: TA kill-chain is an attack oriented model which define what kind of actions are required in order to achieve a successful intrusion, looking at this model from a defense oriented point of view provide valuable indications about how to define and design test sessions, so an exhaustive test on security systems that claim to be able to tackle TA should cover all main attack steps reported by the kill-chain model.

Vision

The reader must always remember what is the main vision of this test methodology: *testing for real, modern threats.*
This can be better achieved adopting ***real-world test-bed environment*** that permits to not just evaluate the efficacy of the detection in realistic conditions, but also the scalability of the system under test w.r.t. realistic workloads. Obviously, test-bed deployment requires more efforts and care.

5.1 Testing the kill-chain

Having said that, is clear the whole test could be split in **several sub-tests**, each one *with distinct purposes*, for example the *reconnaissance sub-test* should be designed for assessing the capability of the security system to detect anomalies related to the information gathering phase of the attack (eg. network scan, web resources scraping, or other suspicious almost-legal activities) meanwhile the *malware delivery sub-test* should be focused on detection of weaponized malware. Following paragraphs provide valuable references for setting up meaningful sub-tests in order to cover major steps of the attack model.

5.1.1 Information Gathering

The retrieval of information about the target is one of the mainstays of a modern targeted intrusion, the capability to detect attack attempts in their earliest phase is really valuable because it should bring target organization to an alert status, minimizing both the risk of a successful attack and the time available to the attacker for performing malicious actions.

Testing products contextually to the *reconnaissance* attack phase is not trivial and require a proper definition of the concept of *"Samples"*, test designer should no more intend it as *"malicious executables file"* or *"artifacts"* in general, he/she has to go back to a more abstract definition, treat it as *"a subset of a population"* and then investigate what kind of population is involved and how to define a meaningful sample in the information gathering context.

An hypothetical sub-test aiming to measure reconnaissance detection performance could involve network monitoring products or even web analytics solutions accompanied with some detection engine. In this case, meaningful samples could be defined as **network traffic samples** where benign samples are obtained by replaying known neutral traffic or generated using traffic models [2][4], meanwhile malicious ones may be generated by effectively gather sensitive information from the test-bed, for example by scraping web resources or fingerprinting network devices.

5.1.2 Weapons Delivery and C2

Delivering malware is a crucial point in the TA model, this attack step could decree the success or the failure of the attack attempt. For this reason, assessing the performance of the detection of advanced, customized and weaponized malware may represent ***the core of the test***. This part of the test should be designed with TA in mind, trying to define meaningful samples for measuring performance against targeted-malware. Tester should ensure to collect good quality samples, trying as hard as possible to prepare a set of samples *not previously known to the detection system.*

Malware delivery detection can be interpreted as the detection of the act of spreading malware with the purpose of compromise hosts inside target network, delivery may happen via several distinct *vectors*: some of them are observable via network monitoring, like *drive by downloads*, *phishing emails* or also *misleading executables*, other may not require network interactions, like *USB sticks*, and also vectors involving observable channels exploited in time and places where no monitoring is possible, perfectly reasonable assumption especially considering *BYOD* diffusion and external contractors.

The possibility of unobservable delivery in a real-world scenario suggest the tester to include ***sample execution*** in test session, this way outcomes of the test will provide feedbacks on detection system capabilities to detect *command and control* channels established by the malicious samples. Note that the channel establishment introduce non-reproducibility in the test session due to dependencies from uncontrolled external resources, for this reason the sample selection must be performed carefully and a functionality check of samples must be performed right before test session start.

Summing up, a test session designed to measure detection of weapon delivery has to contempt *dynamic testing* of carefully selected, recently validated samples, no particular constraint about sample execution scheme - ***one at time***, ***many at time*** - but as reported in 4.2.1, tester should be aware of the likelihood of interferences between samples during a multiple execution.

Testing the Unknown

Selecting samples accordingly to the goal of this sub-test is not trivial. By definition a targeted attack involves advanced, sophisticated and customized malware and concept of ***new/unforeseen/unknown*** is intrinsic when the goal testing systems against TAs or APTs. At this point *Sample Creation* should be seriously considered in order to better test systems with respect to TAs. For this reason the sample-set may include *fresh*, *unforeseen variants/builds* of known malware families, *customized* malware generated using *bot-kits*, and also *ad hoc malware* developed for testing purposes.

Particular attention must be paid when creating ad hoc samples because they *have not to resemble* a malware, they have to implement features which potentially enable them to take control of the infected machine. Evasion techniques are also one of the mainstays of modern malware, so a test designer which decide to create samples should consider to equip them with this kind of technologies.

Creating ad hoc samples inevitably require more skills and efforts to the tester, but also helps to ***increase the reproducibility*** of the test by eliminating external, uncontrollable factors in favor to the adoption of fully controllable objects, sensibly lowering the risk of violation of the *"do not harm"* principle.

5.1.3 Actions

Actions and lateral movements performed by the attacker after a successful infection are typical during a targeted attack and their detection represent the last opportunity to stop the attack attempt before the actual damage. Setting up a meaningful test for measuring detection performance of this particular attack step could be harder than expected because malicious actions can be performed *even with legit and native tools.*

A brief categorization of this kind of actions with some examples of tools/artifacts that could be involved:

- ***Local Actions*** software and commands run in a host, eg. *cmd.exe*, *reg.exe*, *at.exe*, *psexec*, *bitsadmin*, *netstat*, any other program used in a production environment, or also local resources accessed like files, folders and USB devices.

