Towards Improving Productivity in NMap Security Audits

Abstract

Maintaining an adequate security level in computer infrastructures requires periodic assessment of their vulnerabilities. This is specially critical in Internet-facing web servers. These audits require specialized security tools. `nmap` is arguably the most popular one, due to its versatility, powerful features, and low resource usage. However, this versatility turns using `nmap` difficult and error-prone. Finding its most convenient features can be difficult, and usage errors are reported at runtime. This commonly leads users with low experience to suboptimal results. This research aims to decrease this complexity by developing an advanced web GUI for `nmap` that solves the shortcomings of the official one, complemented with a statically typed DSL that early detects syntax, type and semantic usage errors. These products expand the usage possibilities of `nmap`, creating schedulable, distributable, and portable auditing tasks able to find anomalies analyzing their output. Our initial release shows that the web GUI has been well received by several security related media and professionals. The DSL can detect a wide range of potential errors, substantially increasing the robustness of the auditing tasks created with the web GUI or standalone. Therefore, using statically typed DSLs with early detection of type errors can be a suitable tool to lower the complexity of tools with a large number of possibilities such as `nmap`.

Keywords: `nmap`, web GUI, advanced features, productivity, DSL, static type checking .

1 Introduction

Attacks to computer system infrastructures increase each year [30, 32, 47]. They also get more sophisticated and varied, taking advantage of subtle implementation details of protocols and services [15]. This not only concerns private companies [60], but also public institutions, governments [55], and military infrastructures [45]. Their consequences can be also severe: reveal of secrets [3], denial of service [7, 31], stealing private information from users [59], or taking control of attacked infrastructures to perform various malicious activities [20] are some examples. Web applications may be used by a larger and less-trusted user base than traditional client-server applications, specially if facing the Internet. Because of this, both web applications and their hosting servers are preferred targets for malicious activities [24], and thus more potentially vulnerable to different attack types [23]. Therefore, making computer systems more resilient to different forms of attacks (specially web servers) is a critical task [6].

The security status of computer systems can be evaluated through audits. Professionals are hired to use the same techniques that malicious attackers may use against them, but on controlled environments and agreeing to certain limitations. Problems found during these audits are usually written to a report detailing their causes, potential solutions, or mitigations. The result determines the ability of a system and its running services to resist different types of attack. This activity is commonly called *pentesting* [43].

Pentesting activities require specialized tools. The `nmap` network analysis tool [38] is arguably the most popular and widely-used one. `nmap` was created in 1997 to discover hosts and services on computer networks (thus building a “Network Map”). Its user base and popularity greatly increased when it was ported to all major operating systems. It also won the *Linux Journal Editor’s Choice Award* in 2001 [21], appeared in numerous news media [36], and movies [35]. It is also part of popular automatic vulnerability discovery tools (see Section 6).

`nmap` capabilities have increased over the years. Nowadays, `nmap` performs a wide variety of security tests beyond the common service type and version discovery related tasks. `nmap` has an extensive and customizable library of scripts [37] and a scripting engine (NSE), enabling a great variety of more advanced and specialized security tests. For example, it is possible to establish the geographical location of scanned targets (`ip-geolocation-maxmind`), locate malware-spreading hosts via the *Google Safe Browsing* API (`http-google-malware`), locate active e-mail accounts (`http-google-email`), or perform web server specific testing `http-methods`. It also supports a wide range of scanning techniques and integrates firewall/IDS evasion features. This makes `nmap` a very flexible and powerful security tool.

However, with 4 ways of specifying targets, more than 110 options (divided in 10 categories), multiple option combinations that achieve different effects, and more than 600 standard NSE scripts [37] (also supporting the installation of additional third-party ones), using the full range of possibilities of `nmap` can be difficult, especially to students or users with little experience. Detailed information about `nmap` options and scripts is not provided within the tool or its official GUI (*Zenmap* [39], see Section 2.1). Users usually rely on third-party information sources to perform non-basic scans, and so they may miss or misuse some important options to achieve their goals, leading to suboptimal results. Option syntax and parameters is usually very different from each other, so its usage can be complex and prone to mistakes. Errors are always reported at runtime, while the scan process is running. This usually forces users to run a scan several times until its configuration is correct, hence decreasing their productivity.

The research presented in this paper aims to solve these shortcomings developing two different related pieces of software. *NMapGUI* is an advanced, portable and fast web user interface for `nmap`. Its goal is to give users structured access to all `nmap` possibilities, along with detailed usage information to try to decrease its usage difficulty while increasing user productivity, especially when using advanced options or NSE scripts. Complementarily, *NMapDSL* is a statically typed and portable *Domain-Specific Language* (DSL), able to schedule, analyze the results, and early detect `nmap` syntax, type, or semantic usage errors when creating auditing tasks. Tasks designed with *NMapGUI* can generate *NMapDSL* code, but the language is able to run as a stand-alone component to be usable from systems without GUI support. The language also facilitates sharing and distributing auditing tasks, which will encourage knowledge transfer from experienced users to beginners, helping security related courses. We will review the design principles, features, usage scenarios and planned future functionality of our research, and also how it was received by security-related media and individual users.

The rest of this paper is structured as follows. Section 2 describes or web GUI (*NMapGUI*) detailing the advantages it has over to the official GUI (subsection 2.1), main design features (subsections 2.2.1, 2.2.2, 2.2.3, and 2.2.4), and other minor design features (subsections 2.3.1, 2.3.2, and 2.3.3). The web GUI architecture and implementation characteristics will be reviewed in subsection 2.4, along with sample usage scenarios (subsection 2.5). Section 3 describes *NMapDSL*, describing its syntax (subsection 3.1), script validation (subsection 3.2), and execution results (3.3). Finally, Sections 4, 5, and 6 describe the reviews made by security-related media and users to the current release of our research prototype, the conclusions, future, and related work.

2 NMapGUI

2.1 Shortcomings of the official `nmap` GUI

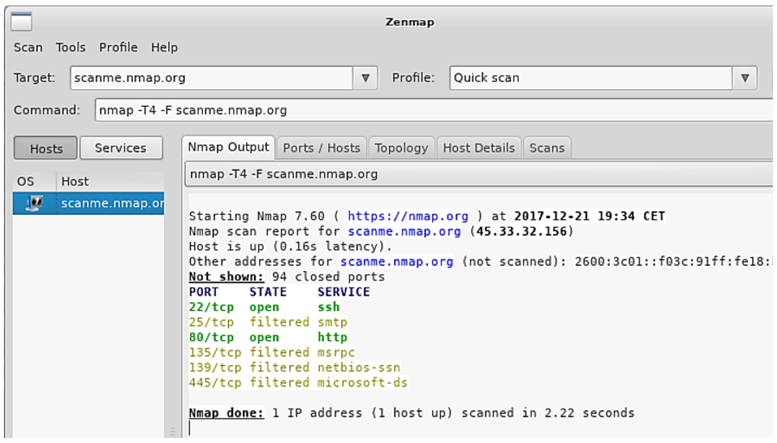
Using `nmap` with its official `nmap` GUI (*Zenmap*) is simple. Users must choose one *scan profile* and provide scan targets. *Scan profiles* are a short list of 10 predefined combinations of `nmap` options, determining the amount or type of information that will be obtained when the scan finishes. However, *Zenmap* do not provide descriptions about the goals of each profile, so users may not be able to fully understand the operations they perform or the amount of information they obtain. For example, scan tasks using profiles that obtain a great deal of information, such as `Intense scan`, may require much more time to finish than using "lighter" ones like `Ping scan`, but this is not indicated to the user. Once a configured `nmap` execution finishes, *Zenmap* shows the following information (see Figure 1):

1. The `nmap` raw output
2. Services running in each target ports
3. Extended information about each discovered service, but only if the scan profile includes it
4. A diagram of the subnets that `nmap` has passed through to reach the targets
5. Extended target details (such as predicted OS type and version), again depending on the chosen scan profile

Unfortunately, *Zenmap* features do not facilitate the usage of the full `nmap` auditing potential, as will be described in the following subsections.

2.1.1 No information about individual `nmap` options

Zenmap scan profiles facilitate `nmap` usage, but do not cover all possible options, combinations, or scan types, just a few of common usages. Customizing auditing tasks requires manually editing the options to be used, which is an error-prone process identical to run `nmap` in a system console: each option and parameter must be manually written in the *Zenmap* command box (see Figure 1). As *Zenmap* do not provide help about individual options, users



Target: scanme.nmap.org
Profile: Quick scan
Command: nmap -T4 -F scanme.nmap.org

```
nmap -T4 -F scanme.nmap.org

Starting Nmap 7.60 ( https://nmap.org ) at 2017-12-21 19:34 CET
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:
Not shown: 94 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
25/tcp   filtered smtp
80/tcp   open     http
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in 2.22 seconds
```

Figure 1 Typical *Zenmap* scan output.

frequently consult the effects and syntax of each available one using third-party support or training materials. This way, finding the correct combination to do a specific audit task can be complex and time-consuming, decreasing productivity.

