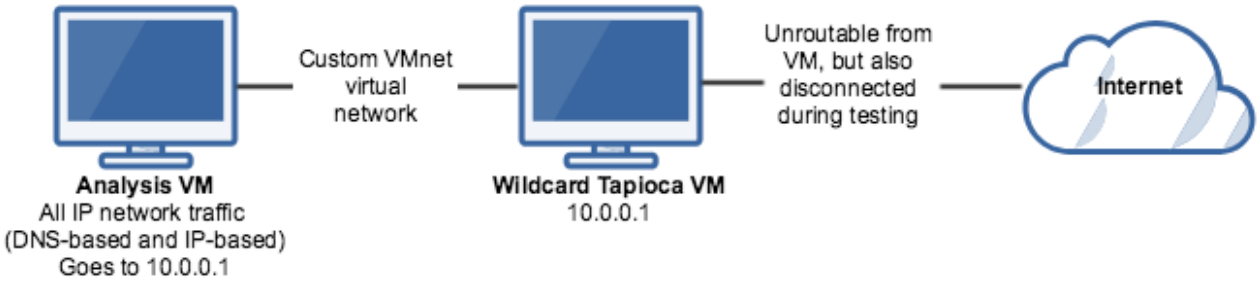
How to create a network wildcard VM using CERT Tapioca for exploit testing

- Overview
- Step-by-step guide
- Using the wildcard Tapioca server

Overview

Let's say you have an exploit, and you're not sure what it does. Many exploits do **something** on the network. It would be nice to be able to observe these network operations, without actually being connected to the internet. Running an unknown exploit on an internet-connected machine is a bad idea. As it turns out, we can simulate an internet-connected machine by turning our CERT Tapioca VM into something that responds to everything (both DNS-addressed, and IP-addressed).



Step-by-step guide

1. Start with a CERT Tapioca VM with dual ethernet adapters. I've found that Ubuntu 17.10 Server works well. For some reason, 18.04 Server has some issues related to VMware tools (copy/paste, HGFS) don't seem to be working well (yet?).
2. Disable systemd-resolved. It gets greedy with what it uses are DNS resolvers, and ends up picking up our wildcard DNS. This breaks DNS lookups on the wildcard VM itself, which we don't want.

   ```
   sudo systemctl disable systemd-resolved.service
   sudo service systemd-resolved stop
   ```

   Put the following line in the `[main]` section of your `/etc/NetworkManager/NetworkManager.conf`:

   ```
   dns=default
   ```

   Delete the symlink `/etc/resolv.conf`

   ```
   rm /etc/resolv.conf
   ```

   Restart network-manager

```
sudo service network-manager restart
```

3. Reconfigure your second (LAN side) NIC. When I made the changes above, it made an additional network adapter, leaving the already-configured one as a zombie. In my case, I clicked the network icon in the top right corner, deleted the old one, and reconfigured the new one.
4. Install tinydns

```
sudo apt install tinydns
```

5. Configure tinydns

```
sudo adduser --no-create-home --disabled-login --shell /bin/false
dnslog
sudo adduser --no-create-home --disabled-login --shell /bin/false
tinydns
sudo tinydns-conf tinydns dnslog /etc/tinydns/ 10.0.0.1
sudo mkdir -p /etc/service ; cd /etc/service ; sudo ln -sf /etc
/tinydns/
```

Edit /etc/tinydns/root/data to resolve everything to 10.0.0.1:

/etc/tinydns/root/data

```
.local:10.0.0.1:a:259200
.0.0.10.in-addr.arpa:10.0.0.1:a:259200
.:10.0.0.1
+*:10.0.0.1:86400
+*.local:10.0.0.1:86400
```

build the tinydns configuration in the `/etc/tinydns/root/` directory:

```
sudo make
```

6. Restart tinydns:

```
sudo svc -h /etc/service/tinydns
```

7. Confirm your dns lookups:

```
tapioca@ubuntu:~/tapioca$ nslookup asdf 10.0.0.1
Server:                10.0.0.1
Address:        10.0.0.1#53

Name:        asdf.localdomain
Address: 10.0.0.1
```