- ***Network Actions*** like network-share access, remote desktop sessions, telnet sessions, network scans, arp spoofing, ftp sessions, intra-net services sessions, mail service access,...

This represents the population where to pick samples. A malicious sample could be defined as ***sequence of actions*** that can be ***manually*** or ***automatically*** performed by an attacker with *explicit malicious* ***intent***, here the tester should have a clear idea of the meaning of the sample he/she is defining and should also be aware of *post-exploitation* and *lateral movements* patterns and objectives.

Defining benign samples can be even more challenging because a *"benign sequence of actions"* is strictly related to the ***role of the host***: running *netstat* or *psexec* tools on the sys-admin machine is more likely to be legit than running them on a clerk's one. For this reason tester must carefully define the context - hopefully accordingly to the *expectation* formulated after the analysis of the products involved 5.2 - of the test and may also decide to adopt real users for producing real benign samples.

5.2 Testing Apples to Apples

According to AMTSO principles[6] particular attention should be paid when defining test session to the *type of security solution considered*, it matter and must be properly evaluated in order to avoid irrelevant, misleading test session like testing malware detection technologies in the context of information gathering. For this reason, in case of testing single specialized products the whole killchain test should be reduced and focused on sub-test related to part of the attack model concerned with the object under test.

In this context ***expectations*** regarding the security solution should be investigated and defined: in other words a preliminary analysis of the products must be performed in order to enable or disable proper sub-tests and figure out what kind of ***channels*** and ***artifacts*** can be analyzed.

5.3 Testing Systems

As seen in 3.2 a working approach for TA detection is the holistic one. It is characterized by the adoption of several security products from distinct vendors, or also from the same vendor, so concerning the case of a vendor providing both specialized security tools and integrated solution using all of them is perfectly sound.

For this reason the testing methodology must also support scenarios where a suite of product is adopted by moving the test goal from *"test TA detection product"* to *"test TA detection system"*, the results of the test designed for the latter goal reveals useful measurement of the effectiveness of the detection system composed by the union of products involved in the test.

When espousing this goal, single products' performance will contribute to the performance of the whole TA detection system, which is object under test. However, when testing systems composed of different component, test designer must also ensure to be able to monitor performance of each part involved and should be able to attribute a *Pass/Fail/NA* judgment for each component deployed.

Valuable information may emerge observing components results in relation to their expectations. A *not applicable* (NA) judgment may help to avoid result bias during sub-test where the particular configuration of the system does not make possible to determine the performance of a component. A basic example scenario could be a system composed by two components in cascade where a benign sample is erroneous blocked by the first one, preventing the second one to analyze it.

Nature of Tested Objects

A well-designed TA system test must consider main characteristics of the object under test, security systems have become more sophisticated and complex, as briefly reported in 3.3 cutting edge technologies frequently rely on *cloud based*, global wide intelligence support forcefully bringing the ***Internet inside test-beds***. Inevitably the *test reproducibility is affected* and ***test timing*** become really important in order to avoid result biases.

5.4 Benchmarking

Comparison between products and solutions is one of the main motivations of real world test sessions and a complete testing methodology must properly tackle comparative tests. According to previous considerations about the nature of objects under test, test session must be performed **at the same time** for each TA detection system/technology.

This design constraint has important ***side effects during test-bed design and setup***, in fact setting up an unbiased test-bed will require further efforts: test-bed designers must ensure that each system deployed in the test-bed will receive the *same stimuli during the session*, roughly speaking they should be potentially able to percept *"the same things"*. Not trivial at all when the TA detection systems compared are composed by several interacting parts, each one with different deployment requirement. Thinking at test scenarios involving products with reaction capabilities clearly raise the problem of ***interference between systems***. For example consider the following scenario:

- Tested products are network based
- One of the them try to terminate suspicions network flows in a reactive manner, for example by flooding infected machine with forged TCP RST segments after its detection.
- The detection capabilities of this "reactive product" are less effective than other ones but its average detection/reaction time is lesser that others, formally:

$$detect_{reactive}(samples) \subset detect_{other}(samples) \ ,$$
$$s \in samples \ ,$$
$$\|neededStimululi_{reactive}(s)\| < \|neededStimululi_{other}(s)\|$$

Where $detect_{system}(samples)$ is the set of malicious samples detected by a system under test, $\|neededStimululi_{system}(sample)\|$ represent the amount of stimuli needed in order to take a decision about the current malicious sample

With this assumption the "reactive product" could interfere with the other ones avoiding further stimuli to reach other products, preventing them from correctly detect the malicious activity even if their real detection capabilities are more effective. Completely different story if the test designer is highly interested in performance, he could explicitly decide to consider delays equivalent to a failed detection, here this kind of interference between systems can be considered negligible.

Having said that is pretty clear that interference between systems should be carefully investigated and successively evaluated accordingly to the test goal.