2.1.2 Lack of support to NSE scripts

Even though the NSE script library gives `nmap` a lot of flexibility, and dramatically enhances its usefulness in multiple scenarios, *Zenmap* does not provide information about it, not even a list of available scripts. This is a very important shortcoming, as scripts perform very specialized tasks and may be key to accurately perform security audits with particular requirements. For example, a script like `http-affiliate-id`, grabs affiliate network IDs from a web page, so pages with the same owner can be identified and studied together. Crawling and analysis of web error pages is offered by the `http-errors` script, which can be used to guess important information about an audited system or the technologies used in their hosted webs [42]. Several scripts also target specific hardware types or vendors: `broadcast-bjnp-discover` discovers *Canon* printers in a network. Others detect backdoors in certain devices, as the `http-dlink-backdoor` script does with certain *D-Link* router models. Therefore, *Zenmap* does not facilitate using the full `nmap` potential, and users with little experience or students may be unaware of the possibilities they really have. Additionally, this also increases the possibility of missing important scripts that could drastically change the outcome of a security audit. For example, if the target network contains *D-Link* routers, being unaware of the `http-dlink-backdoor` script may leave some vulnerabilities undiscovered, giving false negatives.

2.1.3 No parallel scan tasks

Running several scans simultaneously requires multiple *Zenmap* instances: once a scan is configured and started, pressing "Scan" again restarts it. This way, users may accidentally reset running scan processes, losing their work. This may decrease user productivity and forces the user to navigate through different windows to know the status of each running scan.

2.1.4 No scan scheduling or result analysis support

Configured scan tasks are manually run once, as no scheduling support is provided. Additionally, no reporting or output anomaly analysis features are supported, so these have to be performed by third-party tools. This lowers the usefulness of `nmap` in some pentesting scenarios, forcing the user to look for or create tools to effectively use `nmap` in scenarios where, for example, a low-cost surveillance method is needed, such as IoT networks.

2.2 *NMapGUI* main design features

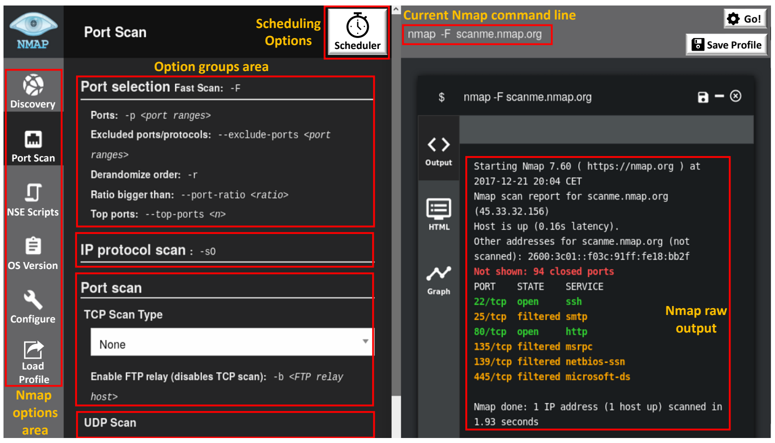
NMapGUI has the following features to solve the described shortcomings.

2.2.1 Classification and usage of all `nmap` options

NMapGUI displays the full range of `nmap` options divided in four areas, with a fifth one ("Global scanning options") containing those that change the behavior of every scan (see Figure 2):

1. *Host discovery*: related to finding different types of targets
2. *Port scanning* (see Figure 2): enabling users to find services running at concrete ports in the targets
3. *Usage of advanced NSE scripts* (see Figure 3): that facilitates using NSE scripts in any scanning task
4. *OS and service version detection*: related to the detection of OS/service types and versions running on the targets

Inside each area, *NMapGUI* groups options by their main purpose (Figure 2). Each individual option has a short description and a syntax example to decrease the probability of errors and avoid consulting external information sources.



Port selection Fast Scan: -F

Ports: -p *<port ranges>*
Excluded ports/protocols: --exclude-ports *<port ranges>*
Derandomize order: -r
Ratio bigger than: --port-ratio *<ratio>*
Top ports: --top-ports *<n>*

IP protocol scan : -sO

Port scan

TCP Scan Type: None

Enable FTP relay (disables TCP scan): -b *<FTP relay host>*

UDP Scan

```
nmap -F scanme.nmap.org
```

```
$    nmap -F scanme.nmap.org
```

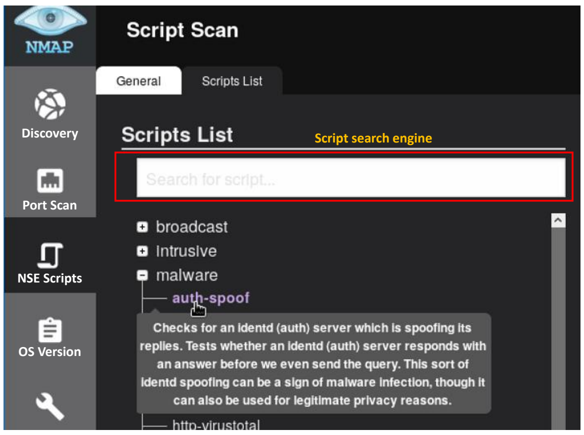
```
Starting Nmap 7.60 ( https://nmap.org ) at
2017-12-21 20:04 CET
Nmap scan report for scanme.nmap.org
(45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not
scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 94 closed ports
PORT     STATE    SERVICE
22/tcp   open     ssh
25/tcp   filtered smtp
80/tcp   open     http
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds

Nmap done: 1 IP address (1 host up) scanned in
1.93 seconds
```

Figure 2 `nmap` UI design and `nmap` output.

NMapGUI allows users to create scan tasks only using GUI elements: clicking on a `nmap` option in any of the described areas automatically adds it to the command line to be executed. Similarly, if the option has been already included, it will be removed. However, some users still wanted to manually customize the command to be run (see Section 4), even if part of it was already specified using GUI elements. Consequently, *NMapGUI* also allows writing `nmap` options directly in the command section, enabling combined manual- and click-based command construction. Pressing the “Go!” button (Figure 2) will run the current command as a background task, so a new scan can be immediately built and run.

2.2.2 `nmap` NSE scripts support

NMapGUI organizes currently installed NSE scripts in a category tree, depending on their main purpose (see Figure 3). A keyword search engine is also implemented to facilitate finding them by description. Script descriptions are obtained from its official documentation and are also shown as a tooltips. This way, users have immediate access to information to decide if a script is suitable or not, decreasing the possibility of consulting external information sources. This is specially aimed to facilitate the work in learning environments.



Checks for an identd (auth) server which is spoofing its replies. Tests whether an identd (auth) server responds with an answer before we even send the query. This sort of identd spoofing can be a sign of malware infection, though it can also be used for legitimate privacy reasons.

Figure 3 NSE script information in *NMapGUI*.

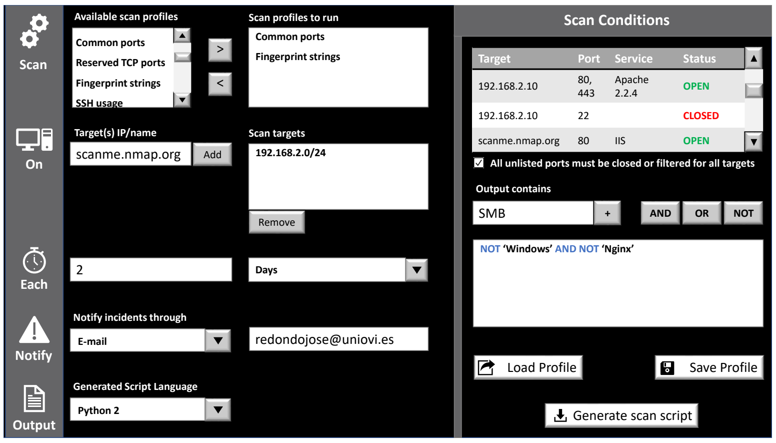
2.2.3 Parallel execution of scanning processes

Users are allowed to run multiple instances of `nmap` with different option sets simultaneously within the same GUI instance. Every scan task runs in a different thread, and its output will be displayed on an independent minimizable window at real-time (see Figure 2). This way, users may increase their productivity, especially if the scans take a long time to complete (as happens for example with those that use IDS evasion techniques).

2.2.4 Scan scheduling and anomaly reporting

Periodically running `nmap` auditing tasks over machine infrastructures analyzing its output looking for anomalies is a very useful and low-resource consumption surveillance practice. An anomaly can be defined as a condition that targets fulfill when they should not to, or vice versa. For example, if a web server is expected to have only the 22 (`SSH`), 80 (`HTTP`) and 443 (`HTTPS`) ports constantly open, presence of additional open ports is anomalous. This might indicate a simple misconfiguration, but also an ongoing data exfiltration process from a compromised machine or web site. Also, finding these ports closed or unresponsive may indicate that a machine is down or its web server processes have crashed, so the service is interrupted.

