CORE SECURITY TECHNOLOGIES

An Argument for an Automated Penetration Testing Framework

With a Technical Introduction to CORE IMPACT

CORE IMPACT

Table of Contents

Introduction

The information security industry is evolving rapidly as the IT infrastructure of enterprises becomes increasingly complex, heterogeneous, extended… and more porous. New security software products -- from firewalls, VPNs, and PKI to vulnerability scanners and intrusion detection systems -- and services have emerged to address the multitude of interconnected technologies deployed. The increase of information-security software and services points to the growing corporate and government awareness of the need to both protect their IT assets *and* conduct their business without disruption.

As information security technology has evolved, so too have the services provided by consulting firms and security companies. New, more sophisticated services have emerged as part of a wider model of risk assessment and risk mitigation. In this context, *penetration tests* have become an accepted practice to assess and improve an organization's system security against would-be attackers.

However, the penetration testing services are still in their infancy, relying on an amalgam of informal knowledge, hacker handiwork, scarce security expertise, and the need for a successful execution of time-constrained tasks during the bounds of a consulting engagement.

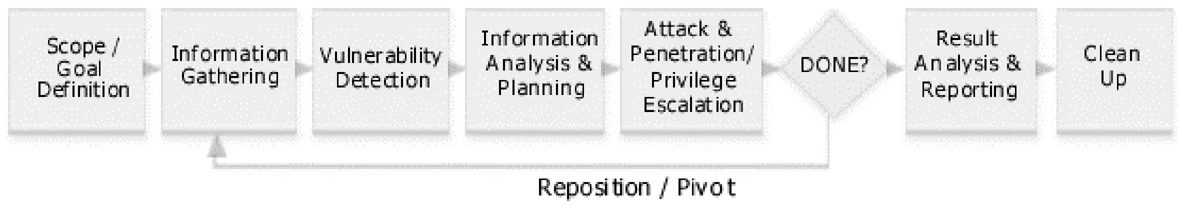
In contrast to the frailties of emerging penetration testing services, CORE IMPACT provides a comprehensive software framework for IT personnel to perform penetration tests as frequently as they need to with a professional methodology and tools. In addition to offering a superior means of conducting penetration testing, CORE IMPACT offers these additional benefits: It increases the productivity of the testers by expediting the penetration testing process, leverages knowledge acquired by the testers by logging all methods and results, ensures timeliness of exploits through real-time updating, and provides actionable results for management decision-making.

The Penetration Test

Penetration Testing is commonly viewed as a subset of a broad computer security audit. In particular, the goal of penetration test is to attempt to gain access to software systems that require authorized access, such as operating systems or applications such as database servers and accounting systems. Therefore, performing a Penetration Test requires that the tester gain access to a system under the testing premise that one has no legitimate access to that system.   Testers commonly employ a variety of informal methods to achieve such access, highly dependent on the sophistication of the tester. However, all penetration tests have definable stages or phases within each test. At present, a penetration test typically requires the use of several disparate software packages, scarcely efficient for conducting so crucial a test process.

Current Methodology of a Penetration Test

While there are no formal standards for performing penetration tests, a penetration test typically has seven stages, which may vary based on how consulting services firms conduct them, but which are otherwise fairly universal.


Scope / Goal Definition
Information Gathering
Vulnerability Detection
Information Analysis & Planning
Attack & Penetration/ Privilege Escalation
DONE?
Result Analysis & Reporting
Clean Up
Reposition / Pivot

Stage 1 – Scope / Goal Definition

This initial stage focuses on defining the success criteria as well as the scope for this penetration test. Scope is usually defined in terms of which attacker profile the tester will use (simple external hacker with no knowledge about the target, skilled external hacker with no knowledge about the target, internal user with access, etc), which systems or networks the test will be conducted on and how long will the test last.

Stage 2 – Information Gathering

This stage of a penetration test is designed to provide the tester with information about the target(s) at hand. Information gathering focuses on both technical details on the target application or target network and on publicly available information about the owner of the network or application in question.

Stage 3 – Vulnerability Detection

This stage focuses on identifying vulnerabilities in the selected targets so that the tester can gain access. Identification of vulnerabilities can be performed with software or manually.

- Automated vulnerability scanning
    - Commercial and freeware software known as *vulnerability* scanners or simply scanners can be deployed to audit the target for known software vulnerabilities. When given a target, vulnerability scanning software will enumerate known flaws that could be used to attack the host to gain access. However, these packages in and of themselves *do not perform a penetration of the target host.* Their capability is limited to the identification of flaws and exploitation of the flaws only in so much as to prove the existence of flaws, often presenting lots of false positives.