5.5 The Whole Picture

The diagram in figure 5.5 represents the workflow of the TA detection system test guideline discussed in previous sections adding emphasis on the relationship between activities and artifacts.

Five principal macro-activities emerge from the description of the methodology already discussed:

Preliminary Activities investigation of the features of the systems, with the goal to provide *expectations* about what should be detected and indications about what kind of test perform.

Samples Activities providing a sample set according to *expectations*, always considering the peculiarities of targeted attacks.

Test-bed Activities setting up test-bed according to *expectations* and investigating possible interferences between systems.

Test Activities run test using provided samples and monitor performance of each component of the systems, provide a *Pass/Fail/NA* result for each sub-test performed.

Result Analysis Activities Evaluate results and formulate considerations about performance of the systems and contribution of each component.

According to these macro-activities is possible to define four test teams:

Analysis team in charge of preliminary and result analysis activities.

Sample team in charge of sample related activities, may help the *Test team* during functionality check of the samples.

Test-bed team in charge of designing and deploying systems and test environment.

Test team in charge of running sub-tests with respect to timing constraints and monitor each component of the systems.

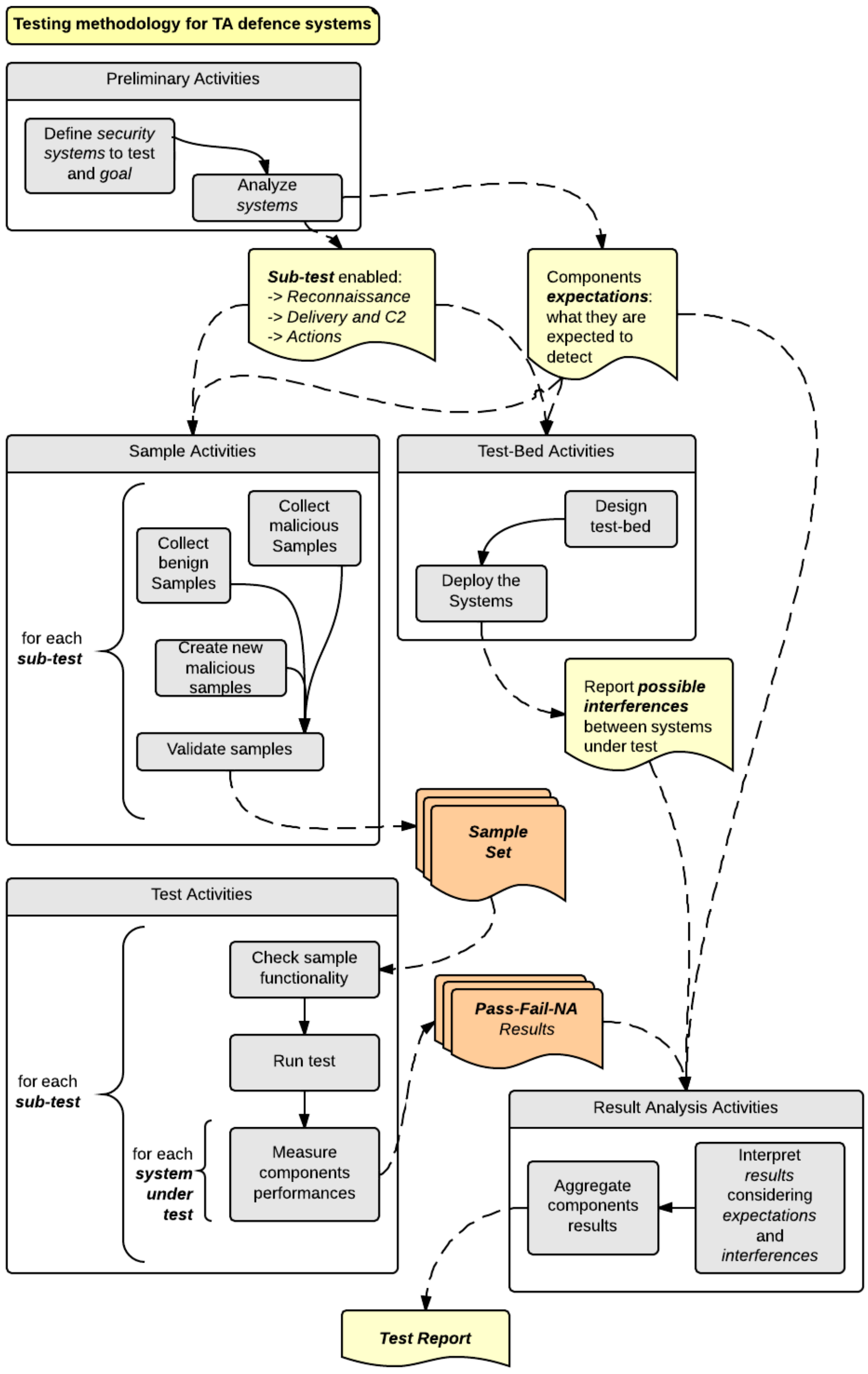

Testing methodology for TA defence systems
Preliminary Activities
Define security systems to test and goal
Analyze systems
Sub-test enabled:
-> Reconnaissance
-> Delivery and C2
-> Actions
Components expectations: what they are expected to detect
Sample Activities
for each sub-test
Collect malicious Samples
Collect benign Samples
Create new malicious samples
Validate samples
Test-Bed Activities
Design test-bed
Deploy the Systems
Report possible interferences between systems under test
Sample Set
Test Activities
for each sub-test
Check sample functionality
Run test
for each system under test
Measure components performances
Pass-Fail-NA Results
Result Analysis Activities
Aggregate components results
Interpret results considering expectations and interferences
Test Report

Figure 5.1: Diagram representing activities and artifacts in the proposed methodology

Chapter 6

Case Study: Testing Two TA Detection Systems

Part of the work of this thesis regarded the set-up of a comparative test of two TA detection systems adopting the methodology introduced in chapter 5. The performed tests will be described with emphasis on activities and artifacts that characterize the proposed methodology.