Periodic surveillance and automated anomaly reporting allow users to detect problems earlier, when the damage may not be very high, or quickly develop corrective responses without further compromising the infrastructure. Using `nmap` for these tasks enhances the security level of an infrastructure with lower resource usage that more complex (albeit more versatile) threat



Available scan profiles
- Common ports
- Reserved TCP ports
- Fingerprint strings
- SSH usage

Scan profiles to run
- Common ports
- Fingerprint strings

Target(s) IP/name: scanme.nmap.org

Scan targets: 192.168.2.0/24

2 Days

Notify incidents through: E-mail redondojose@uniovi.es

Generated Script Language: Python 2

Scan Conditions

Target	Port	Service	Status
192.168.2.10	80, 443	Apache 2.2.4	OPEN
192.168.2.10	22		CLOSED
scanme.nmap.org	80	IIS	OPEN

☑ All unlisted ports must be closed or filtered for all targets

Output contains: SMB

NOT 'Windows' AND NOT 'Nginx'

Figure 4 Current prototype of the scheduling GUI proposed to be added to *NMapGUI*.

detection software [2]. Therefore, this surveillance method can be deployed within infrastructures with potentially low resources and/or bandwidth, such as IoT networks, easier. Unfortunately, neither `nmap` nor *Zenmap* offer support for this type of usage. This problem also has been addressed in the *NMapGUI* design.

The first step to create scan scheduling support is to give configured scans its own entity. Scan task configurations created in *NMapGUI* can be saved as a *named scan profile* (see Figure 2). An optional long description with the purpose of the `nmap` options used in the task can also be provided. So, instead of providing a fixed amount of scan profiles as in *Zenmap* (see Section 2.1), users may create their own ones or reuse them from other users.

named scan profiles are JSON files composed by key-value pairs. They contain the `nmap` command line to be run (except the targets) and a description of its effects, so they can be easily shared and understood. Using multiple *Named scan profiles* in a scan task combines their options. *Named scan profiles* can also be used in *NMapDSL* programs (see Section 3).

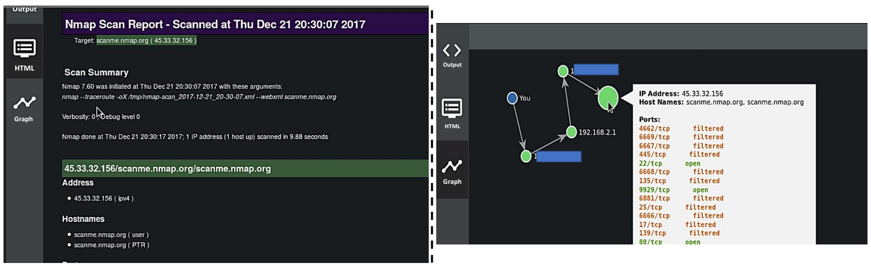
The current release of *NMapGUI* and *NMapDSL* includes 37 different `named scan profiles`, covering a wide range of `nmap` options. Some of them use common features, like different TCP detection techniques (`TCP_IP_scan`, `TCP_SYN_scan`, ...), but others also have more advanced options. For example, there are profiles with firewall/IDS detection and evasion techniques (`detect_firewall`, `avoid_firewall_detection_packet_fragments`, ...), detection of

NETBIOS services (netbios_find), or even testing specific vulnerabilities (netbios_vuln_MS08067). This way, *named scan profiles* facilitate using advanced nmap features without requiring intensive study of the tool, and eases moving certain scan configurations to infrastructures needing them.

Figure 4 shows the *NMapGUI* window that supports scheduling and surveillance features using *named scan profiles*. Multiple ones can be chosen to be run over selected targets at a certain rate. Periodicity of tasks may be specified from minutes to months, depending on the surveillance frequency requirements. Detected anomalies may be reported through email or written to a local or remote file to be consulted later. Support for anomaly detection is divided in two categories:

1. *Unwanted port states or running services*: The GUI uses a table to specify expected target IP/DNS names, ports, running services, and port status. The example in Figure 4 shows that the service named Apache 2.2.4 is expected to be running in ports 80 and 443 of the target IP, all of them open to traffic. Also, the port 22 (SSH) needs to be explicitly closed on the same machine. Additionally, results of the scheduled scan must also indicate that IIS is running in the port 80 of scanme.nmap.org. Service name comparison is case-insensitive, and exact substring matching will be used once trimmed. Finally, by default all unlisted ports in the scan conditions table must not appear in the task output, be closed, or filtered to not to generate an anomaly alert. This works as a blacklist, when everything is denied unless it is explicitly granted, that is the most secure approach in this scenario. The user may lift this restriction.
2. *Unwanted strings in scan results*: This allows users to create simple logical expressions with strings, specifying texts that may (or may not) appear in the scan output. The purpose of this functionality is to provide basic support to detect certain services running in environments when they should not, machines with unwanted software (OS, service types, or service versions), or just check that some expected services are really visible from the outside. In Figure 4, no machine should have a Windows operating system detected and no Nginx web server must be running. Detecting these strings in the scan output will be reported as errors to the users. OR and OR NOT operators have similar semantics, but it will be reported as standard information messages.

Once the scheduling behavior is specified, *NMapGUI* generates the corresponding *NMapDSL* program. These programs can be translated to existing



Nmap Scan Report - Scanned at Thu Dec 21 20:30:07 2017

Target: scanme.nmap.org (45.33.32.156)

Scan Summary

Nmap 7.60 was initiated at Thu Dec 21 20:30:07 2017 with these arguments:
nmap --traceroute -oX /tmp/nmap-scan_2017-12-21_20-30-07.xml --webxml scanme.nmap.org

Verbosity: 0 ; Debug level 0

Nmap done at Thu Dec 21 20:30:17 2017; 1 IP address (1 host up) scanned in 9.88 seconds

45.33.32.156/scanme.nmap.org/scanme.nmap.org

Address

- 45.33.32.156 (ipv4)

Hostnames

- scanme.nmap.org (user)
- scanme.nmap.org (PTR)

```
IP Address: 45.33.32.156
Host Names: scanme.nmap.org, scanme.nmap.org

Ports:
4662/tcp      filtered
6669/tcp      filtered
6667/tcp      filtered
445/tcp      filtered
22/tcp      open
6668/tcp      filtered
135/tcp      filtered
9929/tcp      open
6881/tcp      filtered
25/tcp      filtered
6666/tcp      filtered
17/tcp      filtered
139/tcp      filtered
80/tcp      open
```

Figure 5 HTML report and *traceroute* graph in *NMapGUI*.

scripting languages supported by the scanning machines OS (see Section 3). This also opens the possibility of running surveillance tasks with technologies like *Docker* (creating ad-hoc containers with `nmap` installed), or from automated infrastructure deployment products, such as *Puppet*. These configurations may be also saved or loaded to be modified at any time.

2.3 Other design features

2.3.1 Exportable output

As in *Zenmap* “Host details” tab, *NMapGUI* have been designed to display running `nmap` processes raw output (see Figure 2). It also uses the `nmap` output options to display a HTML report (see Figure 5). So, results of `nmap` scans tasks can be easily included as part of *pentesting* reports. Exporting the output as XML is also supported, to enable its processing by third-party tools.

2.3.2 Enhanced `traceroute` graph

NMapGUI provides an interactive `traceroute` graph to help mapping the scanned network topology (see Figure 5). This graph is an enhancement of the *Zenmap* one: node layout can be customized to facilitate displaying the graph in *pentesting* reports. Additionally, clicking on each node displays basic information about it (including its open ports if it is the final target).

2.3.3 Portability and no-install usage

NMapGUI was designed to run in any system also able to run `nmap`, without requiring any installation process. To favor portability, the *Java* programming language [19] was chosen. Portability also requires not depending of platform-specific graphical libraries, so a web interface was judged as the most suitable solution. As `nmap` do not require external features to be run, *NMapGUI* aims to be a self-contained application to maintain the same ap-

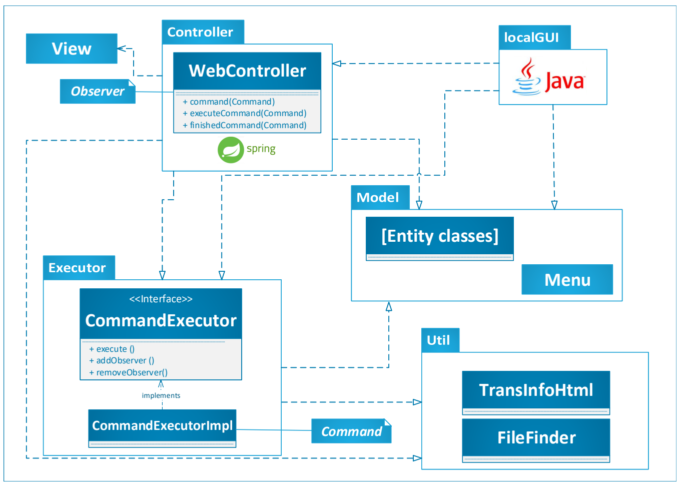


Figure 6 Architecture of *NMapGUI*.

proach. For that reason, it was developed as a web application that runs its own web server, using the *Spring* framework [46]. Lack of installation was achieved with a copy-and-run approach: users just have to copy and run a single `.jar` file into their system to use *NMapGUI*. This can be done by any user account that also has permissions to run `nmap` and *Java*, because no system file is modified.

