Thursday, January 12, 2017

Networking

Port forwarding: A practical hands-on guide

1. Introduction

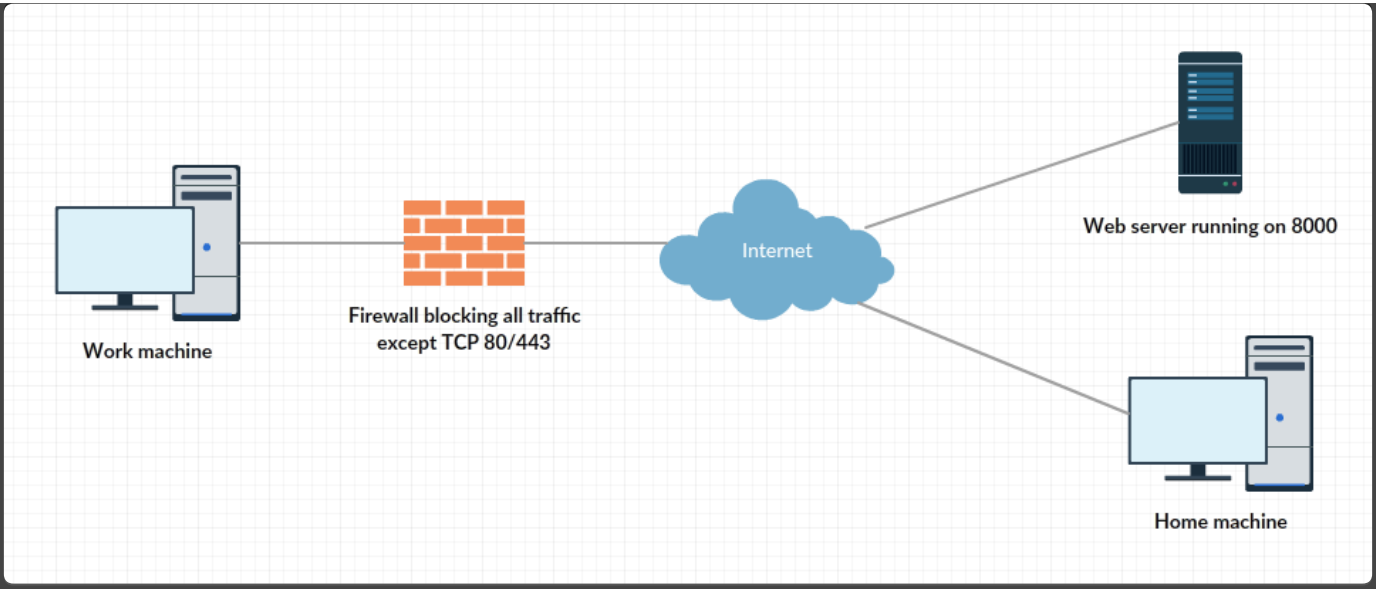
During my preparation for the PWK/OSCP course (starting late January), I had a hard time understanding how port forwarding/tunneling works. Most guides I found were too theoretical for my taste, this guide will allow you to mimic a strict corporate firewall and how to bypass it.

So what is port forwarding? It's a technique that allows you to redirect traffic from one port to another port/IP. What the hell does that mean?

Imagine the following scenario:

Your corp firewall only allows outbound connections to web servers running on standard ports (port 80 for HTTP / port 443 for HTTPS). Your favorite security news website for some reason is run on port 8000. Given the strict corp firewall you won't be able to browse the site any more.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



You decide to use your knowledge of port forwarding to still be able to browse your favorite site. We'll be using rinetd as TCP forwarder.

---

2. Requirements

To follow this guide you'll need the following:

- Host machine capable of running 2 Linux VMs.
- LAN network.
- Some patience.

---

3. Assumptions

All machines are on the same network for simplicity, thus no router configuration is needed.