6.1 Preliminary Analysis

In this context *analysis-team* firstly reviewed systems under test and then report that all of them are composed of the following types of defence tools: *"Next Generation Firewall"*, *"Advanced Application Proxy"*, *"APT Detection Product"*[1]. Despite the similar architecture, different APT detection products characterize these two configurations.

External connection should be concerned in *test-bed design* due to the strong dependencies of the components from cloud-based resources, another constraint emerged from this consideration regards the timing of the test: *test-bed team* must ensure to design the test-bed in order to enable the *test team* to perform required test sessions at the same time for each system under test.

In the following section the first TA detection will be referred as **SYSTEM-1**, the other as **SYSTEM-2**, next generation firewall as **NGF**, advanced application proxy as **APROXY**, **APT-DET-1** APT detection product in *SYSTEM-1* and **APT-DET-2** for the other one.

[1]The reader can refer to tools reviewed in 3.3.

Expectations

Preliminary analysis performed by *analysis-team* was based on the features of the components emerged after their categorization through the taxonomy described in chapter 2.2: both systems should be able to properly detect *malware downloads*, *drive-by-downloads* and *malicious communications* with particular emphasis on *HTTP* based communications, for this reason the *sample-team* and *test-team* will have to focus on this kind of channel.

Another consideration regards which kind of *sub-tests* perform: systems under test do not cover all attack steps typical of TAs, for this reason we limit our test to the *malware delivery* and *command and control* sub-test.

6.2 Test-Bed

Test-bed team deployed the systems in an *enterprise context* where over a thousand of benign/sane hosts were performing their common operations ensuring an absolutely realistic test-bed. In figure 6.2 is represented the network architecture adopted for the comparison test.

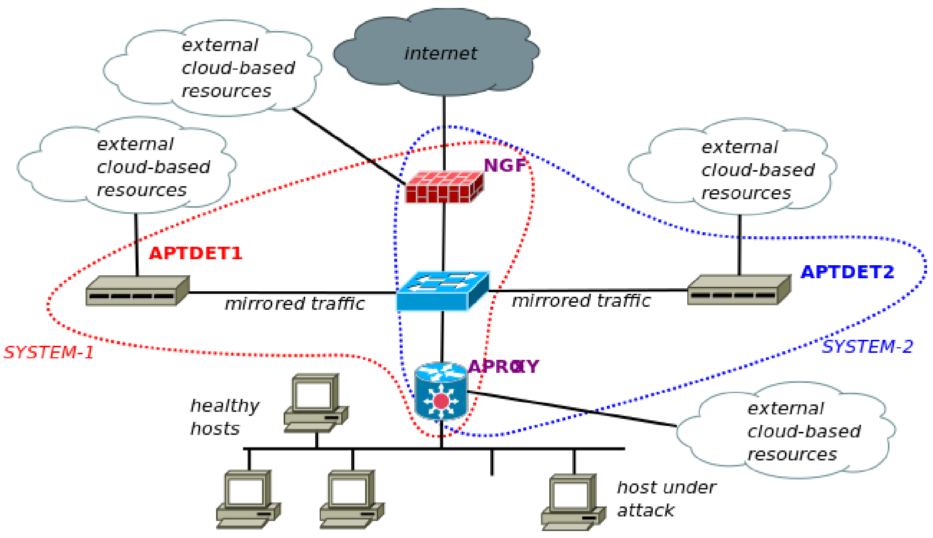

external cloud-based resources
internet
external cloud-based resources
NGF
external cloud-based resources
APTDET1
APTDET2
mirrored traffic
mirrored traffic
SYSTEM-1
SYSTEM-2
APROXY
healthy hosts
external cloud-based resources
host under attack

Figure 6.1: Network architecture of the deployed test bed.

Interferences

During the deployment of the systems, *test-bed team* payed particular attention regards possible interferences between **APT-DET-1** and **APT-DET-2**, they both contempt the possibility to activate *reactive countermea-*

sures such as connection reset, for this reason both product configurations deployed and network architecture adopted have been studied in order to avoid interferences by disabling features and working with mirrored traffic.

Another issue raised from *test-bed team* regards the blocking capabilities of both the *NGF* and the *APROXY* components, which should be concerned by *test team* during their measurement.

6.3 Samples

Sample team decided to prepare the sample set using both collected samples and ad hoc created samples.