2.4 Architecture

Current version of *NMapGUI* is composed by several packages (see Figure 6). As this is an open source project, it has been designed to be easy to understand and extend.

Executing the `.jar` file containing *NMapGUI* initially runs a small startup application created with the *Java* 1.8 *Swing* library. The classes of this application belong to the `localGUI` package. It ensures that the `nmap` tool is installed, using the features of the `Executor` package. Then, it allows users to start/stop an instance of the *Spring* application implementing the *NMapGUI* web interface. *NMapGUI* can only be used from `localhost` to prevent potential security problems via remote command execution or using the machine as a pivot to explore other networks (see Section 4). In our tests we used the version 1.5.3 of the *Spring* framework.

The architecture of the web application follows the MVC design pattern. The `View` is responsible of processing user inputs, passing the corresponding commands to the web server via HTTP requests. Data visualization is enhanced using the *jQuery* 3.2.1 [22] and the *D3* V3 [4] libraries.

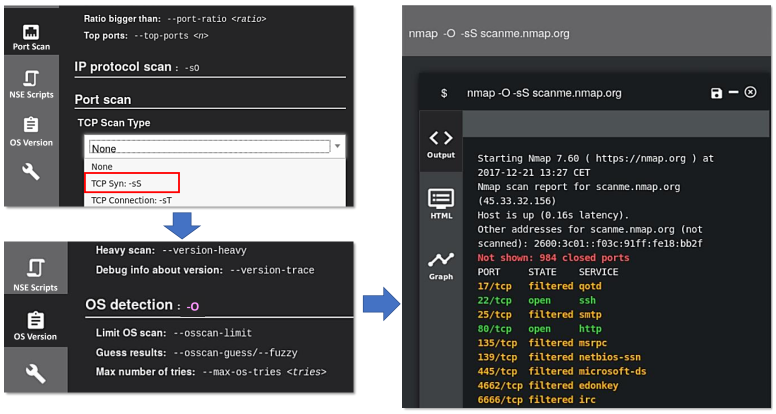
Requests are managed by the `WebController` class until they are executed. Once the user chooses to run a command, the controller calls `CommandExecutorImpl`. This class is the default implementation of the command execution processes of the application, and therefore runs each `nmap` instance with the options provided by the user. Command execution is asynchronous, to enable running `nmap` instances in parallel and to write each command output in its own window. For that reason, the `WebController` uses an *Observer* design pattern, so it gets notified every time a command finishes to update the corresponding view.

The `Model` package contains all classes representing system and scan result entities: `Address`, `Command`, `Hop`, `Host`, `Hostname`, `Port`, `Scan` or `Script`. Additionally, the `Menu` submodule contains entities representing different interface elements, like `Menu`, `Submenu`, `Category` or `Option`. The configuration of these visual elements is read from an XML file, so it can be easily updated.

Finally, the `Util` package contains utility classes to convert from XML to HTML through an XSLT file (`TransInfoHtml`), and to locate files that store command execution results (`FileFinder`). Code coverage of the application tests is controlled using the *jacoco* 0.7.9 library [13].

2.5 Usage Scenarios

This section describes how to use *NMapGUI* to run typical `nmap` scans as part of an auditing process. The first scenario requires to check the TCP reserved ports of a remote system. To avoid potential filtering, the `TCP SYN` scan type will be used. To do that, the user goes to the "Port Scan" area and, inside the "TCP Scan Type" group, locate and click the appropriate `nmap` option for this type of TCP scan (`-sS`). This way, the user must only remember the type of port scan, and *NMapGUI* provides the necessary information about the corresponding option. Then, the command can be enhanced guessing the OS type of the target system. To do that the user goes to the "OS Version" area and click on the "OS detection" option (`-O`). As described on Section 2.2, the interface automatically adds this option to the existing `nmap` command line. Once the user finishes adding all the desired options (using the GUI or manually editing the command), clicking in the "Go!" button shows the scanning progress at real time. All these steps are illustrated on Figure 7.



Ratio bigger than: --port-ratio <ratio>
Top ports: --top-ports <n>
IP protocol scan : -sO
Port scan
TCP Scan Type
None
None
TCP Syn: -sS
TCP Connection: -sT

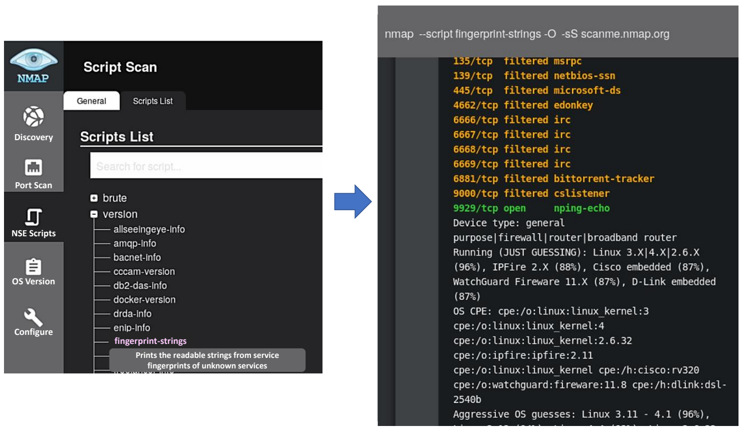
Heavy scan: --version-heavy
Debug info about version: --version-trace
OS detection : -O
Limit OS scan: --osscan-limit
Guess results: --osscan-guess/--fuzzy
Max number of tries: --max-os-tries <tries>

```
nmap -O -sS scanme.nmap.org
$ nmap -O -sS scanme.nmap.org

Starting Nmap 7.60 ( https://nmap.org ) at
2017-12-21 13:27 CET
Nmap scan report for scanme.nmap.org
(45.33.32.156)
Host is up (0.16s latency).
Other addresses for scanme.nmap.org (not
scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 984 closed ports
PORT      STATE    SERVICE
17/tcp    filtered qotd
22/tcp    open     ssh
25/tcp    filtered smtp
80/tcp    open     http
135/tcp   filtered msrpc
139/tcp   filtered netbios-ssn
445/tcp   filtered microsoft-ds
4662/tcp  filtered edonkey
6666/tcp  filtered irc
```

Figure 7 Typical basic scan process over `scanme.nmap.org`.

When the scanning process finishes, the user may decide to explore the remote machine a little further. This begins the second scenario, that uses the features of the `nmap` NSE script library to enhance the results of the previous scan. In this case, the command will incorporate a script that outputs the *fingerprints* (service and versions description strings) of any service not directly recognized by `nmap`. If the name of this script is not known, *NMapGUI* can be used to search it and perform the scan. Figure 8 shows the "NSE Scripts" area and its "Script list" tab. The desired script can be found under the "version" category, and its name is `fingerprint-strings`. Once located, the script information can be checked to be sure that it is the correct one. Finally, clicking on the script adds it to the current command line, so the enhanced scan can be run again. Although *NMapGUI* preserves the previously run `nmap` command line, this is a new scan, so its output is displayed in a new window. Therefore, it is easy to study and compare both outputs later. Additionally, if the previous scan was still running, both scans will be executed in parallel. Figure 8 illustrates this scanning process and the final `nmap` output when executed over the same target machine as the previous one.

These two examples illustrate the way users are expected to work with *NMapGUI*. First, locate the information about concrete scan options, thanks to the GUI classification. Then, check the provided option information, to ensure it contains the desired functionality and that its syntax is correct. Finally, launch the scan and wait for the process to complete, or start doing another scan while waiting for the first one.



```
nmap --script fingerprint-strings -O -sS scanme.nmap.org
135/tcp  filtered msrpc
139/tcp  filtered netbios-ssn
445/tcp  filtered microsoft-ds
4662/tcp filtered edonkey
6666/tcp filtered irc
6667/tcp filtered irc
6668/tcp filtered irc
6669/tcp filtered irc
6881/tcp filtered bittorrent-tracker
9000/tcp filtered cslistener
9929/tcp open     nping-echo
Device type: general
purpose|firewall|router|broadband router
Running (JUST GUESSING): Linux 3.X|4.X|2.6.X
(96%), IPFire 2.X (88%), Cisco embedded (87%),
WatchGuard Fireware 11.X (87%), D-Link embedded
(87%)
OS CPE: cpe:/o:linux:linux_kernel:3
cpe:/o:linux:linux_kernel:4
cpe:/o:linux:linux_kernel:2.6.32
cpe:/o:ipfire:ipfire:2.11
cpe:/o:linux:linux_kernel cpe:/h:cisco:rv320
cpe:/o:watchguard:fireware:11.8 cpe:/h:dlink:dsl-
2540b
Aggressive OS guesses: Linux 3.11 - 4.1 (96%),
```

Figure 8 Scanning `scanme.nmap.org` using NSE scripts.

