ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/318093281

A Raspberry Pi Attacking Guide

Thesis · June 2017

CITATIONS
5

READS
9,245

1 author:

Ryan Murray
University of Westminster
1 PUBLICATION **5** CITATIONS
SEE PROFILE

All content following this page was uploaded by Ryan Murray on 10 July 2017.

The user has requested enhancement of the downloaded file.

A Raspberry Pi Attacking Guide

Ryan Murray
Cyber Security Group,
Department of Computer Science
University of Westminster, UK
w1493304@westminster.ac.uk

***Abstract*—The focus of this paper is to explore the ability of the Raspberry Pi using Kali Linux to foresee penetration testing techniques that can be performed. The Pi has low processing power compared to average computers so certain penetration testing would need to be done on a more powerful computer. This project will show how flexible the Raspberry Pi can be as a penetration testing platform by carrying out the attacks detailed below.**

A Man-in-the-middle attack is when an attacker adversary $\mathcal{ADV}$ places herself in the middle of communications between two parties. The two parties involved are unaware of this and believe they are only communicating to each other. For this attack, it is necessary that the $\mathcal{ADV}$ is on the same network as the target(s). Once in between the victim and the router it will be possible to gather different information. The consequences of this attack can result in data being stolen and monetary loss all depending on what information has been gathered whilst being in the middle.

A Denial of Service attack is when an attacker intentionally attempts to stop legitimate users from accessing services. An attacker can flood a network with information to overload a server so it would stop processing their services to legitimate users. The consequences here are that the services will become unavailable and legitimate users will not be able to access the underlying services. This attack is common and highly effective if trying to deny service to a target.

***Keywords*—Vulnerabilities; IoT Security; Ethical Hacking;**

I. INTRODUCTION

The Internet of Things (IoT), is the concept of setting up everyday appliances, objects and wearable devices with an Internet connection to allow the exchange of data and information. A Thing of washing machines that send you a text when the load is done, or air conditioning you can turn on from an app on your phone. The technology has gained a lot of favourable attention and is becoming increasingly popular. IoT is becoming more extensive than ever within technology. It is more prevalent in millions of households and companies. Due to its low cost nature these devices are gaining in popularity.

The expansion across the world of the IoT and internet technologies has caused a need for an increase in security. Like any new technology, IoT can open new potential vulnerabilities or opportunities. More precisely, it has been observed that IoT has become particularly attractive to cybercriminals who have shown a growing interest in targeting IoT devices, or leveraging the growing ubiquity of IoT devices in order to perform several attacks. Many businesses are dependent on technology and need to have sufficient security defences in place to protect themselves from both internal and external attacks.

This project is focused on IoT and its security related issues using penetration testing to exploit vulnerabilities in existing systems. As attacks involving IoT devices are gaining a lot of popularity. In the past, lessons have been learned regarding security issues in devices like Bluetooth. However, improvements that increase the level of security in such environments have been made. As IoT develops it must improve in the security aspects and lessons need to be learnt from past mistakes. When Bluetooth was new it had issues and it needed to develop. As it developed then so did the security. As the security problems with IoT is now showing the same issues, it is having to deal with this all over again [1] Knowing from the past the IoT should be addressing and focusing on its security problem at present.

Not many years ago, eHealth was seen as an expenditure rather than an investment. During the last decade this has changed significantly, to the extent that eHealth has moved to the top of the development agenda not only for private organizations but also for public administration bodies that have spurred the development of eHealth [2]. To this end, we have seen a steady increase in research focus and funding aiming to modernize existing healthcare systems and to provide reliable and cost effective eHealth services [3], [4]. As a result, nowadays we are faced with a major technological upturn of an industry that for many years has relied on handwritten records and now is expanding at a phenomenal rate.

As adoption of ehealth solutions advances, new computing paradigms - such as cloud computing [5], [6], [7] - bring the potential to improve efficiency in managing medical health records, help reduce costs [8] and support the collaboration between different organizations. However, placing patients' data in remote locations raises many concerns regarding the privacy of users' data.

Lately we have seen some significant developments in cloud computing [9]. Researchers have been focusing not only on creating efficient and flexible cloud-based services [10], [11], [12] but a lot of attention is being invested in designing systems that are secure against various malicious behaviours and attacks [13], [14]. As a result, many companies and individuals have been starting using cloud-based services with main aim to improve their productivity as well as the collaboration between the users.

When it comes to the health industry, many companies and organizations have already migrated their services to the cloud. In addition to that, many of the existing cloud-based services are enhanced with security-related mechanisms that provide the necessary guarantees for the protection of patients' data. However, there is a clear set of mechanisms that will allow different organizations (e.g. hospitals) to host patients' data in different clouds in order to securely share health records. Having such mechanisms can increase the productivity of health practitioners since a patients medical records can be easily transferred to a different hospital. In addition to that, such a sharing functionality has the potential to better support research and enhance the collaboration between specialists and scientists that are geographically apart. This can be exhibited by doctors who have certain specialities can be granted rights to be able to examine records of patients that exhibit certain symptoms. The ease of transferring patients' medical records between hospitals will enable the transferring of patients' care from one hospital to another will alleviate the requirements to repeat investigations.