6.3.1 Collected Samples

Meaningful malicious samples cannot be gathered from malware collections for two principal reasons: consolidated *knowledge* of the sample from AV industry and the high risk of *sink-holed/not-working command and control* services. For these reason *sample team* decided to collect samples directly from their distribution sites.

After a brief analysis of the life-cycle of malware gathered from distribution sites, *sample-team* estimated a delay of about one working day between the first seen of a malicious executable and the its *"signaturization"* by AV-Industry. During this time span the sample could be considered *new/unknown/unforeseen.*

Maintaining a sample set of unknown samples is not trivial because of the limited amount of time needed for its usage, for this reason a two phase ***sample collection strategy*** has been adopted:

1. Same week of the test day:
   - Collect samples from distribution sites.
   - Validate samples and conserve most interesting ones.
   - Take note of samples metadata like file name or ***icon***.
2. Few hours before test:
   - Collect latest samples from distribution sites.
   - Search executables with similar metadata to the samples previously gathered, eg. matching ***icon***, similar names, ...

- Quick re-validation of the samples in order to check their functionalities.

Modern malware heavily use polymorphism and metamorphism and new different *versions* of the executables are packed every day, but an attack campaign from the same *"authors"* last longer and other features like metadata and icons do not change too much. This fact provides a valuable help in reducing the search space associated to the attempt to figure out if the sample under analysis belong to the same family of others previously collected.

At the end of the collection activities the sample set was composed of 5 types of malicious program - *Bot*, *Trojan-Spyware*, *Downloader*, *Worm*, *Trojan-Backdoor* - grayware software and benign executables.

Validation

Collected samples have been validated using the following procedure:

1. Automated dynamic analysis via Sandox technologies.
2. Monitoring network, file-system and registry activities during a manual infection session on virtual machine.
3. Outcome interpretation and attribution to generic malware family.

A complete reverse engineering have been discarded because of the non reasonable time needed with respect to the validity of the sample.

6.3.2 Created Samples

In order to raise the bar of the test and better measure the attitude of the systems under test to detect an unknown advanced threat, *sample team* has contempt the *creation of a new ad hoc advanced malware* capable to get ***"hands on keyboard"*** access to the infected machine.

Like other advanced threats, the created sample has been equipped with some evasion techniques in order to better simulate the intent of the attacker to bypass defences.

Particular care has been taken in order to not simply mimic or theoretically enable malicious behaviours, in fact the rouge software was programmed to behave as a *Trojan-Spyware* by automatically log all the keys pressed to the *command and control server.*

6.4 Running Test: Malware Delivery and C2

According to the two step validation introduced in 6.3.1, a few hours before test beginning samples have been re-collected and validated to make sure they are still meaningful.

Malware delivery and C2 test have been split in two sequential phases: the *delivery detection phase* where malicious samples are downloaded by an internal host via *HTTP* channel, and the *command and control detection phase* where samples have been run on the internal host independently from delivery results. The chosen test type was *one at time*.

Test team has monitored defence tools' consoles and assigned *PASS* judgment in case of detection of malicious sample or no detection of benign sample, *NA* in case of lack of information due to components properties reported by *test-bed team*, *FAIL* otherwise.

	NGF	**APT-DET-1**	**APT-DET-2**	**APROXY**	**SYSTEM-1**	**SYSTEM-2**
Bot	*PASS*	*NA*	*NA*	*NA*	*PASS*	*PASS*
Trojan-Spyware	*PASS*	*NA*	*NA*	*NA*	*PASS*	*PASS*
Downloader	*PASS*	*NA*	*NA*	*NA*	*PASS*	*PASS*
Worm	*FAIL*	*PASS*	*PASS*	*FAIL*	*PASS*	*PASS*
Trojan-Backdoor	*PASS*	*NA*	*NA*	*NA*	*PASS*	*PASS*
Ad hoc Trojan-Spyware	*FAIL*	*FAIL*	*FAIL*	*FAIL*	*FAIL*	*FAIL*
GrayWare 1	*FAIL*	*PASS*	*PASS*	*FAIL*	*PASS*	*PASS*
Benign Samples 1	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*
Benign Samples 2	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*
Benign Samples 3	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*

Table 6.1: Pass/Fail/NA result of malware delivery phase

	NGF	**APT-DET-1**	**APT-DET-2**	**APROXY**	**SYSTEM-1**	**SYSTEM-2**
Bot	*FAIL*	*PASS*	*FAIL*	*FAIL*	*PASS*	*FAIL*
Trojan-Spyware	*FAIL*	*PASS*	*FAIL*	*FAIL*	*PASS*	*FAIL*
Downloader	*FAIL*	*FAIL*	*PASS*	*FAIL*	*FAIL*	*PASS*
Worm	*FAIL*	*PASS*	*PASS*	*FAIL*	*PASS*	*PASS*
Trojan-Backdoor	*PASS*	*NA*	*NA*	*PASS*	*PASS*	*PASS*
Ad hoc Trojan-Spyware	*FAIL*	*FAIL*	*FAIL*	*FAIL*	*FAIL*	*FAIL*
GrayWare 1	*FAIL*	*FAIL*	*PASS*	*FAIL*	*FAIL*	*PASS*
Benign Samples 1	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*
Benign Samples 2	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*
Benign Samples 3	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*	*PASS*

Table 6.2: Pass/Fail/NA result of command and control phase

6.5 Results

Tables 6.46.4 report the results of the *malware delivery and command and control* sub-test showing which components have passed or failed the test. With this information *analysis team* has formulated the following consideration regards test results:

- *SYSTEM-1* performs slightly better due to relevant contribution of *APT-DET-1* in detecting command and control communications.
- *SYSTEM-1* and *SYSTEM-2* have not detected the custom malware which *represent an unforeseen attack-tactic.*
- Tested TA detection systems do not show issues related to *false positive.*

These consideration are adequate for providing useful advice to the buyer of the test.