3 NMapDSL

NMapGUI cannot be installed on machines that typically do not have support to display web pages, like servers and IoT devices. Because of this, *NMapDSL* is a programming language that complements *NMapGUI* to facilitate the creation of advanced `nmap` tasks in these scenarios. The language is structured in blocks modeling different parts of a scanning process. It also includes the scan scheduling, result notification, and output analysis features described on Section 2.2.4. *NMapDSL* programs are validated and translated by the *NTrans* tool to popular scripting languages. These translated scripts may be moved to any machine in the infrastructure with `nmap` (and the target scripting language runtime environment) installed. The scripts are able to run stand-alone, without dependencies of the web GUI components. We plan to support *Python 2*, *Python 3* (and also its fastest implementation, *PyPy* [49]), *Bash script*, and *Powershell* as target scripting languages to be able to run on all major operating systems. As we said, the scheduling section of *NMapGUI* generates *NMapDSL* code, so it can also work as a *NMapDSL* web IDE for external machines not able to run it.

3.1 Syntax

NMapDSL is designed as a simple and easy to understand language that facilitates performing advanced scan tasks, without requiring deep initial knowledge of `nmap` features, increasing productivity. This is also especially important in computer security courses. The language is designed around six different blocks, although several are optional. Line comments are supported using the # character. Sample programs are shown in Figure 9:

The `SCAN` block is the most important one, as it specifies a list of *named scan profiles* or a single list of `nmap` command-line options telling `nmap` what to do. If nothing is specified, `nmap` runs with all-default parameters. Additionally, a series of *scan customizing keywords* may be optionally placed before `SCAN` to run common advanced scan configurations without using options or profiles. These keywords do not cover the full range of possibilities of `nmap`, but allow users with limited experience to access advanced scan types easily:

- *Timing options*: `PARANOID`, `SNEAKY`, `POLITE`, `NORMAL`, `AGGRESSIVE`, and `INSANE`, representing values from 1 to 5 of the `-T<number>` option
- *Scan purposes*: keywords representing certain typical options, such as `STANDARD` (runs standard scan techniques, `-sC` option), `ACTIVEONLY` (only lists hosts, `-Sp` option), `AGGRESIVE` (runs more common scan options, `-A` option), or `VERSION` (obtain versions of the operating system and running services, `-O` option). Other implemented keywords also uses popular NSE scripts to achieve useful effects, like `GEO` (geolocalizes targets using the *Maxmind* database, `ip-geolocation-maxmind` script), `WHOIS` (obtain DNS information about targets, `whois` script), `MALWARE` (uses the *Google Safe Browsing* API [16] to determine if the target is identified as a malware distributor, `http-google-malware` script), `EMAIL_ADDRESS` (locates active email accounts, `http-google-email` script), or `WEB` (runs a series of scripts that specifically target web servers and CMSs, such as `http-methods` and `http-enum`).
- *IDS Evasion*: `EVASIVE`, automatically uses several `nmap` options and techniques to evade firewalls or IDSs (`-f`, `--randomize_hosts`, `--badsum`, `-D RND:10`, `--spoof-mac 0`).

This way, a valid `SCAN` block may be like this: `INSANE ACTIVEONLY SCAN <profiles or NMap options>`. Scan profiles or options have more priority in case of overlapping with the keywords.

The `ON` block is used to specify targets. It accepts lists of IPs, networks (CIDR format) or DNS names. An optional `PORT` keyword allows to restrict which ports are going to be scanned. The `EACH` block enables task scheduling functionality (see Section 2.2.4), accepting a number and a time unit (`MINUTES`, `HOURS`, `DAYS`, `WEEKS`, or `MONTHS`). If not specified, the task is run just once. The also optional `NOTIFY` block deals with scan notifications, accepting email addresses to send scan results or a file path to write them to.

The two final optional blocks deal with anomaly detection. `WITH` block describes the expected port/service distribution for targets within the range specified in the `ON` block. The `WITH ONLY` variant is similar but treats as anomalies every port found `OPEN` outside the ones specified in the script source code. Finally, the `OUTPUT CONTAINS` block allows creating simple `AND`/`OR` logical expressions (optionally prepending `NOT`) with strings, to check `nmap` output against them. Both blocks behave as the corresponding blocks described in the scheduling section (Section 2.2.4).

3.2 Validation of scripts

Unlike statically typed languages like *Java*, *C#* or *C++*, dynamically typed ones do not perform type-checking at compile time. Static typing offers the programmer early detection of type errors, making possible to fix them immediately rather than discovering them at runtime [51]. Although `nmap` it is not a programming language, usage errors are also reported at runtime. Therefore, a programming language created to model `nmap` auditing tasks also has this problem.

In order to counteract the lack of robustness due to no early type error detection, several research lines have developed different solutions applied to dynamic languages [41], or the dynamic metaprogramming features of statically-typed languages [25]. All of them proved that it is possible to increase program robustness without losing flexibility by enabling early detection of type errors. Therefore, we can also increase the robustness of *NMapDSL* programs validating types along with syntax and semantic errors prior to execution, without losing flexibility when creating audit tasks (see Figure 10). This enables us to create scan tasks with a substantially lower amount of errors. This is also a useful feature in security courses, as the detected errors will be valuable to learn how to correctly use `nmap`.

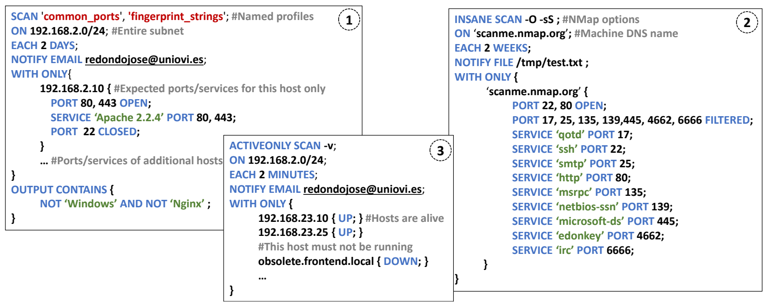
The current implementation of *NMapDSL* validates the syntax of all `nmap` options. Usage of malformed, misspelled or non-existing options will be reported along with wrong types in its parameter values. Warnings are also used; for example, with the usage of NSE scripts not installed in the current system, as the machine used to create the script may not be the same running it.

Regarding NSE scripts, every script has its own parameters and acceptable value types. Unfortunately, this information is not properly documented. Script descriptions may be extracted from its source code (parsing `description` entries), and parameter names are usually documented via

`@args` or `@arg` tags. However, parameter documentation is frequently incomplete, and there is no syntax to specify which are mandatory or its expected type. For this reason, *NTrans* includes a special folder `script_info` that contains JSON files specifying this information. This way, if we use the `dns-zone-transfer` script and a `dns-zone-transfer.json` file is present in this directory, *NTrans* will load the file to read what parameters are `mandatory` and its expected type (`integer`, `string`, `url`,...). If the corresponding `.json` file do not exist, the script source code will be parsed, and a warning will be thrown if an undocumented parameter is used (scripts frequently leave some of their accepted parameters undocumented). This way we use the available information without compromising scan flexibility, providing a mechanism to enhance script validation by developing more script info files. The initial version of *NTrans* is supplied with some `.json` files corresponding to standard NSE scripts as examples.

Additionally to syntax, multiple semantic conditions are also checked. The following are the most important ones:

- `SCAN`: `nmap` option syntax (either written directly or extracted from existing *named scan profiles*) is checked, throwing errors if they are malformed or the parameter types are invalid (see Figure 10). Each `nmap` option has its own entry in the language AST tree, so a special `Visitor` class is used to also include detailed information about the options and NSE scripts used in each translated script (see left part of the Figure 10), extracted from the official documentation. This information will be displayed when a script is run in verbose mode, helping debugging or learning processes.
- `ON`: An error is thrown if an invalid IP, DNS name or CIDR is specified. Additionally, a warning is reported if their syntax is correct, but it cannot be resolved through a DNS query during the translation process. This could be caused by a temporary network outage or because only the scanning machine has access to these targets (hence issuing a warning) but helps to mitigate potential script creation errors. The same type of validation is done every time an IP, CIDR or DNS is used, such as in the `nmap` command-line options of the `SCAN` block.
- `EACH`: The translator warns when using time units higher than 1000, considering that it is way too much time for a periodic task regardless its time unit. Only unsigned integer numbers are accepted.
- `NOTIFY`: If an email is specified, an error is thrown if the address does not comply with the email *Internet Message Format RFC2822 Stan-*



```
SCAN 'common_ports', 'fingerprint_strings'; #Named profiles
ON 192.168.2.0/24; #Entire subnet
EACH 2 DAYS;
NOTIFY EMAIL redondojose@uniovi.es;
WITH ONLY{
        192.168.2.10 { #Expected ports/services for this host only
           PORT 80, 443 OPEN;
           SERVICE ‘Apache 2.2.4’ PORT 80, 443;
           PORT  22 CLOSED;
        }
        … #Ports/services of additional hosts
}
OUTPUT CONTAINS {
        NOT ‘Windows’ AND NOT ‘Nginx’ ;
}
```

```
INSANE SCAN -O -sS ; #NMap options
ON ‘scanme.nmap.org’; #Machine DNS name
EACH 2 WEEKS;
NOTIFY FILE /tmp/test.txt ;
WITH ONLY {
        ‘scanme.nmap.org’ {
                PORT 22, 80 OPEN;
                PORT 17, 25, 135, 139,445, 4662, 6666 FILTERED;
                SERVICE ‘qotd’ PORT 17;
                SERVICE ‘ssh’ PORT 22;
                SERVICE ‘smtp’ PORT 25;
                SERVICE ‘http’ PORT 80;
                SERVICE ‘msrpc’ PORT 135;
                SERVICE ‘netbios-ssn’ PORT 139;
                SERVICE ‘microsoft-ds’ PORT 445;
                SERVICE ‘edonkey’ PORT 4662;
                SERVICE ‘irc’ PORT 6666;
        }
}
```

```
ACTIVEONLY SCAN -v;
ON 192.168.2.0/24;
EACH 2 MINUTES;
NOTIFY EMAIL redondojose@uniovi.es;
WITH ONLY {
        192.168.23.10 { UP; } #Hosts are alive
        192.168.23.25 { UP; }
        #This host must not be running
        obsolete.frontend.local { DOWN; }
        …
}
```

Figure 9 Samples of the current proposed syntax of *NMapGUI* DSL.