Machine	IP
abatchy-work	192.168.20.130

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

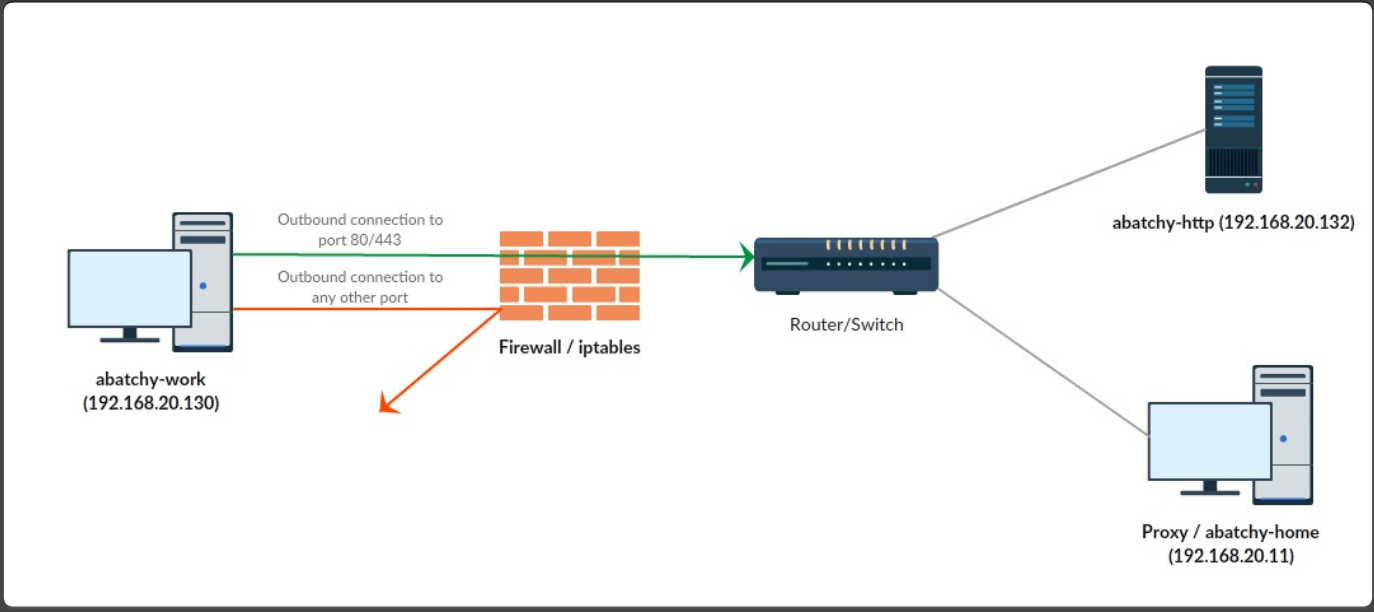
abatchy-proxy	192.168.20.131
abatchy-http	192.168.20.132

- **192.168.20.130**: Work machine, ideally it's behind a NAT and firewalled. For simplicity it's in the same LAN and iptables is used instead.
- **192.168.20.131**: Home machine, used as proxy to forward inbound traffic on port 80 to 192.168.20.132 on port 8000.
- **192.168.20.132:** Web server running your favorite site on non-standard port 8000, inaccessible directly from your work machine because of firewall.

---

4. Setup Before Port Forwarding

We'll be configuring the following setup:



4.1 Setting up Apache server

We'll start by setting up our Apache server on 192.168.20.132 (Web server)

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
// Install Apache server
abatchy@abatchy-http:~$ sudo apt-get install apache2

// Change default port Apache is listening on to 8000
abatchy@abatchy-http:~$ sudo nano /etc/apache2/ports.conf
// Change virtual host from
// <VirtualHost *:80>
// to
// <VirtualHost *:8000>
abatchy@abatchy-http:~$ sudo nano /etc/apache2/sites-enabled/000-default

// Restart Apache server
abatchy@abatchy-http:~$ sudo service restart apache2
```

Now let's verify it's working as expected.

```
//sudo is needed since netstat won't show processes not owned by abatchy
abatchy@abatchy-http:~$ sudo netstat -antp | grep apache2

tcp        0      0 0.0.0.0:8000            0.0.0.0:*               LISTEN      4523/a
```

4.2 Configuring iptables

Next, we'll configure iptables on `192.168.20.130` (work machine) to drop any outgoing traffic except for ones using TCP port 80 and 443.