Chapter 7

Conclusions

Proper testing methodologies for modern defence tools is a key point in the battle versus advanced cyber-attacks that are scourging IT infrastructures all around the world. Understand the effectiveness of adopted countermeasures give precious feedback to both organizations and vendors, and represent an indispensable requirement for increasing the awareness of modern threats and better evaluate risks and opportunities.

Literature and guidelines on testing methodologies available at the time of writing does not cover sufficiently the peculiarities of targeted attacks, a great amount of documents about the testing of security systems is malware centric, leaving a gap between what is measured and what actually happens.

This work aims to provide indications and guidelines for preparing meaningful test for security systems and defence tools with particular emphasis on modern attacks: Targeted Attacks. Proposed methodology has been designed to deal with the most important attack steps that characterize this type of threat, it provides a schematic workflow, activities description and team-oriented organization of the testing process.

A real-world test has been successfully conducted using methodology introduced in this work providing precious feedbacks for better compare two TA detection systems with emphasis on attacks that are really capable to significantly damage organizations, showing the applicability of the provided guideline and its effectiveness in filling part of the gap between what we actually test and what we actually tackle.

Appendix A

Clarifications

A.1 Developed Tools

During the preparements of the test reported in 6 several tools have been developed like the *Malicious Sample Scraper*, sample upload automation scripts and *the created malicious sample.* These instruments cannot be attached at this document due to their development context: this thesis was developed during my internsiph and I was asked to avoid disclousure of these tools.

A.2 Products and Test Results

Test reported in the case study has been setted up in the context of my intership and test-bed was actually deployed in an important enterprise environment. Names of the products involved in this test have been explicitely avoided due to the confidential nature of the test. However the purpose of this document is **not** to compare products, it's rather the definition and the experimentation of a methodology for better fill the gap between what is tested and what is actually tackled.

Bibliography

[1] OSSTMM - Open Source Security Testing Methodology Manual. April 2009.

[2] 3rd Generation Partnership Project 2. Cdma2000 evaluation methodology, 2004.

[3] AlienVault. Ossim. http://en.wikipedia.org/wiki/OSSIM, 2013.

[4] AMTSO. Amsto best practice for dynamic testing, 2008.

[5] AMTSO. Amtso best practices for validation of samples, 2008.

[6] AMTSO. The fundamental principles of testing, 2008.

[7] AMTSO. Amtso best practices for testing in-the-cloud security products, 2009.

[8] AMTSO. Amtso issues involved in the "creation" of samples for testing, 2009.

[9] AMTSO. Amtso sample selection for testing, 2012.

[10] AMTSO. Amtso use and misuse of test files, 2012.

[11] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, behavior-based malware clustering, 2009.

[12] Ulrich Bayer, Christopher Kruegel, and Engin Kirda. Ttanalyze: A tool for analyzing malware, 2006.

[13] Matthew A. Bishop. *The Art and Science of Computer Security.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[14] BitDefender. B-have the road to success. http://www.bitdefender.com/files/Main/file/B-HAVE_The_Road_to_Success.pdf, October 2013.

[15] Virus Bulletin. Virus bulletin: Covering the global threat landscape. http://www.virusbtn.com/, 2013.

[16] Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-per-install: the commoditization of malware distribution. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, pages 13–13, Berkeley, CA, USA, 2011. USENIX Association.

[17] E. Casey. Determining intent - opportunistic vs targeted attacks. *Computer Fraud and Security*, 1(4):8–11, 2003.

[18] CERT/CC. Cert coordination center. http://www.cert.org/, 1989-2013.

[19] Eric Chien and Péter Ször. Blended attacks exploits, vulnerabilities and buffer-overflow techniques in computer viruses.

[20] Cisco. Sourcefire products and solutions. http://www.sourcefire.com/, 2013.

[21] Frederick B Cohen. *Protection and security on the information superhighway*. Wiley, New York, NY, 1995.

[22] Michael P. Collins, Carrie Gates, and Gaurav Kataria. A model for opportunistic network exploits: The case of p2p worms. In *WEIS*, 2006.

[23] AV Comparatives. Indipendent test of anti-virus software. http://www.av-comparatives.org/, 2013.

[24] Dambrella. Failsafe. https://www.damballa.com/solutions/damballa_failsafe.php, October 2013.

[25] Harley David, Willems Eddy, and Myers Lysa. Test files and product evaluation: the case for and against malware simulation in testing, 2010.

[26] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31(8):805–822, 1999.

[27] D.E. Denning. An intrusion-detection model. *Software Engineering, IEEE Transactions on*, SE-13(2):222–232, 1987.