- Manual scanning
  - The tester can then manually probe the target host for common misconfigurations or flaws because a vulnerability scanner can fail to identify certain vulnerabilities.
- In-house research
  - Finally the tester can elect to obtain a mirror copy of the software application or system within the testing lab and examine it exhaustively for previously unknown security flaws.

Stage 4 – Information Analysis and Planning

This stage collates the information gathered in previous stages. Once technical and public information is collated, the tester can begin the high-level attack planning, from the overall approach of the penetration test to identifying which targets require further research.

Stage 5 – Attack & Penetration / Privilege Escalation

This stage is divided into two sub-stages:

Attack & Penetration

This stage of the penetration testing process attempts an actual break-in. This stage is dependent on the successful completion of previous stages and can be broken down more granularly as follows:

- Known/available exploit selection
  - At this point, the tester acquires publicly available software programs to exploit a problem identified in Stage 3. These programs are referred to as *exploits.*
- Exploit customization
  - The tester must often customize the exploit software program to work as desired against their selected target. Different environments, software revisions, and other factors must be treated differently, necessitating customization.
- Exploit development
  - In some cases there may be no exploit program available. In such cases, the tester is then faced with writing one for the specific target host.

- Exploit testing
  - Each exploit must be tested before use in a formal penetration test in order to avoid damage to an environment if vulnerabilities are improperly executed, resulting in system downtime or destruction.
- Attack
  - The exploit is leveraged against the target in an attempt to gain unauthorized access.

Privilege Escalation

This stage of the process is reached when the tester has successfully gained access to, and acquired the necessary privileges on the target system. This stage also has exceptions for when a tester has gained access to an intermediate system that can serve as a waypoint to further their attack on the target system. An example of this is when a host behind a firewall is compromised so that the tester may stage attacks against hosts previously protected by this security device. This method is classified within this document as pivoting.

In order to perform the tasks detailed in this stage, it is important to note than a tester will often significantly alter the breached or compromised systems because in order to attain privilege escalation or to prepare a machine to act as a staging point, it is often necessary to change the system itself. This is often achieved by installing new software packages to help in the tester's efforts or by altering current functionality within an OS or application. A wide variety of tools from disparate sources can be used to make these changes. It should be noted that after the tester positions himself in the newly compromised system, the cycle returns to stage 2, now using the acquired privileges.

Stage 6 – Result Analysis & Reporting

At this point, the tester must organize available data and related result sets in order to report to internal management or to a client. This work typically entails consolidation of information gathered, analysis and extraction of general conclusions and recommendations, and generation of the final presentation and deliverables.

Stage 7 – Clean-Up

As noted in Stage 5, there are often significant changes made to audited systems during the penetration testing process as vulnerabilities are identified. The final stage of a penetration test therefore involves cleaning up all that has been done during the testing process. In this last phase, the tester returns the system and any compromised hosts to the precise configurations they had prior to the penetration test.

Shortcomings of Current Penetration Test Methodologies

The current *ad hoc* method of performing penetration tests does not lend itself to consistent execution. The processes by which it is performed are extremely inefficient because of their reliance on disparate software packages and non-formalized methodologies. Furthermore, because of the wide spectrum of methods and software packages available, one tester's ability to perform consistently professional tests entails a significant and onerous learning curve. This in turn makes the process expensive and inefficient financially because of the lack of dependable replication. In the following section we delineate specific shortcomings in four of the seven stages.

Shortcomings > Stage 4 – Information Analysis and Planning

- It is difficult and time-consuming to consolidate all the information gathered from previous stages and to extract high-level conclusions that will help to define an attack strategy.
- It requires a lot of time and effort to maintain an up-to-date overview of the components and their interaction.
- There are no specific tools aimed at addressing this phase.
- Experienced and knowledgeable resources are required for this stage, and overall time constraints can limit the extent of their work.
- No formal processes or tools exist to help estimate time and effort.

Shortcomings > Stage 5 – Attack & Penetration / Privilege Escalation

Shortcomings > Attack & Penetration

- Publicly available exploits are generally unreliable and require customization and testing (quick hacks, proof of concept code).
- Exploits developed in-house are generally aimed at specific tasks or penetration test engagements and are not readily re-usable in future engagements.
- Knowing that a vulnerability exists does not always mean that it can be exploited easily. Therefore, although penetration is theoretically possible, it's highly possible that there isn't time during the test process to prove the vulnerability. As such, the tester's deliverables may be viewed as incomplete or less than satisfactory.
- Penetration-testing knowledge and specialization are required for exploit and tool development, and these advanced skills are expensive.
- Considerable lab infrastructure is required for successful research, development and testing (e.g., platforms, mixed OS and versions, applications, networking equipment).