```
// Verify that there are no rules defined yet

abatchy@abatchy-work:~$ sudo iptables -L
[sudo] password for abatchy:

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination


// Block all traffic

abatchy@abatchy-work:~$ sudo iptables -I OUTPUT -j DROP

// Allow outbound traffic on port 80

abatchy@abatchy-work:~$ sudo iptables -I OUTPUT -p tcp --dport 80 -j ACCEPT

// Allow outbound traffic on port 443

abatchy@abatchy-work:~$ sudo iptables -I OUTPUT -p tcp --dport 443 -j ACCEPT

// View current rules defined

abatchy@abatchy-work:~$ sudo iptables -L

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
**ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:https
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http**
DROP       all  --  anywhere             anywhere
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Notice the order the rules were applied. `-I` was used to put the rules on top of the chain, thus overriding the "block all traffic" rule. One more thing is that you can't make DNS requests / resolve URLs since UDP port 53 is included in the rule.

Next, let's ensure that the current rules work properly.

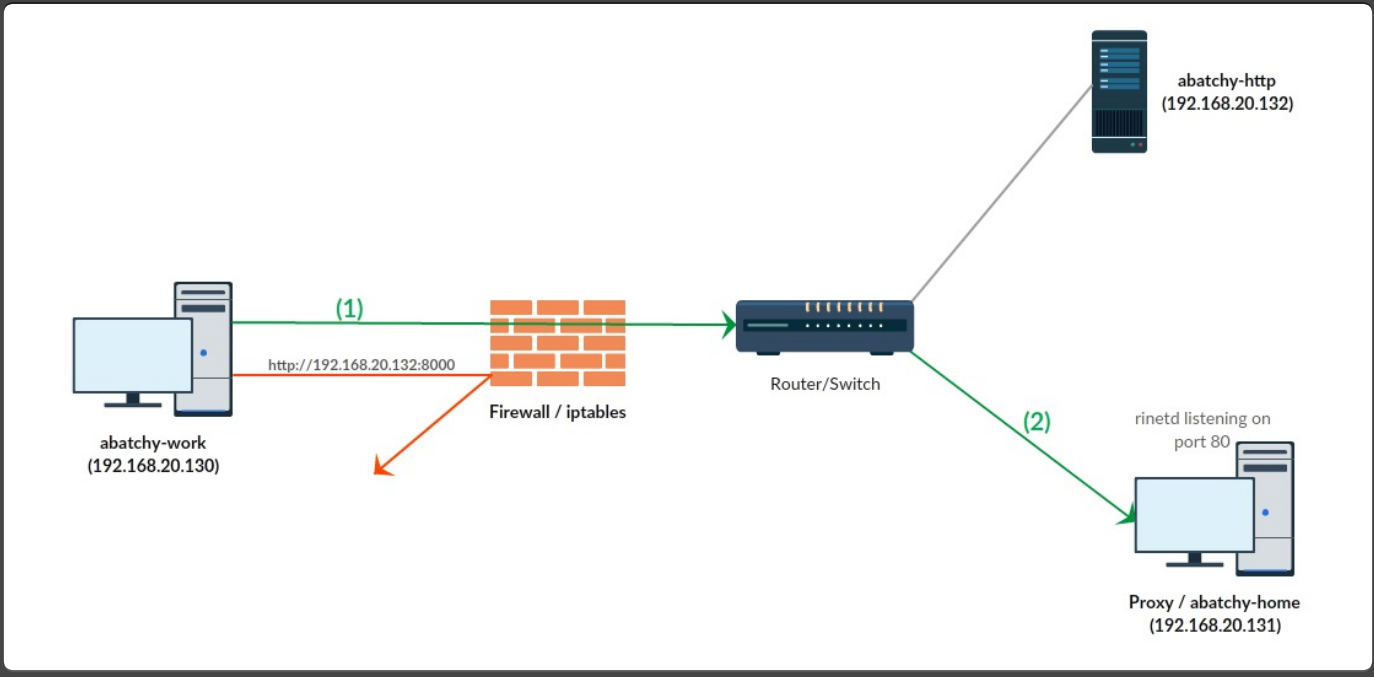
```
abatchy@abatchy-work:~$ wget http://192.168.20.132:8000
--2017-01-12 03:01:24--  http://192.168.20.132:8000/
Connecting to 192.168.20.132:8000... ^C
```

As expected, we're not able to connect to `http://192.168.20.132:8000`. In case you want to see logs of the dropped packets check this.