[28] Nitesh Dhanhani, Billy Rios, and Brett Hardin. *Hacking, The Next Generation*. O'Reilly Media, first edition, 2009.

[29] EICAR. European institute for computer antivirus research. http://www.eicar.org/, 2013.

[30] Verizon Enterprise. 2013 data breach investigations report. Technical report, Verizon Enterprise, 2009.

[31] Verizon Enterprise. Veris framework. http://www.verizonenterprise.com/resources/whitepapers/wp_verizon-incident-sharing-metrics-framework_en_xg.pdf, October 2013.

[32] FireEye. Be the change, test methodologies for advanced threat prevention products, 2013.

[33] FireEye. Fireeye products and solutions. http://www.fireeye.com/, 2013.

[34] FIRST. Forum of incident response and security teams. http://www.first.org/, 1989-2013.

[35] FortiNet. Fortigate products and solutions. http://www.fortinet.com/, 2013.

[36] Simon Hansman and Ray Hunt. A taxonomy of network and computer attacks. *Computers & Security*, 24(1):31–43, 2005.

[37] Keith Harrison and Gregory White. A taxonomy of cyber events affecting communities. In *Proceedings of the 2011 44th Hawaii International Conference on System Sciences*, HICSS '11, pages 1–9, Washington, DC, USA, 2011. IEEE Computer Society.

[38] Kelly Jackson Higgins. How lockheed martin's 'kill chain' stopped securid attack, February 2013. http://www.darkreading.com/attacks-breaches/how-lockheed-martins-kill-chain-stopped/240148399.

[39] Trustwave Holdings. A multi-level approach to addressing targeted attacks. https://www.trustwave.com/whitePapers.php, 2012.

[40] John D. Howard. *An Analysis of Security Incidents on the Internet 1989-1995*. PhD thesis, Carnegie-Mellon University, Pittsburg, PA 15213, April 1997.

[41] John D. Howard and Thomas A. Longstaff. A common language for computer security incidents, 1998.

[42] Eric M. Hutchins, Michael J. Cloppert, and Rohan M. PH.D. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. In *Proc. 6th Int'l Conf. Information Warfare and Security (ICIW 11)*, pages 113–125. Academic Conferences Ltd., 2010.

http://www.lockheedmartin.com/content/dam/lockheed/data/corporate/documents/LM-White-Paper-Intel-Driven-Defense.pdf.

[43] Grégoire Jacob, Ralf Hund, Christopher Kruegel, and Thorsten Holz. Jackstraws: Picking command and control connections from bot traffic. In *USENIX Security Symposium*. USENIX Association, 2011.

[44] Jay Jacobs. Ask the data: On opportunistic attacks, part 1. http://www.verizonenterprise.com/security/blog/?postid=1587, August 2012.

[45] Jay Jacobs. Ask the data: On opportunistic attacks, part 2. http://www.verizonenterprise.com/security/blog/?postid=1589, September 2012.

[46] Jay Jacobs. Ask the data: On opportunistic attacks, part 3. http://www.verizonenterprise.com/security/blog/?postid=1593, September 2012.

[47] Lt. Col. Lionel D. Alford Jr. Cyber warfare: A new doctrine and taxonomy. *The Journal of Defense Software Engineering*, 2001.

[48] Eugene Kaspersky. Eugene kaspersky blog. http://eugene.kaspersky.com/, October 2013.

[49] Kaspesky. Anti-malware for workstation. http://www.kaspersky.com/business-security/endpoint-advanced, October 2013.

[50] Kevin S. Killourhy, Roy A. Maxion, and Kymie M. C. Tan. A defense-centric taxonomy based on attack manifestations. In *DSN*, pages 102–. IEEE Computer Society, 2004.

[51] Maria Kjaerland. A taxonomy and comparison of computer security incidents from the commercial and government sectors. *Computers & Security*, 25(7):522–538, 2006.

[52] LastLine. Analyzing and detecting malicious flash advertisements. http://www.lastline.com/papers/flashads.pdf, 2009.

[53] LastLine. Detection and analysis of drive by download attacks and malicious javascript code. http://www.lastline.com/papers/driveby.pdf, 2010.

[54] LastLine. Analyzing malicious shellcode dynamically. http://www.lastline.com/papers/shellcode_analysis.pdf, 2011.

[55] LastLine. The holy grail: Automatically identifying command and control connections from bot traffic. http://www.lastline.com/papers/commandandcontrol.pdf, 2011.

[56] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 36(1):16 – 24, 2013.

[57] C. Meyers, S. Powers, and D. Faissol. Taxonomies of cyber adversaries and attacks: a survey of incidents and approaches. Technical report, Lawrence Livermore National Laboratory, 2009.

[58] J. Mieres. Russian prices of crimeware. http://evilfingers.blogspot.com/2009/03/russian-prices-of-crimware.html, March 2009.

[59] Brown Moira West, Stikvoort Don, Kossakowski Klaus-Peter, Killcrece Georgia, Ruefle Robin, and Zajicek Mark. Handbook for computer security incident response teams (csirts), 2003.

[60] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 231–245, 2007.

[61] ENISA European Union Agency For Network and Information Security. Incident taxonomy. http://www.enisa.europa.eu/activities/cert/support/incident-management/browsable/incident-handling-process/incident-taxonomy/how-to-use-a-taxonomy, October 2013.