8. Edit the `~/tapioca/iptables_noproxy.sh` file, wiping out the `NAT Magic` part at the end, replacing it with:

```
# NAT magic
iptables -t nat -F PREROUTING
iptables -t nat -A PREROUTING -i $internal_net -j DNAT --to-
destination 10.0.0.1
```

   The iptables DNAT line will rewrite the target of traffic arriving on the LAN-side adapter to be handled by `10.0.0.1`

9. Edit the `~/tapioca/iptables_mitmproxy.sh` file, wiping out the NAT Magic part, replacing it with:

```
# mitmproxy interception
iptables -t nat -F PREROUTING
iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-
ports 8080
iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-
ports 8080
iptables -t nat -A PREROUTING -i $internal_net -j DNAT --to-
destination 10.0.0.1
```

10. Create a `~/tapioca/wildcard.sh` script to start mitmproxy HTTP(S) interception automatically on boot:

~/tapioca/wildcard.sh

```
#!/bin/bash

echo Setting network redirection rules...
cd /home/tapioca/tapioca
/home/tapioca/tapioca/proxy.sh
```

    Ensure that the `~/tapioca/wildcard.sh is executable`

```
chmod +x ~/tapioca/wildcard.sh
```

11. Configure the script to start automatically on X starting.

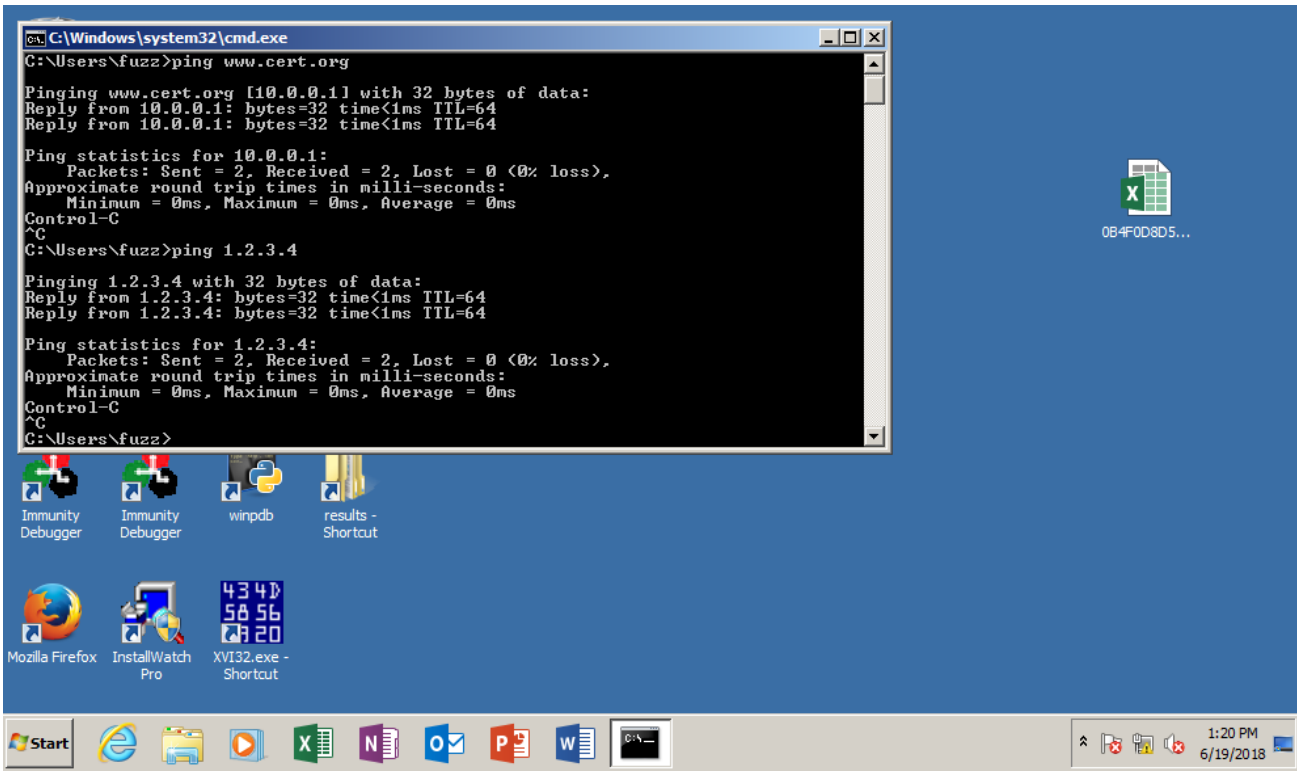
```
mkdir -p ~/.config/autostart
nano -w ~/.config/autostart/.desktop
```