A. Our Contribution

In this paper, we propose HealthShare – a forward-looking approach for secure ehealth data sharing between multiple organizations that are hosting patients' data in different clouds. The proposed protocol is based on a Revocable Key- Policy Attribute-Based Encryption scheme and allows users to share encrypted health records based on a policy that has been defined by the data owner (i.e. patient, a member of the hospital, etc). Furthermore, access to a malicious or compromised user/organization can be easily revoked without the need to generate fresh encryption keys. We hope that this work will give valuable insights to the designers of cloud-based eHealth services in order to help design and develop mechanisms that will support data sharing in a multi-cloud environment.

B. Organization

The remainder of this paper is organized as follows: In Section II, we describe the most important relevant works that have been published in the area while in Section III, we present the main entities that participate in our model and we proceed by defining the problem statement. In Section IV, we describe our tool as well as the implemented attacks including a step by step guide. In Section V, we show a technical overview of the system including coding techniques learned and implemented. Finally, in Section VI we conclude the paper and we also present some future steps regarding the enhancement of our tool in order to support more attacks.

II. Related Work

This section presents related works that have been done in the same area. In addition to that, we briefly describe how each tool works and we present a list of advantages and drawbacks. Finally, for each described related work we provide a brief comparison with our approach in order to highlight the differences and clearly present the contribution of our work.

Ettercap – GUI [15] is a man-in-the-middle attack suite that can sniff packets and dissect many protocols. It also includes many features for network and host analysis. Ettercap – GUI has a simple graphical user interface that is easy to use. In Ettercap – GUI, the user needs to edit configuration files and settings to be able to implement correctly a man-in-the-middle attack. After setting up configuration and initiating an attack the tool is functional. As security is always evolving there are new measures that have been put in place to prevent tools using attacks like these. It is important to have a well-maintained application and be adaptable to new security mechanisms. This application has been out a long time and the techniques used to implement attacks are now out dated and partially non-functional. After we tested this tool on old versions of operating systems it showed us how well the tool functioned previously. The application is not being updated frequently therefore, giving the users an unreliable service and causing problems when attempting live attacks. This is a major drawback on the application and making it less popular.

The similarities between our application and Ettercap – GUI is that they both have a simple and easy to use graphical interfaces and both implement types of Man-in-the-Middle attacks. After testing Ettercap – GUI we discovered that the configuration needed prior to the attack was an inconvenience for the user. Having to change different configuration file prior to each attack meant the user spent more time preparing an attack than actually running one. After using Ettercap – GUI, we decided to make our application as user friendly as possible. Making it simple and easy to use with little to none configuration needed. Taking into consideration the poor functionality of Ettercap – GUI we needed to ensure our application had relevant implementation to bypass current security measures.

Subterfuge [16] is a Man-in-the-Middle framework that automates the process of various attacks. It is a web based application that offers different attack vectors. The graphical user interface gives the user different attacks to pick from and provides more than just Man-in-the-Middle attack. It also provides code injection and Denial-of-Service. This application gives many different attack vectors which is an advantage when wanting to use a single framework that automates processes for the user. The drawback on this is that Subterfuge, has too many options on the application and at first glance it can be overwhelming and difficult for the user to know which attack to implement, how to implement it and might possibly not have the knowledge of all the attack vectors on offer. After using Subterfuge application, we decided to give our graphical user interface a basic setup giving easy navigation for users and not make it complex so it is usable for a wider audience.

III. System Model and Problem Statement

In this section, we will be discussing the system model. More precisely, we will walk through each element of the system with a brief overview for each.

a) Sensor Nodes:: The platform environment used to complete the penetration testing is using a group of sensor nodes that are connected to the same network. In our implementation we will be explicitly using a set of Raspberry Pi sensor devices. The set of all Raspberry Pi devices that are participating in our framework is denoted by $\mathcal{P} = \{p_1, \dots, p_n\}$, where p_i is a raspberry pi with a unique identifier i. Furthermore, for the needs of our tool we assume that all nodes can communicate with each other while the unique identifier of each node is known to all members of $\mathcal{P}$.

b) Master Node (p_m):: Let $p_m \in \mathcal{P}$ denote the master node. The master node is responsible for initiating a driving attack. To do so, p_m will first have to locate the target that it wishes to attack and then create a list of zombie nodes that will be supporting p_m in order to successfully implement the attack. The master node will be running Kali Linux ARM which is an open source penetration toolkit.

c) Target (u_t):: Let $\mathcal{U} = \{u_1, \dots, u_m\}$ be the set of all legitimate devices that are connected to the same network as the members of $\mathcal{P}$. Then, the device u_t that has been selected by p_m to launch an attack will be called the victim or target node. Regarding the capabilities u_i's our framework does not make any assumption apart from the legitimate argument that each u_i must be connected in the same network as p_m and must be discoverable.

d) Slave Nodes:: The master node p_m chooses an arbitrary number of nodes out of those in set $\mathcal{P}$. Let $\mathcal{P}_{\mathcal{Z}} \subseteq \mathcal{P}$ denote the set of all nodes that were chosen by p_m. Each $p_z \in \mathcal{P}_{\mathcal{Z}}$ will be used by p_m as a slave node with main purpose to successfully perform an attack on u_t. Slave nodes our also known as *zombies*. A zombie can be also considered as a node that has been compromised p_m and acts maliciously on behalf of the master node [17].

e) Attack Engine: : The attack engine is responsible for automating the process of the attack from given input. The attack engine will run on p_m by receiving specific input and generate the attack on u_t using sub processes. Whilst the attack engine is processing it will store any live data gathered throughout the attack into a database and display.