Shortcomings > Privilege Escalation

- Some tools and exploits require customization and testing (e.g., local host exploits, backdoors, sniffers, sniffing/spoofing libraries) and are often unreliable, in part because they are frequently not maintained.
- Setting up a new acquired vantage point is monotonous and time-consuming work (involving the installation of software and tools, compiling for the new platforms, and account configuration details).
- Pivoting is currently a non-formalized process, but it is critical to successful completion of a thorough penetration test.
- Considerable lab infrastructure is required for research, development, customization and testing.
- There is often a lack of security architecture for the penetration test itself. The actual test should be secure itself, particularly when transmitting information about vulnerabilities in a target host.

Shortcomings > Stage 6 – Result Analysis & Reporting

- Maintaining a record of all actions, commands, inputs and outputs of tasks performed during the entire penetration test occurs only if there is self-enforcement by the tester(s) so there is no guarantee of accountability or compliance or successful replication of such a test.
- Gathering and consolidating all log information from all phases, including all the program and tools used, is time-consuming and prone to error.
- Organizing information in a format suitable for analysis and extraction of high-level conclusions and recommendations is difficult.
- Performing analysis developing general conclusions and recommendations requires experienced and knowledgeable resources.
- The actual writing of final reports is often tedious and therefore prone to error. A high level of expertise and experience is required to ensure quality, but such skilled resources could be better assigned to other IT endeavors.
- No specialized tools exist to cover these issues.

Shortcomings > Stage 7 – Clean-Up

- A detailed, precise list of all actions performed must be maintained, yet there are only rudimentary tools available for such record-keeping.
- Clean-up of compromised hosts must be done securely and without affecting normal operations. This stage of penetration testing often fails, resulting in system down-time.
- The clean-up process should be verifiable and irrefutable, but current practices do not address this problem.
- Clean-up is often left as a back-up/restore job for IT, affecting normal operations and IT resources.

The challenge is to discover whether sound penetration testing methodologies and tools can be integrated into a framework so that penetration testing can fulfill the information-security assessment needs of complex organizations with thousands of hosts. That was the impetus for creating CORE IMPACT.

CORE IMPACT

CORE IMPACT aims to consolidate in one application the ability to perform a penetration test without being reliant on various software packages or varied methodologies. To accomplish this, the architecture must:

- Make available valuable resources for the more important phases: high-level overview and analysis, strategic attack planning, results analysis, and recommendation formulation.
- Encompass all seven phases of a penetration test in a single framework.
- Define and standardize the penetration test methodology for maximum efficiency and replication of effort.
- Enforce the use of the organization’s methodology and ensure quality results.
- Improve the security of the penetration testing practice itself.
- Simplify and speed execution of monotonous and time-consuming tasks.
- Ultimately improve the information security of the organization in which it is implemented.

CORE IMPACT Architecure

In one application framework, CORE IMPACT consolidates the ability to perform a deep and professional penetration test without being reliant on diverse software packages or varied manual means. IMPACT offers a consistent methodology and broad set of tools to the pen tester.

Its architecture has the following characteristics that together provide the efficiency and depth of analysis required for scrupulous assessment of enterprise vulnerabilities.

An integrated, user-friendly interface. The IMPACT console presents all phases of a penetration test in an intuitive graphical interface, even when running modules on different operating systems and architectures.

Commercial-grade exploit code. IMPACT provides the tester with a range of up-to-date, professionally developed and maintained exploits for different platforms, operating systems and applications, and multiple combinations of these. IMPACT exploits allow the tester to both audit for vulnerabilities and exploit the vulnerabilities to gain and retain access on the target host or application.

Transparent pivoting. IMPACT permits modules to run from intermediate compromised hosts without modification. This powerful capability allows the tester to seamlessly stage or proxy attacks through intermediate hosts. It also elevates the value of attack code developed inside the IMPACT framework as this code becomes available on every system the tester has compromised.

Complete set of information gathering tools. IMPACT includes information-gathering modules ranging from network discovery to port scanning and OS stack fingerprinting. All these modules can be run from any acquired system without modification. Integration with common information-gathering packages, such as nmap and Nessus, is provided in the form of import modules.