[62] Mary C. Parmelee. Toward the semantic interoperability of the security information and event management lifecycle, 2010.

[63] Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, pages 26–26, Berkeley, CA, USA, 2010. USENIX Association.

[64] Vassilis Prevelakis and Diomidis Spinellis. Sandboxing applications, 2001.

[65] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holzaff. Automatic analysis of malware behavior using machine learning. *J. Comput. Secur.*, 19(4):639–668, December 2011.

[66] Martin Roesch. Snort - lightweight intrusion detection for networks. In *Proceedings of the 13th USENIX conference on System administration*, LISA '99, pages 229–238, Berkeley, CA, USA, 1999. USENIX Association.

[67] Cuckoo Sandbox. Cuckoo sandbox. http://cuckoosandbox.org/, 2013.

[68] Chris Simmons, Charles Ellis, Sajjan Shiva, Dipankar Dasgupta, and Qishi Wu. Avoidit: A cyber attack taxonomy. 2005.

[69] Jungsuk Song, Hayato Ohba, Hiroki Takakura, Yasuo Okabe, Kenji Ohira, and Yongjin Kwon. A comprehensive approach to detect unknown attacks via intrusion detection alerts. In Iliano Cervesato, editor, *Advances in Computer Science – ASIAN 2007. Computer and Network Security*, volume 4846 of *Lecture Notes in Computer Science*, pages 247–253. Springer Berlin Heidelberg, 2007.

[70] Axelsson Stefan. Intrusion detection systems: A survey and taxonomy. Technical report, 2000.

[71] Symantec. Symantec intelligence. http://www.symantec.com/connect/symantec-blogs/symantec-intelligence, October 2013.

[72] Michael Tanji. Beating hackers at their own game. http://www.carbonblack.com/beating-hackers/, November 2011.

[73] James Tarala. Implementing the 20 critical controls with security information and event management (siem) systems. http://www.sans.org/reading-room/analysts-program/siem-systems-arcsight, 2011.

[74] AV Test. The indipendent it-security institute. http://www.av-test.org/, 2013.

[75] Gordon Thomson. Byod: enabling the chaos. *Network Security*, 2012(2):5–8, 2012.

[76] Olivier Thonnard, Leyla Bilge, Gavin O'Gorman, Seán Kiernan, and Martin Lee. Industrial espionage and targeted attacks: Understanding the characteristics of an escalating threat. In Davide Balzarotti, SalvatoreJ. Stolfo, and Marco Cova, editors, *Research in Attacks, Intrusions, and Defenses*, volume 7462 of *Lecture Notes in Computer Science*, pages 64–85. Springer Berlin Heidelberg, 2012.

[77] C.J. Tucker, S.M. Furnell, B.V. Ghita, and P.J. Brooke. A new taxonomy for comparing intrusion detection systems. *Internet Research*, 2007.

[78] Symons Institute Berkley University. Theoretical foundations of big data analysis. http://simons.berkeley.edu/programs/bigdata2013, 2013.

[79] WebSense. Websense products and solutions. http://www.websense.com/, 2013.

[80] Lok Kwong Yan and Heng Yin. Droidscope: Seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 29–29, Berkeley, CA, USA, 2012. USENIX Association.

[81] Bonnie Zhu, Anthony Joseph, and Shankar Sastry. A taxonomy of cyber attacks on SCADA systems. In *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, ITHINGSCPSCOM '11, pages 380–388, Washington, DC, USA, 2011. IEEE Computer Society.

List of Figures

1.1 AVOIDIT Taxonomy (from [68]) 5
1.2 Taxonomy suggested by ENISA (form [61]) 7

2.1 Functional taxonomy of an IDS (form [26]) 18
2.2 Footprint of IDSs using IDS matrix taxonomy (form [77]) . . . 21

5.1 Diagram representing activities and artifacts in the proposed methodology 61

6.1 Network architecture of the deployed test bed. 64

List of Tables

3.1 Pros and cons of the three main types of IDS 26
3.2 Anubis[11, 12] (sandbox, malware detector) summary table . . 33
3.3 Chuckoo Sandbox[67] (sandbox) summary table 34
3.4 Malheur [65](sandbox-report analyzer) summary table 34
3.5 OSSIM [3](SIEM) summary table 35
3.6 FireEye Platform[33] (APT detection, SIEM, Sandbox, Network Sensor) summary table 36
3.7 LastLine Previct (APT detection, SIEM, Sandbox, Network Sensor) summary table 37
3.8 FortiGate[35] (NGF, sandbox, network sensor, APT detection) summary table 38
3.9 SourceFire FireAMP [20](APT detection, SIEM, sensors, sandbox) summary table 39
3.10 WebSense Web Security Gateway[79] (Advanced Web Proxy, sandbox) summary table 40
3.11 Damballa Failsafe[24] (APT detection, SIEM, Sandbox, Network Sensor) summary table 41
3.12 Symantec Endpoint Protection (APT detection, behavioural AV) summary table 42

6.1 Pass/Fail/NA result of malware delivery phase 67
6.2 Pass/Fail/NA result of command and control phase 68