IV. Attack Tool

In this section, we will be discussing and implementing various attacks using the master node p_m. More precisely, we will walk through all the procedures taken to implement each attack and the commands used during the attack process. The tools used for these attacks can be found on Kali Linux and have been optimised for the Raspberry pi.

A. Man-in-the-Middle Attack

The main purpose of this attack, is to gather the credentials of users who are trying to access popular websites that require users to enter a username and a password. The attack can be only implemented on users who are connected on a local area network and we achieve the attack by using different tools and configurations. Figure 2 shows an overview of the attack.

During the initialisation phase, the master node p_m selects a website that wishes to steal credentials from. Then, p_m selects a list of slave nodes. After the initialisation phase, p_m needs to prepare the necessary environment that will allow her to successfully launch the attack. To this end, p_m uses the SET [18] tool and clones the website that was selected during the initialisation step. Once the cloning is finished, the cloned website is automatically hosted onto the local host of p_m as a running Apache server. Once the cloning procedure has been successfully completed, our attacking tool listens for any input from the local host server. Effectively the local host Apache server will be displaying what looks to be an exact copy of website that was chosen earlier by p_m.

Now that SET tool is listening, p_m has prepared the ground for the successful implementation of the attack. To do this, p_m needs to select a target node by choosing one of the following two options:

1) Scans the entire network and selects one or more specific nodes out of those in the set $\mathcal{U}$;
2) All members in $\mathcal{U}$ are considered as targets;

Despite the option that will be selected by p_m the procedure is the same[1]. More precisely, p_m needs to make sure that the victim(s) on the local area network will be redirected from the intended website to the localhost site that p_m cloned earlier.

Originally we attempted this man-in-the-middle attack using a tool called Ettercap and only against websites that rely on the Hypertext Transfer Protocol (HTTP). However, this scenario is not considered as realistic since most of the websites that require users to login by providing their credentials are using the secure version of HTTP using the Secure Layer Socket (SSL) [19] protocol to encrypt the communication channel between the server and the client. As a result, we extended our use case by targeting only websites that were operating under the HTTPS protocol. However, the use of Ettercap arose various issues as the corresponding website would block the redirect and would display the page as unavailable. As a result, our attack could not be launched and we had to look for alternative ways to overcome that limitation.

A tool to help bypass this is called SSLStrip [20] and forces the site to communicate in plain text over HTTP – therefore stripping the secure layer. However, once this tool became popular, a new security measure was put in place called HTTP Strict Transport Security (HSTS) [21]. HSTS is a HTTP header that forces web browsers to communicate only through HTTPS. As a result, HSTS prevents attacks like man-in-the-middle or downgrade attacks using SSLStrip. However, sites that are SSL-enabled do not necessarily use HSTS. So, the next step was to check whether the underlying website is HSTS-aware. To do so, a command was used to see what the sites server response is [22]. When checking a secure site like Facebook using this command curl − s − D − https : //www.facebook.com/|grep − iStrict the server will respond with a response if HSTS enabled as seen in figure 3. If there is no message from the server, it shows that HSTS is not in the header.

As we wanted our tool to work in both cases (i.e. websites that do or do not support HSTS) we had to find a solution that

[1]The only difference is that in the first case our tool will attack only the nodes that were selected by p_m as victims while in the second case any $u_i\mathcal{U}$ that tries to access the website that was specified by p_m will be attacked.