Logging and storage of test data in a centralized information repository. Every piece of information about the target network accumulated during the penetration test and all IMPACT-initiated penetration testing activities are logged and stored in a structured format inside a single IMPACT database. This always up-to-date view of the target system aids the tester in high-level strategic analysis of the current scenario, as well as later when reporting results of the test.

A flexible and customizable framework. IMPACT provides a framework completely adaptive to the tester's definition and re-definition of methodologies and targets. IMPACT lets information security experts develop and customize new or existing tools quickly, thus maximizing IT efficiency by reusing knowledge and experience from successive penetration tests and different penetration-testing teams. It also enables the environment-specific design of exploit code and scripts to leverage access independent of a target operating-system or hardware architecture.

Components

The IMPACT architecture is designed to:

- Execute and control modules
- Control agent deployment
- Centralize information gathering
- Display module output and report generation

There are three primary components of the IMPACT architecture working in conjunction with one another to obtain information about a target, compromise it and communicate with it. These three components are Agents, Modules and the Console. All knowledge obtained during automated penetration tests with IMPACT is consolidated in its central information repository, the Entity Database.

Agents

An IMPACT agent is a software program whose primary purpose is to perform operations requested by the console host, ultimately representing the tester's orders. These operations are defined in Modules. Agents can also perform operations on other agents, creating a 'chaining' of agents. These agent chains aid the tester in penetrating deep into networks where a single point-of-entry is possible (because there is a single weak point in the perimeter, either due to OS patch levels or to network filtering).

These agents are commonly deployed on a compromised system immediately following the compromise (for instance, by exploiting a vulnerability on a target). These agents enable the tester to take advantage of the compromised system's credentials and trust relationships within the target network by allowing for the execution of any IMPACT module on them, as if the tester were virtually sitting on the system.

Different agent implementations are available to the tester depending on how intrusive the penetration test, ranging from a completely volatile all-assembly-code implementation that self-destructs when disconnected to a fully persistent multi-tasking agent with a completely authenticated and encrypted communication channel.

Modules

An IMPACT module is an individual operation or group of operations to be executed by an agent. For example, a module can implement specific attacks to be launched against a target host, such as a web server, or implement information gathering techniques ranging from packet sniffing to active port scanning. And, by using the macro-building capability in IMPACT, a tester can cause a module to execute other modules.

IMPACT modules are written in the Python language (www.python.org) and run using a Python Virtual Machine embedded in the console. Core Security Technologies chose Python as the scripting language for IMPACT because it is a straightforward and easy to learn object-oriented language, backed by an open source standard, that in turn is supported by an open community. Its clarity and fast learning curve facilitate the tester's ability to understand and extend IMPACT modules.

The Console

The console is the graphical user interface of IMPACT exposed to the tester. The console presents the launching point for all modules, a management panel to visualize the attacked network, and a reporting tool for the viewing of result information. The console has an embedded agent inside – the *localagent* – and by default is the starting point of a penetration test. By interacting with the IMPACT console, the tester controls the execution of modules. Since modules run on a specific agent, there is always a selected agent for execution, the *default source agent*. By default, when the console starts, the *localagent* is selected as the *default source agent*.

Furthermore, the console consolidates all information obtained from agents deployed across multiple targets and varying operating systems. The console provides visualization of data from a specific network scan output to a module's success against a remote system.

Entity Database

The entity database is the centralized repository of all information pertaining to a penetration test, containing information on module output, activity logs, information about target systems (hosts that are known, operating systems they are running, open ports, and so on), and agent deployment. By accessing this database, the tester can assess the state of the whole penetration test at any point in time.

Structured information inside this database, such as target networks, hosts, deployed agents, open ports on a host and found user accounts, are represented as objects within this database. These database objects are referred to in the product as entities. Entities can also compare different versions of them and resolve conflicts (for example, allowing the tester to choose between different portscan results for the same hosts). Upon initialization, some default entities are created and added to the database. These entities are:

- A host entity representing the local console host *(localhost)*
- An agent representing the local agent *(localagent)*

This database is implemented on top of a Microsoft Access database engine, which is completely accessible to the tester, allowing the tester to export the data and manipulate it easily in other programs or to export the data to an XML format.

Summary

This paper has described the scope of the CORE IMPACT architecture, designed to facilitate professional, comprehensive and actionable penetration testing of complex IT environments, and to facilitate *re-use* of both the exploits and the knowledge base of the testers so that penetration testing can be conducted time- and cost-effectively by the organization sponsoring the effort.