Edit the `.desktop` file to look like this:

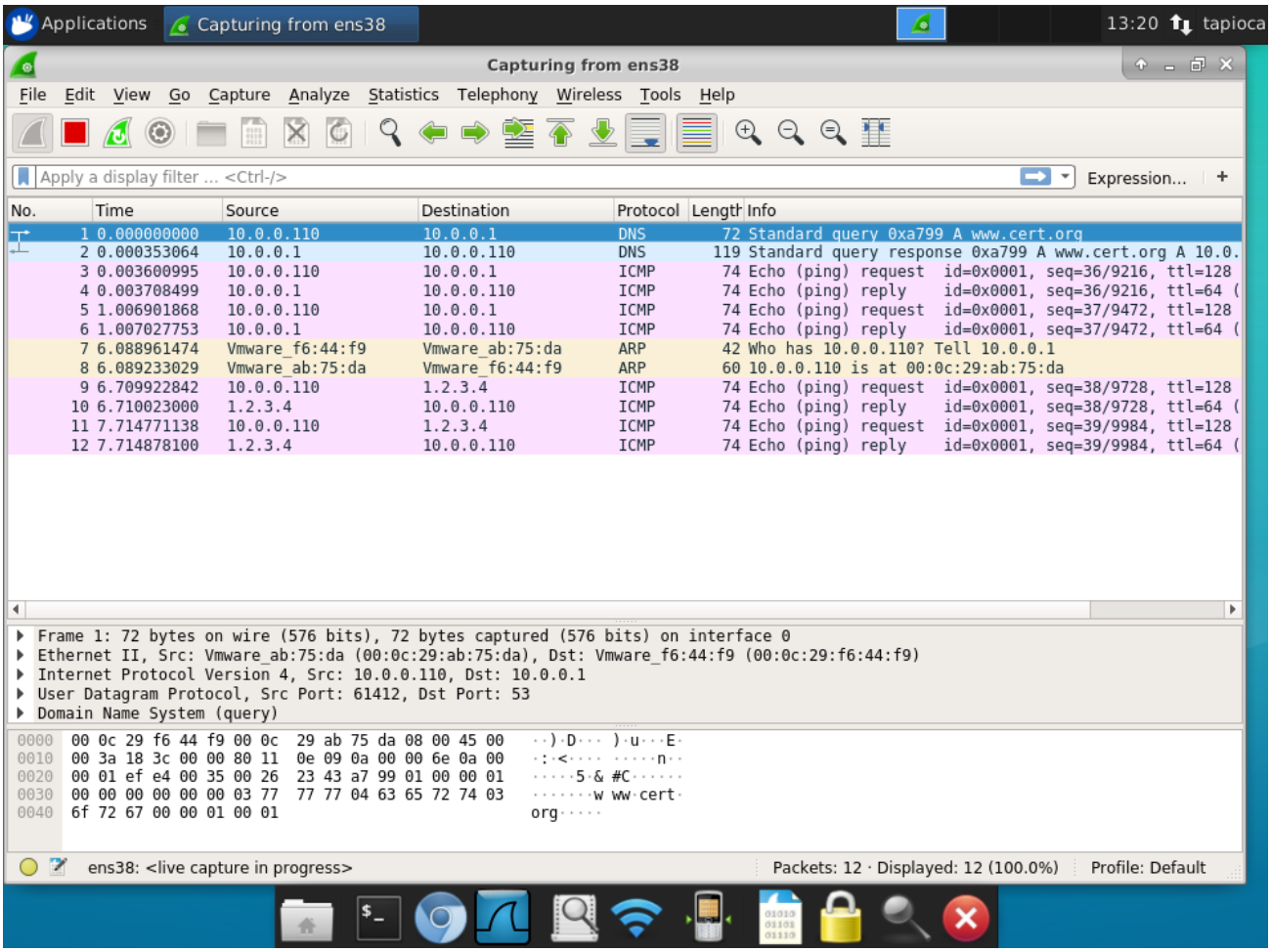
~/.config/autostart/.desktop

```
[Desktop Entry]
Encoding=UTF-8
Name=Wildcard
Comment=Wildcard network redirection
Exec=/home/tapioca/tapioca/wildcard.sh
Terminal=false
```