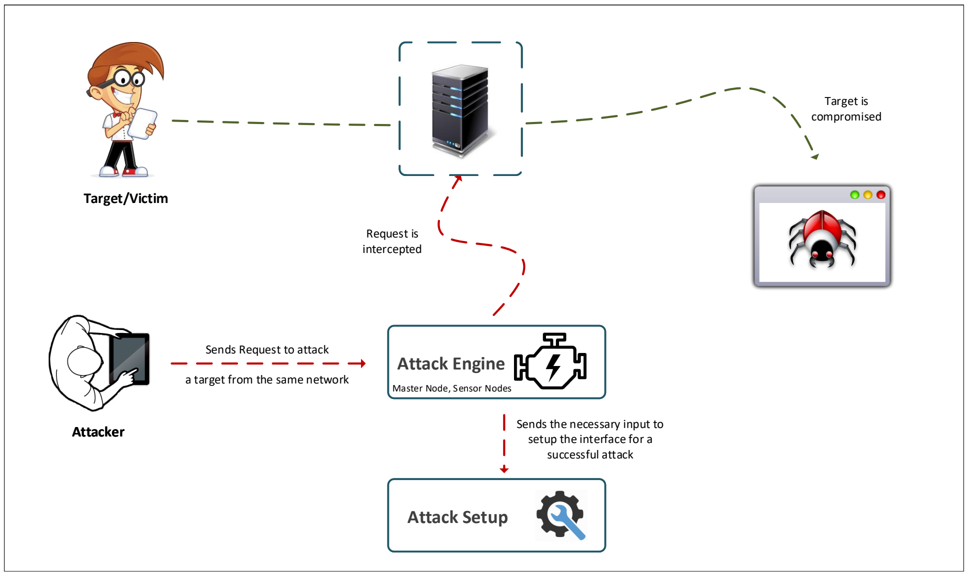

Target/Victim
Target is compromised
Request is intercepted
Sends Request to attack
a target from the same network
Attacker
Attack Engine
Master Node, Sensor Nodes
Sends the necessary input to setup the interface for a successful attack
Attack Setup

Fig. 1. System Model

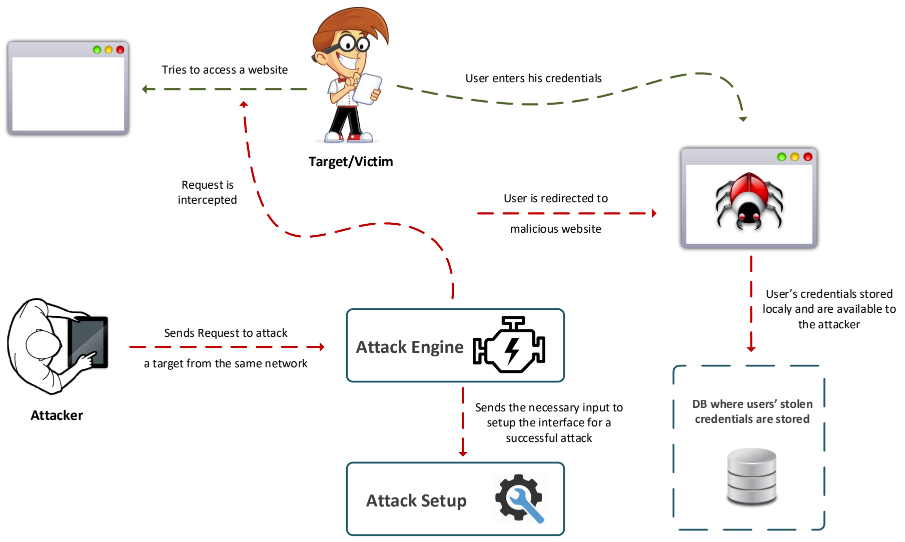
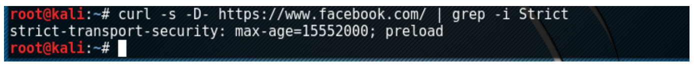
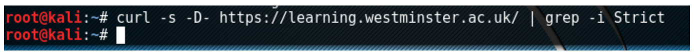

Tries to access a website
User enters his credentials
Target/Victim
Request is intercepted
User is redirected to
malicious website
User's credentials stored localy and are available to the attacker
Sends Request to attack
a target from the same network
Attacker
Attack Engine
Sends the necessary input to setup the interface for a successful attack
Attack Setup
DB where users' stolen credentials are stored

Fig. 2. Man-in-the-Middle Attack



```
root@kali:~# curl -s -D- https://www.facebook.com/ | grep -i Strict
strict-transport-security: max-age=15552000; preload
root@kali:~# █
```

Fig. 3. Check HSTS



```
root@kali:~# curl -s -D- https://learning.westminster.ac.uk/ | grep -i Strict
root@kali:~# █
```

Fig. 4. Check Websites HSTS

would allow us to bypass the HSTS header. We achieved that by using Bettercap [23]) – a man-in-the-middle framework that uses a built in SSLStrip function. Even though the use of Bettercap allowed us to bypass the security provided by the HSTS our attack was still not applicable to certain websites (e.g. Facebook, Gmail) as these sites have no HTTP versions available for the SSLStrip to strip the HTTPS to HTTP.

B. Step-by-Step Configuration

In this section we present a detailed guide with all the steps taken in order to successfully configure the attack described earlier.

a) Phase 1 – SET and Bettercap Installation: First was the installation of The Social-Engineer tool kit (SET). SET is currently only supported for platforms Linux and MacOS X. To install the latest version we need to clone the repository from git[2] in the command line. SET should install with all its dependencies during installation. Then next step within

[2]Git is a version control system used by SET.

the command line is the installation process of Bettercap. This tool is a command line interface Man-in-the-Middle framework built to be portable and easily extensible. This tool can perform various types of Man-in-the-Middle attacks. Once Bettercap and SET are installed our graphical user interface has the necessary dependencies for its functionality.

b) Phase 2 – Local IP: In this step we gathered p_m local IP address once the graphical user interface was started. This is so SET could initialise the server and prepare for listening. To do this we used socket programming within Python. Firstly we opened a INET socket, then used it to send a request to a website via the port specified. Once the socket sent the request we gathered the socket name and stored this into a variable called localIP. The response which now should be stored in the variable localIP will have the p_m local IP address.

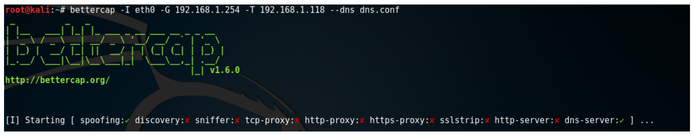
c) Phase 3 – Clone site: In this step we need to clone a website on p_m. After user enters the url for the website to be cloned it is stored in a string variable named website. The website is then passed into a function called clonedSite. This function runs SET as a sub process and automatically enters the correct options, the local IP address that was collected in Phase 2 and the website variable. Once these steps have been successfully completed, SET would have created a cloned website and will be listening for the credentials on the local host.

d) Phase 4 – Create malicious DNS file: In this step we need to create the malicious DNS file that is responsible for the redirection. The user can decide what variations of the url to use to redirect the target device u_t to the local host p_m. The text given will be stored in a string variable called dnsFile then passed into a function called createDns which will write into a file called dns.conf.

e) Phase 5 – Network interface and Gateway address: In this step we need to find what network interface on p_m is connected to the internet. This is done by running a system command that will return the network interface which is stored in a variable named netInterface. The same process will be done to gather the p_m gateway address which is stored in a variable named gatewayIP.

f) Phase 6 – Initialise Bettercap*:* In this step we need to initialise Bettercap by passing in various parameters. The user can decide to attack one specific target u_t or the entire network u_i. Once the selection is done the graphical user interface will create a sub process and run Bettercap with the necessary parameters that have been gathered from Phases 4 and 5 (network interface, Gateway IP address and dns.conf file). Once the attack is live the user will be notified that attack is active. This will enable a button to stop the attack.

g) Phase 7 – Display the credentials: In this final step we need to display any credentials that have been harvested during the attack. Once the attack is in process and new credentials are gathered, the graphical user interface will display the credentials of the u_t. The credentials will be stored in a file called harvester.txt and a function will be called to read from the text file. This is then displayed into a label on the graphical user interface.

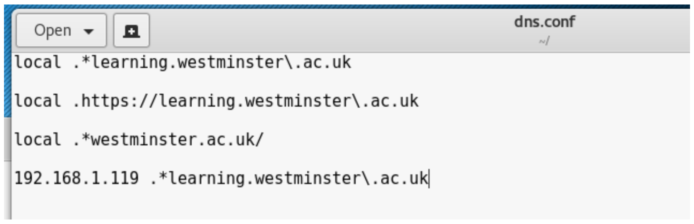


```
root@kali:~# bettercap -I eth0 -G 192.168.1.254 -T 192.168.1.118 --dns dns.conf

                                  |_| v1.6.0
http://bettercap.org/

[I] Starting [ spoofing:✓ discovery:✗ sniffer:✗ tcp-proxy:✗ http-proxy:✗ https-proxy:✗ sslstrip:✗ http-server:✗ dns-server:✓ ] ...
```

Fig. 5. Bettercap arguments passed

-I is for the interface, -G is the gateway IP address on the local network, -T is the targets IP address, the -dns argument is using the dns.conf file to spoof.

Figure 6 shows the dns.conf file we created which is redirecting learning.westminster.ac.uk to the local machine p_m (i.e. the attackers computer).



```
local .*learning.westminster\.ac.uk

local .https://learning.westminster\.ac.uk

local .*westminster.ac.uk/

192.168.1.119 .*learning.westminster\.ac.uk
```

Fig. 6. DNS configuration file

What this attack does is tell the target u_t that the attackers p_m machine is the router, then the p_m forwards it on to the router and once the p_m receives a response it is then forwarded back the u_t hence being the man in the middle. After this the DNS spoofing will be redirecting attempts at the cloned site back to the localhost (p_m cloned site) instead of the official site. If the u_t does not realise that she is on a cloned site and enters her credentials, these credentials will be sent back to the attacker's machine and then stored in a file and displayed.

C. Denial-of-Service Attack

The main purpose of a Denial-of-Service attack [24], [25], [26], [27], is to disable target(s) u_t access to the network. This is a network based attack. The attack can be only implemented on users who are connected on a local area network. We achieve the attack by manipulating packets on the network using Scapy [28] within Python and flooding the target. Scapy is a packet manipulation program that allows us to send forge packets over the network. When flooding the network with large amounts of packets this will leave the victim u_t unable to handle all the requests. Figure 7 shows an overview of the attack.

During the initialisation phase, the master node p_m needs to select a target node by choosing one of the following two options:

1) Scans the entire network and selects one or more specific nodes out of those in the set $\mathcal{U}$;
2) Target u_t is already known and chosen;

Despite the option selected by p_m the procedure is the same[3]. More precisely, p_m needs to make sure that the

[3]The only difference is that in the first case our tool will scan the network for the nodes by p_m. while in the second case any $u_i \mathcal{U}$ known already will be attacked by p_m.

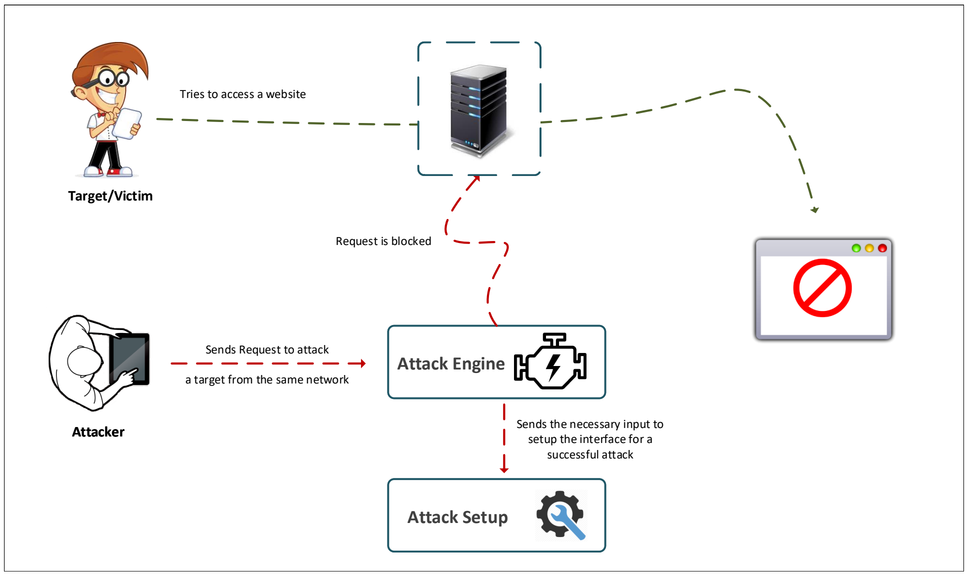

Tries to access a website
Target/Victim
Request is blocked
Sends Request to attack
a target from the same network
Attacker
Attack Engine
Sends the necessary input to setup the interface for a successful attack
Attack Setup

Fig. 7. Denial-of-Service Attack

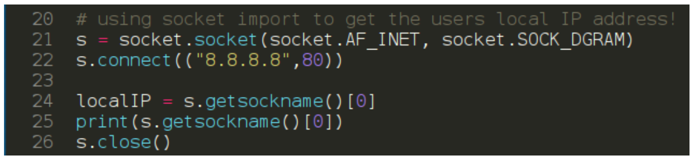
victim(s) on the local area network will be over flooded with packets from p_m.

V. Implementation

In this chapter, we will be discussing the implementation. To this end, we will be covering subjects such as programming language and techniques used, code extracts and the description of some of the challenges encountered while programming progressed.

a) Python:: The programming language used to create the attack tool was Python. The reason being is Python is a suitable scripting language for penetration testing because Python is a cross-platform object-oriented language that has a huge framework of libraries. For the graphical user interface a toolkit called PyQt [29] was used to facilitate communication between the user and the system.

b) Socket Programming: : For the tool to function it is necessary to gather the local IP address from the master node p_m. This was done by socket programming within Python. Firstly we had to create an open socket and connect to the internet. Once a connection was established we then stored the socket into a variable. This was then used for the local IP address as seen in figure 8.

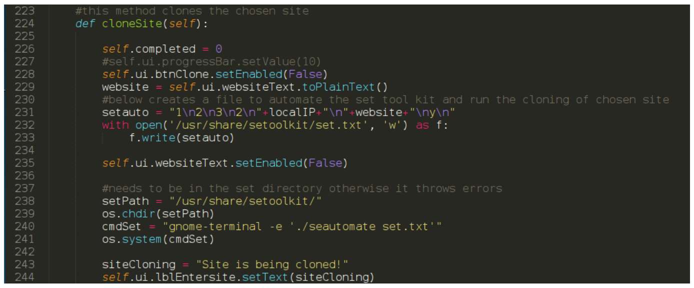


```
# using socket import to get the users local IP address!
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8",80))

localIP = s.getsockname()[0]
print(s.getsockname()[0])
s.close()
```

Fig. 8. Socket Programming

c) Cloning site: : On one hand we needed the tool to clone a website only having the p_m inputting the url address. On the other, SET has a series of steps that need to be followed in order for a site to be cloned. Automating those steps was the challenging part because the method used initially failed. The initial attempt to solve this challenge was to use a Python library called Pexpect [30] and code keystrokes into SET manually. Unfortunately this did not work. Through further analysis it was discovered that SET contains an executable process using Pexepect that allows you to create a file containing the desired inputs for a given task.