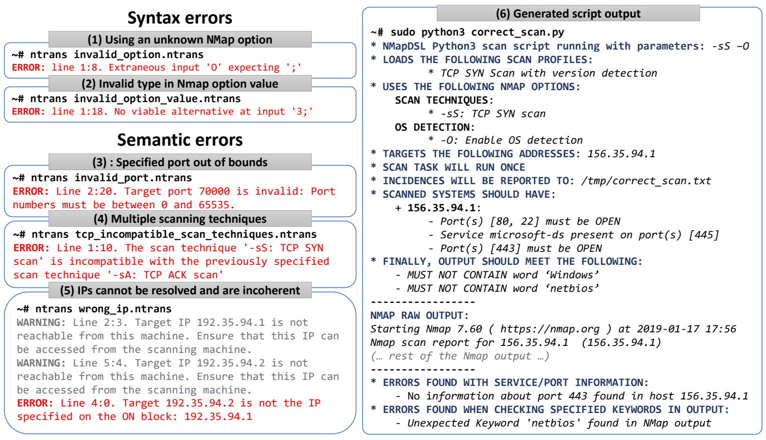
dard [18], using the validators of the *Apache Commons* project [14]. The existence of the email domain is also checked, although only a warning is reported if not found for the same reasons we did with IPs or DNS names. If a file is specified, its path is checked and a warning is issued if the specified file is found or will be overwritten, again considering that the scanning machine might be different than the script creation one.

- `WITH`: Validation of IPs and DNS names is identical to the described in the `ON` block. Port numbers are forced to be unsigned integers in the 1..65536 range. Additionally, coherency is checked between the targets specified in the `ON` block and the ones present in the `WITH` block. For example, if a CIDR is specified in the `ON` part, and IP addresses are used in this block, every specified IP is checked to be within the CIDR range. Moreover, if the `ON` block uses a DNS name and this block uses and IP (or vice versa), both are checked to see it they match.

Some of these conditions can only be checked if a net connection is available in the machine running `NTrans`, giving a warning otherwise. Validation result examples can be shown on the right part of Figure 10

3.3 Scripts samples and output

Figure 9 shows sample programs of the current specification of *NMapDSL*. Program 1 contains most of the scheduling modeled on Figure 4: a subnet is monitored every two days to check opened ports on certain machines and ensuring that no machine reports *Windows* or *Nginx* systems installed. In a real scenario this may detect unwanted servers or rogue machines. Program 2 scans the machine *scanme.nmap.org* and ensures that a list of expected port and service names is strictly observed. In a real scenario, not fulfilling this



Syntax errors
(1) Using an unknown NMap option
(2) Invalid type in Nmap option value
Semantic errors
(3) : Specified port out of bounds
(4) Multiple scanning techniques
(5) IPs cannot be resolved and are incoherent
(6) Generated script output

```
~# ntrans invalid_option.ntrans
ERROR: line 1:8. Extraneous input 'O' expecting ';'
```

```
~# ntrans invalid_option_value.ntrans
ERROR: line 1:18. No viable alternative at input '3;'
```

```
~# ntrans invalid_port.ntrans
ERROR: Line 2:20. Target port 70000 is invalid: Port
numbers must be between 0 and 65535.
```

```
~# ntrans tcp_incompatible_scan_techniques.ntrans
ERROR: Line 1:10. The scan technique '-sS: TCP SYN
scan' is incompatible with the previously specified
scan technique '-sA: TCP ACK scan'
```

```
~# ntrans wrong_ip.ntrans
WARNING: Line 2:3. Target IP 192.35.94.1 is not
reachable from this machine. Ensure that this IP can
be accessed from the scanning machine.
WARNING: Line 5:4. Target IP 192.35.94.2 is not
reachable from this machine. Ensure that this IP can
be accessed from the scanning machine.
ERROR: Line 4:0. Target 192.35.94.2 is not the IP
specified on the ON block: 192.35.94.1
```

```
~# sudo python3 correct_scan.py
* NMapDSL Python3 scan script running with parameters: -sS -O
* LOADS THE FOLLOWING SCAN PROFILES:
        * TCP SYN Scan with version detection
* USES THE FOLLOWING NMAP OPTIONS:
    SCAN TECHNIQUES:
        * -sS: TCP SYN scan
    OS DETECTION:
        * -O: Enable OS detection
* TARGETS THE FOLLOWING ADDRESSES: 156.35.94.1
* SCAN TASK WILL RUN ONCE
* INCIDENCES WILL BE REPORTED TO: /tmp/correct_scan.txt
* SCANNED SYSTEMS SHOULD HAVE:
    + 156.35.94.1:
        - Port(s) [80, 22] must be OPEN
        - Service microsoft-ds present on port(s) [445]
        - Port(s) [443] must be OPEN
* FINALLY, OUTPUT SHOULD MEET THE FOLLOWING:
    - MUST NOT CONTAIN word ‘Windows’
    - MUST NOT CONTAIN word ‘netbios’
-----------------
NMAP RAW OUTPUT:
Starting Nmap 7.60 ( https://nmap.org ) at 2019-01-17 17:56
Nmap scan report for 156.35.94.1  (156.35.94.1)
(… rest of the Nmap output …)
-----------------
* ERRORS FOUND WITH SERVICE/PORT INFORMATION:
    - No information about port 443 found in host 156.35.94.1
* ERRORS FOUND WHEN CHECKING SPECIFIED KEYWORDS IN OUTPUT:
    - Unexpected Keyword 'netbios' found in NMap output
```

Figure 10 Output examples of scripts generated with *NMapDSL*.

might mean that the machine has been compromised. Finally, program 3 just checks a network for a series of machines. Extra ones in the list may warn about rogue or unwanted machines (an old server is accidentally brought up) in a network. Less may also indicate that a server has failed or have been misconfigured and unable to run, detecting outages.

Figure 10 (left) shows the typical output of scripts translated by `NTrans` and also examples of errors detected by `NMapDSL` (right). Apart from the syntax errors we mentioned, the error output of using illegal port numbers (3) or incoherent/unreachable IPs (5) we mentioned in subsection 3.2 is shown. Additionally, usage of incompatible options is also detected (4), as `nmap` cannot use multiple scan techniques in a single scan. Figure 10 (right) shows the step by step detailed information provided by translated scripts about the purpose of every option and NSE script used, apart from the `nmap` raw output.

4 Reviews

We released a preview version of *NMapGUI* implementing most of the described functionality to obtain feedback from the community. This preview version still lacks support to create or load *named scan profiles*, although the file format is defined and supported by the current release of *NMapDSL*. It does not integrate yet the DSL validation capabilities for `nmap` options and values we saw in the previous section. Finally, the scheduling interface under construction, and therefore not available in the main scan screen yet.

Several security-related websites made articles reviewing this initial release of *NMapGUI* [1,28,54,56,58,61]. Reviews praise the interface design and capabilities, especially the ability to execute parallel commands and the integrated information about `nmap` options and NSE scripts. Main concern about *NMapGUI* has been the potential security problems that may occur if a malicious user accesses the interface from an external machine, using the scanning machine as a pivot to access other networks. This is the reason why we decided to restrict connections to the GUI to `localhost`.

User feedback obtained from social networks also shows a positive reaction, praising *NMapGUI* features. However, some users state that `nmap` should only be used manually as a command-line tool due to their complexity and number of options, as a GUI could not capture its complexity, as happens with *Zenmap* (see Section 2.1). However, mastering `nmap` is difficult precisely because its flexibility and complexity, hence the motivation of creating *NMapGUI* to lower it. Advanced users can manually modify `nmap` commands in *NMapGUI* if they wish to do so. This way, they can take advantage of the GUI features while being in full control of the command structure, as they requested. Additionally, for those users that just use command-line tools, using only *NMapGUI* to configure *NMapDSL* auditing tasks could be a valid alternative.