12. Click the Red 'X' in the wildcard VM to run the new wildcard rules (or just reboot).
13. Connect a testing VM to the same vmnet network as the LAN adapter of your new wildcard VM, and try some network stuff.



14. Confirm the traffic on the server side:



15. (optional) configure wildcard VM to automatically log in the tapioca user

```
sudo apt install mingetty
systemctl edit getty@tty1
```

Configure it to look like this:

```
[Service]
ExecStart=
ExecStart=-/sbin/mingetty --autologin tapioca --noclear %I
```

16. Install apache and php, as you'll likely at least want a web server to simulate

```
sudo apt-get install apache2
sudo apt-get install php libapache2-mod-php
sudo a2enmod ssl
sudo a2ensite default-ssl
sudo systemctl reload apache2
```

17. Create a shortcut for the apache2 log tail

1. Right click in the XFCE panel
2. Click **Panel Add new items**
3. Click **Launcher** and then **Add**
4. Right-click the new panel icon and click **Properties**
5. Click **`Add a new empty item`**
6. Call it what you want in the **`Name`** field. e.g. "Apache log tailer"
7. In the `Command` field, enter: `sudo tail -F /var/log/apache2/access.log`
8. For the icon, select **`logviewer`**
9. Check the **`Run in terminal`** checkbox

Using the wildcard Tapioca server

1. For the VM where you'd like to analyze an exploit or malware, connect the virtual network adapter to the same vmnet number as the LAN adapter in the wildcard VM. Power on this VM and ensure that it gets an IP in the 10.0.0.x subnet.
2. Ensure that the WAN adapter on the wildcard VM is disconnected. While the wildcard Tapioca VM will not route traffic originating on the LAN-side adapter to the WAN adapter, disabling the WAN adapter will ensure that it is not possible to reach the internet.
3. Snapshot the wildcard VM. If you are running a truly unknown exploit or malware, you'll have to assume that the wildcard VM may get compromised. Given that CERT Tapioca relaxes security controls to allow for simple network analysis, the wildcard VM should be considered insecure.
4. Ensure that the analysis VM is hitting the wildcard VM. e.g. open http://www.cert.org in the VM. It should load the Apache default page.
5. For HTTPS interception to work, you must install the mitmproxy CA certificate in the analysis machine:
    1. Visit http://mitm.it in the analysis VM.
    2. Download the mitmproxy CA certificate file for the platform you are on
    3. For Windows, place the certificate in a **manually-specified** certificate store: `Trusted Root Certification Authorities`
       For other platforms, refer to the mitmproxy guidance for installing the CA certificate.
    4. Confirm with IE that you can visit https://www.cert.org and you get no warning
    5. This only needs to happen once for any machine you're using for analysis
6. Launch Wireshark to capture traffic on the LAN adapter.
7. Open the exploit or any software you wish to observe in an isolated environment.
8. Observe the network traffic to see what network resources the exploit requests.
9. Place any resources as needed in the webroot `/var/www/html`
10. Repeat (starting at step 7) as necessary.
11. Revert the wildcard CERT Tapioca VM snapshot to the clean state created in step 3. above.

If you wish to **not** perform HTTP and HTTPS interception with mitmproxy, you can click the red 'X' on the far right of the icon list at the bottom of the screen. Click the Apache log tail icon to keep an eye on HTTP requests only. HTTPS requests will not be visible using this technique, as the HTTPS certificate will not be valid for the requests being made by the client.