---

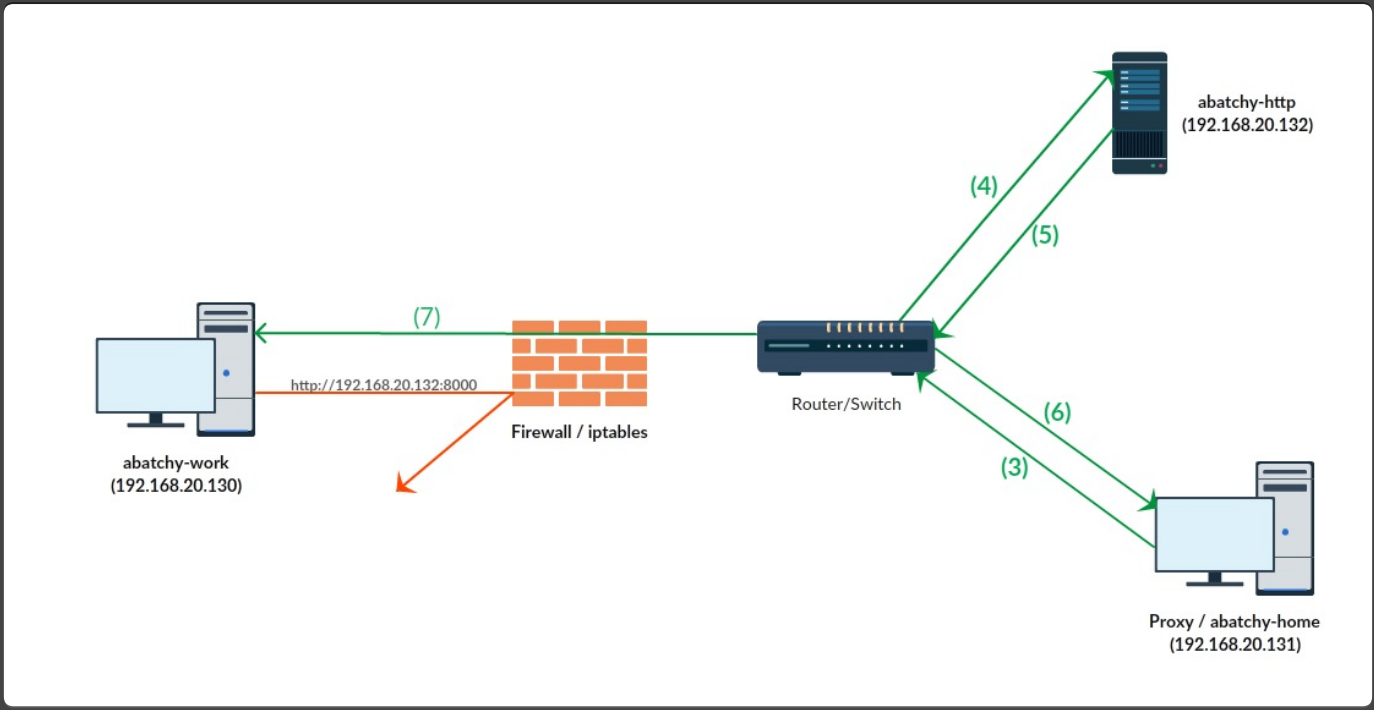
5. Setup After Port Forwarding

5.1 Data flow



Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

(1) Outgoing traffic to `192.168.20.131:80` which our firewall will let through to our switch.
(2) On our proxy machine, we will have `rinetd` listening on port 80.



(3) and (4) Traffic is forwarded by `rinetd` to our webserver, listening on `192.168.20.132:8000`.
(5) and (6) Web server replies back to our proxy machine.
(7) Proxy replies back to our work machine, traffic still flowing as it's on port 80.