A group of users that have reviewed the product also questions the need of a new GUI, as an official one exists. *NMapGUI* is an alternative to the official one, designed to try to solve its deficiencies and increase user productivity. When creating *NMapGUI*, we primarily target occasional users, students or users with limited `nmap` experience. We think that this GUI facilitates learning how to use `nmap` properly in very different scenarios and provides adequate and more efficient ways to explore the different possibilities of the tool. Finally, *Rawsec's Cybersecurity Inventory* has added *NMapGUI* as one of their recommended tools, so it can be easily found by more security professionals [64]

5 Conclusions

The work described in this paper enables users with different experience levels to make better use of the `nmap` security auditing capabilities with less effort. *NMapGUI* is a web application that improves the official `nmap` GUI, facilitating its usage and solving all its shortcomings. The full set of `nmap` options is classified in four areas, so they can be located and used easier. This also applies to the extensive library of NSE scripts, providing a search engine and enhanced information about them. Ability to run multiple scanning tasks in parallel increases user productivity. We have also provided means to directly include the GUI output into *pentesting* reports.

NMapDSL is powerful tool to develop robust and advanced scanning tasks easier, thanks to its early validation of type errors and direct support for common useful `nmap` features, even without the web GUI. Early error validation of scripts covers critical areas of the scanning process, so the probability of generating erroneous scripts decreases. The detailed information displayed by translated scripts about its purpose is also a valuable learning and debugging tool, as the user knows exactly what it is running. Also, modeling scan tasks in a simple DSL eases sharing them between systems or users with different skill levels, which also is a valuable addition to use in security courses. We believe that our work may increase the productivity of `nmap` users, ease the design of auditing tasks, the analysis of their results, and enabling some user types, like students, to better understand and effectively use the wide set of capabilities of this highly popular security tool.

We are planning to use these tools in higher education security-related courses, along with other complementary tools ([50], [27]). First users will be the *Computer Security* course of the *Software Engineering* degree [8], along with the *Web Security Systems* and *Web Server Administration* courses of the *Master on Web Engineering* [10], all from the *University of Oviedo*. Some courses of the future *Master on Cybersecurity in Software Engineering* [9] (also from the same university) may also benefit from using them. The purpose of this is twofold: gathering feedback to improve the software and facilitate the usage of `nmap` as part of their contents. We are also studying the addition of more options to the interface, such as enhanced reporting options, one-click scan to intermediate nodes in the interactive node graph with the same scan profile as the target one, and ways to compare scan results performed to the same machine.

Finally, the next evolution of *NMapDSL* will expand language possibilities by enabling generated scripts to automatically calculate a security rating of each scanned target. This will be done extracting each service type and version information, using the `cve-search` project features [52] to search for corresponding CVEs (*Common Vulnerability Exposures*) containing vulnerabilities of the products found. This way, the users will have immediate feedback about the vulnerability of each scanned machine, also integrating the features of the `CVE-Scan` [33] project. This can also be used to prioritize the machines or infrastructure parts with more critical problems to implement corrective actions. We are also exploring the possibility of applying DSL with early validation of type errors to model the complexity or other security tools with similar complexity.

The current version of *NMapGUI* can be downloaded from `https://github.com/danicuestasuarez/NMapGUI`. At the date of writing this paper, the project has earned 220 *GitHub* stars, 62 users have cloned it in their own *GitHub* account, and 23 have subscribed to project updates and notifications. An alpha version of the *NTrans* translation tool, able to output *Python* language scripts (versions 2 and 3, also compatible with *PyPy* 2 and 3), can be downloaded from `https://www.dropbox.com/s/du0rf0u75k28ny4/ntrans.alpha.v01.zip`. The distribution also include the full source code, tests, ANTLR [44] grammar, documentation, and sample scripts. Once the DSL is finished, both projects will merge into a single product on the first repository.

6 Related Work

Additionally to the `nmap` official GUI (*Zenmap* [39]), which is an evolution to a previous GUI called *NMapFE*, there are several desktop-based projects that created a `nmap` GUI to facilitate its usage to inexperienced users or increase their productivity. *Nmap GUI* [34] simplifies performing simple scanning tasks by just specifying a target a scan profile, like the *Zenmap* approach. It also mimics the `nmap` output and allows the organization of scans using timestamps.

NMapWin [29] is another GUI with similar capabilities, although aimed only to `Win32` platforms, with an outdated interface and no longer maintained by their authors. A more advanced GUI for `nmap` is *NmapSi4* [5], that targets the *Qt* library. This GUI offers a series of options to use `nmap` in an organized way, grouping the different capabilities of the tool in a set of sections, similar to the approach followed by *NMapGUI*. Another similar GUI is *NMapW* [63], a *Windows* tool that organizes the main `nmap` options in a category tree like the one we used in *NMapGUI*, but do not have explicit support for NSE scripts. It also allows users to store different scan configurations.

There are also web-based GUIs for `nmap`. *WMap* [62] allows the user to execute a limited set of common `nmap` commands easily, listing them along with information about its expected output. Additionally, *Nmap-webgui* [53], currently under development, is a project that aims to give users advanced capabilities to handle `nmap` scans, like scan scheduling, comparison of results using `diff`, and review of scan results. *nmap-cgi* [11] web GUI allows the usage of typical `nmap` options (scan IPs, subnets, OS detection...) through a very clear and simple interface. It incorporates user and group management, so it can assign different rights over individual `nmap` options to them. It also incorporates three types of scans (single, scheduled and periodic) and exportable output format in XML. Another example is *WebMap* [57], a *Django* `nmap` web interface currently under development with several features similar to *NMapGUI*, such as the `traceroute` graph. It imports and parses `nmap` XML generated files instead of working with its raw output, and also allows to run and schedule auditing tasks from its dashboard, with a focus on reporting features and interacting with external applications. It does not provide the same degree of information than *NMapGUI* about `nmap` options and scripts to maximize its usage potential, and also requires Docker installed to run. None of these products offer early type error detection and validation features comparable to the ones implemented in *NMapDSL*.

We can also find GUIs for `nmap` targeting *Android* devices. Examples are *nmap-gui* [12], *Cydia Tweak Nmap GUI* [17] and the lightweight *pentesting* tool *Lightpen* [27] that, although does not allow to directly control `nmap` options, it executes some of its most common scan types as part of its functionalities.

Finally, the `nmap` tool is also an integral part of more powerful and popular *pentesting* tools such as *OpenVAS* [40], *Metasploit* [48] or the *IVRE Network Scanning and Analysis* tool [26]. However, the complexity of these tools abstracts the usage of `nmap` within the *pentesting* processes they perform, so users have no direct control over every specific `nmap` option that it is used.

7 Acknowledgements

This work was supported by the European Union, through the European Regional Development Funds (ERDF), and the Principality of Asturias, through its Science, Technology and Innovation Plan (grant GRUPIN14-100). The authors have also received funds from the Banco Santander through its support to the Campus of International Excellence.

References

[1] 1024Megas. NMapGUI - Graphical User Interface. `http://www.1024megas.com/2017/09/nmapgui.html`, 2017. (Jan 30, 2019).

[2] AlienVault. Alienvalut: Threat detection, incident response product. `https://www.alienvault.com/`, 2018. (Jan 30, 2019).

[3] Taylor Armerding. The 17 biggest data breaches of the 21st century. http://www.csoonline.com/article/2130877/data-breach/the-16-biggest-data-breaches-of-the-21st-century.html, 2018. (Jan 30, 2019).

[4] Mike Bostock. D3: Data-driven documents. https://d3js.org/, 2018. (Jan 30, 2019).

[5] Francesco Cecconi. Nmapsi4. https://github.com/nmapsi4/nmapsi4, 2015. (Jan 30, 2019).

[6] Checkpoint. Live cyber attack threat map. http://threatmap.checkpoint.com/ThreatPortal/livemap.html, 2018. (Jan 30, 2019).

[7] Jieren Cheng, Jinghe Zhou, Qiang Liu, Xiangyan Tang, and Yanxiang Guo. A ddos detection method for socially aware networking based on forecasting fusion feature sequence. *The Computer Journal*, 61(7):959–970, 2018.

[8] Universidad de Oviedo. Escuela de ingeniería informática. https://ingenieriainformatica.uniovi.es/infoacademica/grado/, 2018. (Jan 30, 2019).

[9] Universidad de Oviedo. Master en ciberseguridad en ingenieria del software. https://www.mcis.uniovi.es/, 2018. (Jan 30, 2019).

[10] Universidad de Oviedo. Máster universitario en ingeniería web. https://ingenieriainformatica.uniovi.es/infoacademica/masterydoctorado, 2018. (Jan 30, 2019).

[11] Julien Delange. nmap-cgi project. http://nmap-cgi.tuxfamily.org/, 2006. (Jan 30, 2019).

[12] Fernando Dominguez. Nmap-gui. https://github.com/FernandoDoming/nmap-gui, 2017. (Jan 30, 2019).

[13] ECL Emma. Jacoco java code coverage library. http://www.eclemma.org/jacoco/, 2018. (Jan 30, 2019).

[14] Apache Foundation. Apache commons. https://commons.apache.org/, 2018. (Jan 30, 2019).

[15] Yossi Gilad and Amir Herzberg. Off-path tcp injection attacks. *ACM Trans. Inf. Syst. Secur.*, 16(4):13:1–13:32, April 2014.