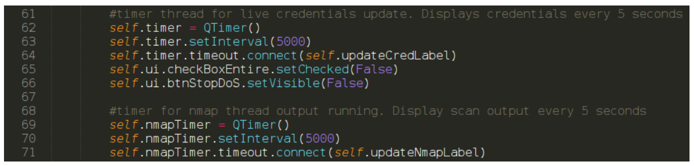
Figure 9 shows the code created to solve this problem. Line 231 details the hardcoded options, the localIP obtained and the website to be cloned. Lines 232 and 233 include the process in which the information is saved into a file called set.txt. In line 240 the file is used as a parameter for SET automating executable process.



```
223     #this method clones the chosen site
224     def cloneSite(self):
225
226         self.completed = 0
227         #self.ui.progressBar.setValue(10)
228         self.ui.btnClone.setEnabled(False)
229         website = self.ui.websiteText.toPlainText()
230         #below creates a file to automate the set tool kit and run the cloning of chosen site
231         setauto = "1\n2\n3\n2\n"+localIP+"\n"+website+"\ny\n"
232         with open('/usr/share/setoolkit/set.txt', 'w') as f:
233             f.write(setauto)
234
235         self.ui.websiteText.setEnabled(False)
236
237         #needs to be in the set directory otherwise it throws errors
238         setPath = "/usr/share/setoolkit/"
239         os.chdir(setPath)
240         cmdSet = "gnome-terminal -e './seautomate set.txt'"
241         os.system(cmdSet)
242
243         siteCloning = "Site is being cloned!"
244         self.ui.lblEntersite.setText(siteCloning)
```

Fig. 9. Cloning site with SET automation

d) Threads: : One of the most challenging parts of coding was the need to run multiple threads simultaneously to show live updates of credentials or whilst scanning the network for targets u_t. The graphical user interface is running the main thread for p_m to interact with it. Two extra threads (timer and nmapTimer) were created to run along side the main one.



```
#timer thread for live credentials update. Displays credentials every 5 seconds
self.timer = QTimer()
self.timer.setInterval(5000)
self.timer.timeout.connect(self.updateCredLabel)
self.ui.checkBoxEntire.setChecked(False)
self.ui.btnStopDoS.setVisible(False)

#timer for nmap thread output running. Display scan output every 5 seconds
self.nmapTimer = QTimer()
self.nmapTimer.setInterval(5000)
self.nmapTimer.timeout.connect(self.updateNmapLabel)
```

Fig. 10. Threads implementation

The timer thread was needed for displaying the credentials as each target u_t posts them. The thread would need to start once the attack is active and display onto the screen at a five second interval.

The second one, nmapTimer, was used for the live scanning of the network. When the network is scanned a thread needs to run to show all connected devices as they are found. Once the thread starts it will populate the interface with details of the connected devices.

Figure 10 shows the creation of both thread objects, their time interval whilst running and the connection to their methods.

VI. Conclusion

In this paper, we proposed a protocol for secure and efficient sharing of ehealth data in a multi-cloud environment. The proposed protocol was based on a Revocable Key-Policy Attribute-Encryption and allowed users to control access rights directly on encrypted data. Moreover, the proposed protocol allows secure and efficient revocation of access to data, for a certain user that is compromised, misbehaving or is no longer part of a user group, without having to decrypt and re-encrypt the original data.

As future steps, we plan to implement our protocol in order to measure its performance and prove its effectiveness in a real cloud environment. Furthermore, we plan to explore the incorporation of our protocol with mobile sensing applications and with privacy-preserving reputation systems for cloud-based participatory sensing applications [31]. The envisioned system will be based on [32], [33], [34] and will effectively maintain the privacy and anonymity of users [35].

References

[1] Palmer, D, “History repeating: How the IoT is failing to learn the security lessons of the past.” http://www.zdnet.com/article/history-repeating-how-the-internet-of-things-failed-to-learn-the-security-lessons-of-the-past/, 2016. Online; accessed 20-April-2017.

[2] A. Michalas and R. Dowsley, “Towards trusted ehealth services in the cloud,” in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pp. 618–623, Dec 2015.

[3] K. Yigzaw, A. Michalas, and J. Bellika, “Secure and scalable statistical computation of questionnaire data in r,” *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2016.

[4] K. Y. Yigzaw, A. Michalas, and J. G. Bellika, “Secure and scalable deduplication of horizontally partitioned health data for privacy-preserving distributed statistical computation,” *BMC Medical Informatics and Decision Making*, vol. 17, no. 1, p. 1, 2017.

[5] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hbsch, and I. Paraskakis, “Paasword: A holistic data privacy and security by design framework for cloud services,” in *Proceedings of the 5th International Conference on Cloud Computing and Services Science*, pp. 206–213, 2015.

[6] Y. Verginadis, A. Michalas, P. Gouvas, G. Schiefer, G. Hübsch, and I. Paraskakis, “Paasword: A holistic data privacy and security by design framework for cloud services,” pp. 1–16, 2017.

[7] N. Paladi and A. Michalas, ““One of our hosts in another country”: Challenges of data geolocation in cloud storage,” in *Wireless Communications, Vehicular Technology, Information Theory and Aerospace Electronic Systems (VITAE), 2014 4th International Conference on*, pp. 1–6, May 2014.

[8] A. Michalas, N. Paladi, and C. Gehrmann, “Security aspects of e-health systems migration to the cloud,” in *e-Health Networking, Applications and Services (Healthcom), 2014 IEEE 16th International Conference on*, pp. 212–218, IEEE, 2014.

[9] A. Michalas and M. Bakopoulos, “Secgod google docs: Now i feel safer!,” in *2012 International Conference for Internet Technology And Secured Transactions*, pp. 589–595, Dec 2012.

[10] R. Dowsley, A. Michalas, and M. Nagel, “A report on design and implementation of protected searchable data in iaas,” tech. rep., Swedish Institute of Computer Science (SICS), 2016.

[11] A. Michalas, “Sharing in the rain: Secure and efficient data sharing for the cloud,” in *2016 International Conference for Internet Technology And Secured Transactions*, pp. 589–595, Dec 2016.

[12] A. Michalas and K. Y. Yigzaw, “Locless: Do you really care your cloud files are?,” in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pp. 618–623, Dec 2015.

[13] N. Paladi, C. Gehrmann, and A. Michalas, “Providing user security guarantees in public infrastructure clouds,” *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[14] N. Paladi, A. Michalas, and C. Gehrmann, “Domain based storage protection with secure access control for the cloud,” in *Proceedings of the 2014 International Workshop on Security in Cloud Computing*, ASIACCS ’14, (New York, NY, USA), ACM, 2014.

[15] Alberto Ornaghi, “Ettercap.” https://ettercap.github.io/ettercap/, 2004. Online; accessed 3 April 2017.

[16] Subterfuge, “Subterfuge-Framework.” https://github.com/Subterfuge-Framework/Subterfuge, 2014. Online; accessed 1 April 2017.

[17] T. Wilhelm, *Professional Penetration Testing, Second Edition: Creating and Learning in a Hacking Lab*. Syngress Publishing, 2nd ed., 2013.

[18] Trusted Sec, “The Social-Engineer Toolkit (SET).” https://www.trustedsec.com/social-engineer-toolkit/, 2016. Online; accessed 1 December 2016.

[19] Admin EV, “What is SSL?.” http://info.ssl.com/article.aspx?id=10241, 2016. Online; accessed 05 February 2017.

[20] Moxie Marlinspike, “Software ¿¿ sslstrip.” https://moxie.org/software/sslstrip/, 2011. Online; accessed 7 February 2017.

[21] Christian, “HTTP Strict Transport Security .” https://tutorialinux.com/strict-transport-security/, 2016. Online; accessed 9 February 2017.

[22] nameCheap, “How to check if HSTS is enabled.” https://www.namecheap.com/support/knowledgebase/article.aspx/9711//how-to-check-if-hsts-is-enabled, 2016. Online; accessed 9 February 2017.

[23] EvilSocket, “Bettercap is the state of the art, modular, portable and easily extensible MITM framework..” https://www.bettercap.org/, 2017. Online; accessed 9 February 2017.

[24] A. Michalas, N. Komninos, N. R. Prasad, and V. A. Oleshchuk, “New client puzzle approach for dos resistance in ad hoc networks,” in *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference*, pp. 568–573, IEEE, 2010.

[25] A. Michalas, N. Komninos, and N. R. Prasad, “Mitigate dos and ddos attack in mobile ad hoc networks,” *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 3, no. 1, pp. 14–36, 2011.

[26] A. Michalas, N. Komninos, and N. Prasad, “Multiplayer game for ddos attacks resilience in ad hoc networks,” in *Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on*, pp. 1–5, Feb 2011.

[27] A. Michalas, N. Komninos, and N. R. Prasad, “Cryptographic puzzles and game theory against dos and ddos attacks in networks,” *International Journal of Computer Research*, vol. 19, no. 1, p. 79, 2012.

[28] Philippe Biondi, “Scapy.” http://www.secdev.org/projects/scapy/doc/introduction.html, 2007. Online; accessed 13 April 2017.

[29] Riverbank, “PyQt.” https://riverbankcomputing.com/software/pyqt/intro, 2016. Online; accessed 17 April 2017.

[30] Noah Spurrier, “Pexpect.” https://pexpect.readthedocs.io/en/stable/, 2013. Online; accessed 19 April 2017.

[31] A. Michalas and N. Komninos, “The lord of the sense: A privacy preserving reputation system for participatory sensing applications,” in *Computers and Communication (ISCC), 2014 IEEE Symposium*, pp. 1–6, IEEE, 2014.

[32] T. Dimitriou and A. Michalas, “Multi-party trust computation in decentralized environments,” in *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5, May 2012.

[33] T. Dimitriou and A. Michalas, “Multi-party trust computation in decentralized environments in the presence of malicious adversaries,” *Ad Hoc Networks*, vol. 15, pp. 53–66, Apr. 2014.

[34] A. Michalas, V. A. Oleshchuk, N. Komninos, and N. R. Prasad, “Privacy-preserving scheme for mobile ad hoc networks,” in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pp. 752–757, June 2011.

[35] A. Michalas, M. Bakopoulos, N. Komninos, and N. R. Prasad, “Secure amp; trusted communication in emergency situations,” in *Sarnoff Symposium (SARNOFF), 2012 35th IEEE*, pp. 1–5, May 2012.

View publication stats