5.2 Rinetd on proxy machine

Rinetd is a very light-weight, simple-to-use TCP forwarder, we will set it up so it redirects incoming traffic on port 80 to `192.168.20.132:8000` (our web server).

```
// Install rinetd
abatchy@abatchy-proxy:/home$ sudo apt-get install rinetd

// Add the following rule below the comment
abatchy@abatchy-proxy:/home$ sudo nano /etc/rinetd.conf
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
# bindadress    bindport  connectaddress  connectport
192.168.20.131  80      192.168.20.132  8000

// Restart the service
abatchy@abatchy-proxy:/home$ sudo service rinetd restart
```

---

6. Did it work?

```
abatchy@abatchy-work:~$ wget http://192.168.20.131:80
--2017-01-12 19:56:35--  http://192.168.20.131/
Connecting to 192.168.20.131:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 20 [text/html]
Saving to: 'index.html'

100%[=====================================>] 20          --.-K/s   in 0s

2017-01-12 19:56:35 (6.33 MB/s) - 'index.html' saved [20/20]

abatchy@abatchy-work:~$ cat index.html
Super cool website!
```

It worked! A wireshark capture below on the proxy shows the traffic flow.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

No.	Source	Destination	Protocol	Length	Info
2266	192.168.20.130	192.168.20.131	TCP	76	57319→80 [SYN] Seq=0 Win=
2267	192.168.20.131	192.168.20.130	TCP	76	80→57319 [SYN, ACK] Seq=0

tcp.port == 80 || tcp.port == 8000

No.	Source	Destination	Protocol	Length	Info
2266	192.168.20.130	192.168.20.131	TCP	76	57319→80 [SYN] Seq=0 Win=
2267	192.168.20.131	192.168.20.130	TCP	76	80→57319 [SYN, ACK] Seq=0
2268	192.168.20.130	192.168.20.131	TCP	68	57319→80 [ACK] Seq=1 Ack=
2269	192.168.20.130	192.168.20.131	HTTP	180	GET / HTTP/1.1
2270	192.168.20.131	192.168.20.130	TCP	68	80→57319 [ACK] Seq=1 Ack=
2271	192.168.20.131	192.168.20.132	TCP	76	39258→8000 [SYN] Seq=0 Wi
2272	192.168.20.132	192.168.20.131	TCP	76	8000→39258 [SYN, ACK] Seq
2273	192.168.20.131	192.168.20.132	TCP	68	39258→8000 [ACK] Seq=1 Ac
2274	192.168.20.131	192.168.20.132	HTTP	180	GET / HTTP/1.1
2275	192.168.20.132	192.168.20.131	TCP	68	8000→39258 [ACK] Seq=1 Ac
2276	192.168.20.132	192.168.20.131	HTTP	370	HTTP/1.1 200 OK (text/ht
2277	192.168.20.131	192.168.20.132	TCP	68	39258→8000 [ACK] Seq=113
2278	192.168.20.131	192.168.20.130	HTTP	370	HTTP/1.1 200 OK (text/ht
2279	192.168.20.130	192.168.20.131	TCP	68	57319→80 [ACK] Seq=113 Ac
2280	192.168.20.130	192.168.20.131	TCP	68	57319→80 [FIN, ACK] Seq=1
2281	192.168.20.131	192.168.20.130	TCP	68	80→57319 [FIN, ACK] Seq=3
2282	192.168.20.131	192.168.20.132	TCP	68	39258→8000 [FIN, ACK] Seq
2283	192.168.20.130	192.168.20.131	TCP	68	57319→80 [ACK] Seq=114 Ac
2284	192.168.20.132	192.168.20.131	TCP	68	8000→39258 [FIN, ACK] Seq
2285	192.168.20.131	192.168.20.132	TCP	68	39258→8000 [ACK] Seq=114

2266-2268: Three way handshake between Work and Proxy:80.
2269-2270: GET request and ACK.
2271-2273: Three way handshake between Proxy and WebServer:8000.
2274-2275: GET request and ACK
2276-2277: WebServer responds and Proxy acknowledges.
2278-2279: Proxy forwards received packet back to Work.
2280-2285: Work terminates connection with Proxy and Proxy terminates connection with WebServer.