[16] Google. Google safe browsing. https://safebrowsing.google.com/, 2018. (Jan 30, 2019).

[17] IDroid.us. Cydia tweak nmap gui. https://web.archive.org/web/20121030090623/https://idroid.us/cydia-tweak-nmap-gui-0-93.html, 2012. (Jan 30, 2019).

[18] IETF7. Internet message format. https://tools.ietf.org/html/rfc2822, 2001. (Apr, 2001).

[19] Oracle Inc. Java programming language. https://www.java.com/en/download/, 2018. (Jan 30, 2019).

[20] Infosec Institute. Icmp reverse shell. http://resources.infosecinstitute.com/icmp-reverse-shell/, 2014. (Jan 30, 2019).

[21] Linux Journal. Editors' choice awards. https://www.linuxjournal.com/article/5525, 2001. (Jan 30, 2019).

[22] jQuery Foundation. jquery: write less, do more. https://jquery.com/, 2018. (Jan 30, 2019).

[23] Dan Kaplan. Don't sleep on web applications: The 5 most common attack types and how to better defend them. https://www.trustwave.com/en-us/resources/blogs/trustwave-blog/don-t-sleep-on-web-applications-the-5-most-common-attack-types-and-how-to-better-defend-them/, 2016. (Jan 30, 2019).

[24] Daljit Kaur and Parminder Kaur. Empirical analysis of web attacks. *Procedia Computer Science*, 78:298 – 306, 2016. 1st International Conference on Information Security & Privacy 2015.

[25] Ignacio Lagartos, Jose Manuel Redondo, and Francisco Ortin. Towards a java library to support runtime metaprogramming. In Ernesto Damiani, George Spanoudakis, and Leszek Maciaszek, editors, *Evaluation of Novel Approaches to Software Engineering*, pages 224–242, Cham, July 2018. Springer International Publishing.

[26] Pierre Lalet. Ivre official web page. https://ivre.rocks/, 2018. (Jan 30, 2019).

[27] Iñigo Llaneza. Lightpen: Modular mobile pentesting tool. https://github.com/LlanezaAller/Lightpen, 2018. (Jan 30, 2019).

[28] Sergio De Luz. NMapGUI: Conoce esta interfaz grafica de Nmap basada en Java. https://www.redeszone.net/2017/09/03/nmapgui-conoce-esta-interfaz-grafica-de-Nmap-basada-en-java/, 2017. (Jan 30, 2019).

[29] Gordon Lyon and Jens Vogt. Nmapwin. https://sourceforge.net/p/nmapwin/wiki/Home/, 2002. (Jan 30, 2019).

[30] ICT Security Magazine. 2017 data breach investigations report 10th edition. http://www.ictsecuritymagazine.com/wp-content/uploads/2017-Data-Breach-Investigations-Report.pdf, 2018. (Jan 30, 2019).

[31] Tim Matthews. Imperva incapsula survey: What ddos attacks really cost businesses. Imperva incapsula technical report, Imperva Incapsula, https://lp.incapsula.com/rs/incapsulainc/images/eBook\%20-\%20DDoS\%20Impact\%20Survey.pdf, 2014.

[32] Charles McLellan. Cybersecurity predictions for 2016: How are they doing? http://www.zdnet.com/article/cybersecurity-predictions-for-2016-how-are-they-doing/, 2016. (Jan 30, 2019).

[33] Pieter-Jan Moreels. Cve scan. https://github.com/NorthernSec/CVE-Scan/, 2018. (Jan 30, 2019).

[34] Ole Morten. Nmap gui. https://sourceforge.net/projects/nmapgui/, 2016. (Jan 30, 2019).

[35] Nmap.org. Nmap in the movies. https://nmap.org/movies/, 2018. (Jan 30, 2019).

[36] Nmap.org. Nmap in the news (and press). https://nmap.org/nmap_inthenews.html, 2018. (Jan 30, 2019).

[37] Nmap.org. Nmap nse scripts official documentation. http://nmap.org/nsedoc/, 2018. (Jan 30, 2019).

[38] Nmap.org. Nmap: The network mapper. http://nmap.org/, 2018. (Jan 30, 2019).

[39] Nmap.org. Nmap zenmap official gui. https://nmap.org/zenmap/, 2018. (Jan 30, 2019).

[40] OpenVAS. Openvas open source vulnerability scanner and manager. http://www.openvas.org/, 2018. (Jan 30, 2019).

[41] Francisco Ortin, Baltasar G. Perez-Schofield, and Jose Manuel Redondo. Towards a static type checker for python. In *European Conference on Object-Oriented Program-*

ming (ECOOP), Scripts to Programs Workshop, STOP, volume 15, pages 1–2, Prague, Czech Republic, July 2015. ECOOP.

[42] OWASP. Testing for error code (otg-err-001). `https://www.owasp.org/index.php/Testing_for_Error_Code_(OTG-ERR-001)`, 2014. (Jan 30, 2019).

[43] OWASP. Penetration testing methodologies. `http://www.owasp.org/index.php/Penetration\_testing\_methodologies`, 2016. (Jan 30, 2019).

[44] Terence Parr. Antlr (another tool for language recognition). `http://www.antlr.org/`, 2018. (Jan 30, 2019).

[45] Cheryl Pellerin. Cybercom: Pace of cyberattacks have consequences for military. `https://www.defense.gov/News/Article/Article/1192583/cybercom-pace-of-cyberattacks-have-consequences-for-military-nation/`, 2017. (Jan 30, 2019).

[46] Pivotal. Spring framework. `https://spring.io/`, 2018. (Jan 30, 2019).

[47] Rapid7. Common types of cybersecurity attacks: A look inside the attacker's toolkit. `http://www.rapid7.com/fundamentals/types-of-attacks/`, 2018. (Jan 30, 2019).

[48] Rapid7. Metasploit: The world's most used penetration testing framework. `https://www.metasploit.com/`, 2018. (Jan 30, 2019).

[49] J. M. Redondo and F. Ortin. A comprehensive evaluation of common python implementations. *IEEE Software*, 32(4):76–84, July 2015.

[50] Jose Manuel Redondo and Leticia del Valle. Filesync and era literaria: Realistic open sourcewebs to develop web security skills. *Journal of Web Engineering*, 17(5):1–22, 2018.

[51] Jose Manuel Redondo and Francisco Ortin. Efficient support of dynamic inheritance for class- and prototype-based languages. *Journal of Systems and Software*, 86(2):278 – 301, 2013.

[52] Wim Remes, Alexandre Dulaunoy, and Pieter-Jan Moreels. A tool to perform local searches for known vulnerabilities. `https://github.com/cve-search/cve-search/`, 2018. (Jan 30, 2019).

[53] Ronald Savon. Nmap-webgui. `https://github.com/savon-noir/nmap-webgui`, 2013. (Jan 30, 2019).

[54] Div Security. NMapGUI: Interfaz gráfica de usuario para Nmap. `http://security.divdesign.mx/nmapgui-interfaz-grafica-de-usuario-para-nmap/`, 2017. (Jan 30, 2019).

[55] HelpNet Security. Rise of cyber attacks against the public sector. `http://www.helpnetsecurity.com/2016/09/23/cyberattacks-public-sector/`, 2016. (Jan 30, 2019).

[56] Homputer Security. Découvrez NMapGUI la version graphique de Nmap. `http://homputersecurity.com/2017/10/26/decouvrez-nmapgui-la-version-graphique-de-nmap/`, 2017. (Jan 30, 2019).

[57] Rev3rse Security. Webmap: Nmap dashboard and reporting. `https://github.com/Rev3rseSecurity/WebMap`, 2019. (Jan 30, 2019).

[58] Penetration Testing: Security Training Share. NMapGUI: Advanced Graphical User Interface for Nmap. `https://securityonline.info/nmapgui-advanced-graphical-user-interface-nmap/`, 2017. (Jan 30, 2019).

[59] Richard Shay, Saranga Komanduri, Adam L. Durity, Phillip (Seyoung) Huh, Michelle L. Mazurek, Sean M. Segreti, Blase Ur, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. Designing password policies for strength and usability. *ACM Trans. Inf. Syst. Secur.*, 18(4):13:1–13:34, May 2016.

[60] Mark Smith. Huge rise in hack attacks as cyber-criminals target small businesses. `http://www.theguardian.com/small-business-network/2016/feb/08/huge-rise-hack-attacks-cyber-criminals-target-small-businesses`, 2016. (Jan 30, 2019).

[61] StackTrender. Nmap GUI Java / Web Front End for Nmap - YouTube. `https://stacktrender.com/post/st/nmap-gui-java-web-front-end-for-nmap-youtube`, 2017. (Jan 30, 2019).

[62] Eric Suarez. Wmap. `https://github.com/ericsuarez/wmap`, 2017. (Jan 30, 2019).

[63] Syhunt. Nmapw: Free port scanner for analyzing network security or internet exploration. `http://nmapw.software.informer.com/`, 2018. (Jan 30, 2019).

[64] Alexandre Zanni. Rawsec's cybersecurity inventory: An inventory of tools and resources about cybersecurity. `http://inventory.rawsec.ml/tools.html`, 2018. (Jan 30, 2019).