---

7. No love for Windows?

Rinetd supports Windows and is pretty straight-forward as on Linux as well. Binary can be found here.

Assuming rinetd is installed in: `C:\rinetd`, just add the same rule to `rinetd.conf`:

```
192.168.20.131  80      192.168.20.132  8000
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Then run rinetd:

```
C:\rinetd>rinetd.exe -c rinetd.conf
```

And done.

Hopefully this guide sheds some light on port forwarding and more advanced traffic manipulation.

Follow @abatchy17

Follow @abatchy17

Categories

- .Net Reversing
- Backdooring

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- DefCamp CTF Qualifications 2017
- Exploit Development
- Kernel Exploitation
- Kioptrix series
- Networking
- OSCE Prep
- OSCP Prep
- OverTheWire - Bandit
- OverTheWire - Leviathan
- OverTheWire - Natas
- Powershell
- Programming
- Pwnable.kr
- SLAE
- Shellcoding
- Vulnhub Walkthrough
- rant

Blog Archive

January 2018

- [Kernel Exploitation] 7: Arbitrary Overwrite (Win7 x86)
- [Kernel Exploitation] 6: NULL pointer dereference
- [Kernel Exploitation] 5: Integer Overflow
- [Kernel Exploitation] 4: Stack Buffer Overflow (SMEP Bypass)
- [Kernel Exploitation] 3: Stack Buffer Overflow (Windows 7 x86/x64)
- [Kernel Exploitation] 2: Payloads
- [Kernel Exploitation] 1: Setting up the environment

October 2017

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- [DefCamp CTF Qualification 2017] Don't net, kids! (Revexp 400)
- [DefCamp CTF Qualification 2017] Buggy Bot (Misc 400)

September 2017

- [Pwnable.kr] Toddler's Bottle: flag
- [Pwnable.kr] Toddler's Bottle: fd, collision, bof
- OverTheWire: Leviathan Walkthrough

August 2017

- [Rant] Is this blog dead?

June 2017

- Exploit Dev 101: Bypassing ASLR on Windows

May 2017

- Exploit Dev 101: Jumping to Shellcode
- Introduction to Manual Backdooring
- Linux/x86 - Disable ASLR Shellcode (71 bytes)
- Analyzing Metasploit linux/x86/shell_bind_tcp_random_port module using Libemu
- Analyzing Metasploit linux/x86/exec module using Ndisasm
- Linux/x86 - Code Polymorphism examples
- Analyzing Metasploit linux/x86/adduser module using GDB
- Analyzing Metasploit linux/x86/adduser module using GDB
- ROT-N Shellcode Encoder/Generator (Linux x86)
- Skape's Egg Hunter (null-free/Linux x86)
- TCP Bind Shell in Assembly (null-free/Linux x86)

April 2017

- Shellcode reduction tips (x86)

March 2017

- LTR Scene 1 Walthrough (Vulnhub)
- Moria v1.1: A Boot2Root VM
- OSCE Study Plan

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

- Powershell Download File One-Liners
- How to prepare for PWK/OSCP, a noob-friendly guide

February 2017

- OSCP-like Vulnhub VMs
- OSCP: Day 30
- Mr Robot Walkthrough (Vulnhub)

January 2017

- OSCP: Day 6
- OSCP: Day 1
- Port forwarding: A practical hands-on guide
- Kioptrix 2014 (#5) Walkthrough
- Wallaby's Nightmare Walkthrough (Vulnhub)

December 2016

- Kiopritx 1.3 (#4) Walkthrough (Vulnhub)
- Kioptrix 3 Walkthrough (Vulnhub)
- Kioptrix 2 Walkthrough (Vulnhub)
- OverTheWire: Natas 17

November 2016

- OverTheWire: Natas 16
- OverTheWire: Natas 14 and 15
- Kioptrix 1 Walkthrough (Vulnhub)
- PwnLab: init Walkthrough (Vulnhub)
- OverTheWire: Natas 12
- OverTheWire: Natas 11

October 2016

- Vulnix Walthrough (Vulnhub)
- OverTheWire: Natas 6-10
- OverTheWire: Natas 0-5

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Mohamed Shahat © 2018

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD