



KNIGHT Device Technical Manual

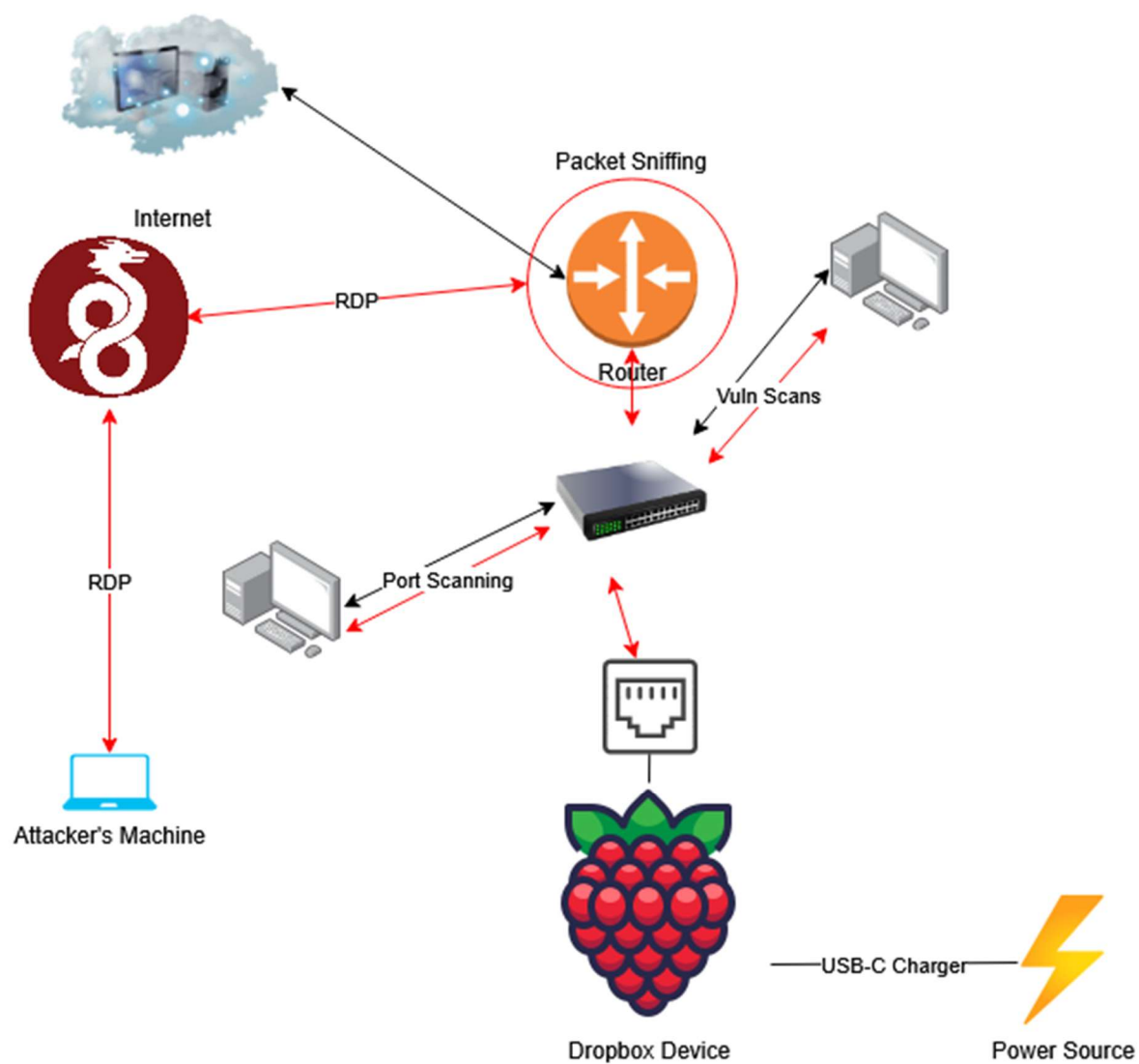
By Siege Engine

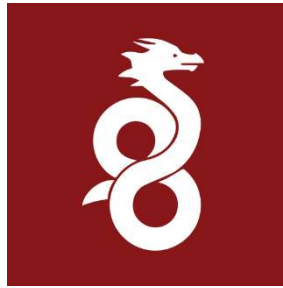
Matthew Alexander, Michael Astfalk, Nicolas Domico, Nathan Helmer,
and Timothy Pollock

Table of Contents

Wireguard	5
PiVPN.....	6
Remote Desktop Connection	6
Kali	7
Flask	7
NMAP	8
Raspberry Pi 4 Model B	8
USB Type C Power Supply	9
Ethernet Port	9
Installing from an Image	10
AWS and VPN Setup	11
Setting up AWS	11
Setting up VPN.....	14
File System Overview	17
Updating With Git	17
Python Packages and Dependencies.....	17
Manually Starting the UI	17
Connecting to Metasploit	18
Appendix 1: Description of Python Files and Functions	19
app.py	19
vuln_scan.py	19
vuln_scanner(ipaddr, ports='1-1000')	19
net_scan.py.....	19
port_scan(ipaddr, port_nums='1-1000', scan_flags='', path='').....	19
get_flags(line, ipaddr, ports)	20
meta.py	20
connect_meta()	20
search_exploit(cve).....	20

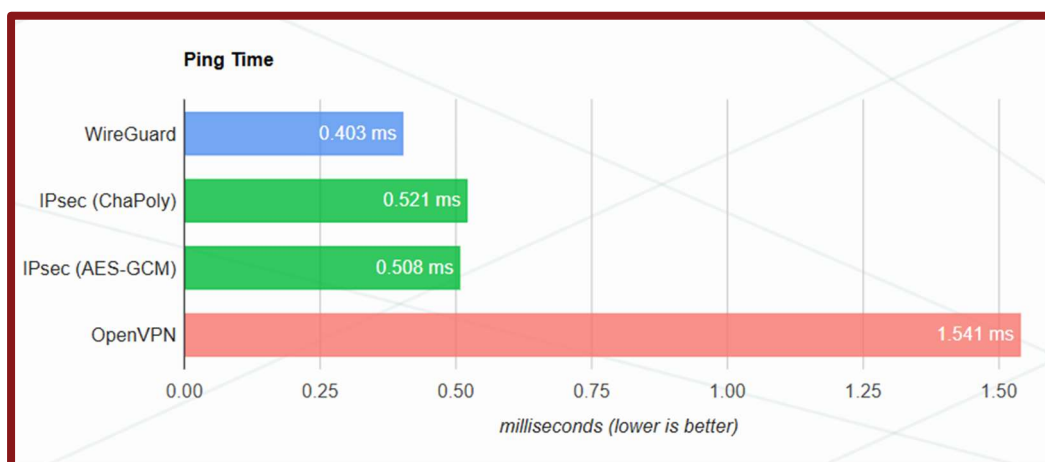
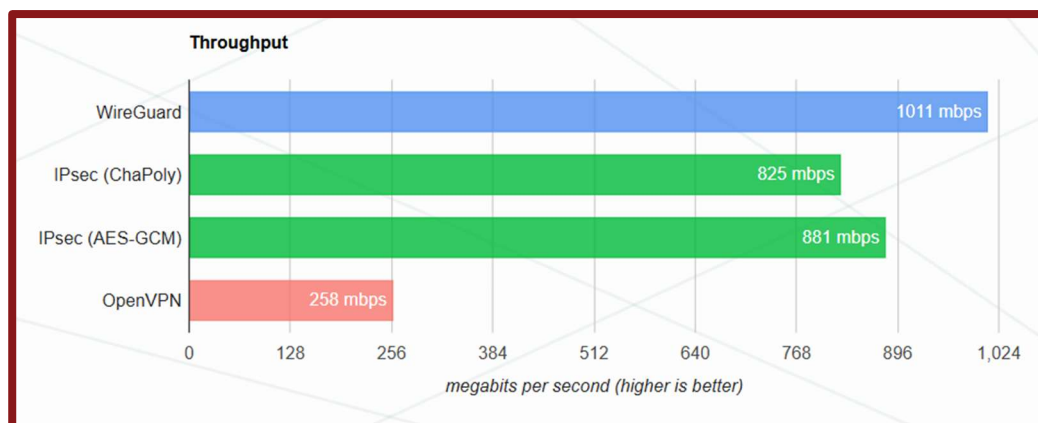
test.py	20
run_scans()	20
run_vulns()	20
osdetection.py	21
nmap_results_path()	21
vuln_results_path()	21
nmap_logs_path()	21
vuln_logs_path()	21
results_path()	21
file_zip_path()	21



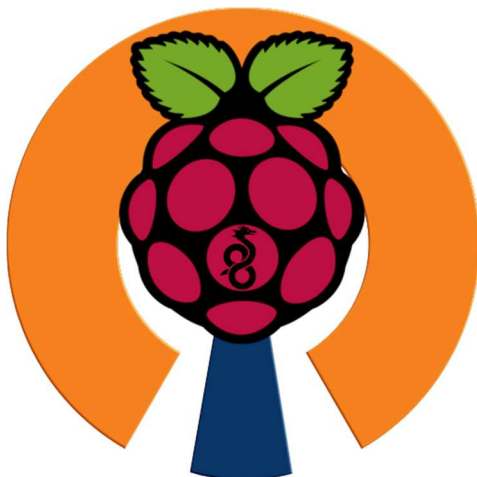


Wireguard

The KNIGHT team needed a way to connect to the KNIGHT device from anywhere. This meant a wide array of networks. A VPN is needed to accomplish this. WireGuard is a secure VPN that was practical due to the fact that a member of the KNIGHT team had previously used it. WireGuard is a free tool that is supported by Kali Linux out of the box. Another free option is OpenVPN, but WireGuard is faster.



Note that these tests were conducted by the WireGuard team. The network in which the KNIGHT team developed the product was not as fast as what is displayed in these images.



PiVPN

The KNIGHT Team used PiVPN to set up Wireguard due to its simplicity and ease of use. PiVPN simplified the setup by handling key generation, firewall rules, and network configurations, allowing the KNIGHT team to quickly deploy a secure VPN without extensive manual work. Even though the team had experience with manual WireGuard setup, using PiVPN saved time and reduced potential misconfigurations. Additionally, PiVPN's built-in client management made it easier to add or revoke access when needed.



Remote Desktop Connection

Although there are other services, the KNIGHT team recommends using the Microsoft preinstalled Remote Desktop Connection. The User can remotely connect to the KNIGHT

device from anywhere. This means that the user does not need to set up any input or output devices to the KNIGHT. All that would need to be connected to the KNIGHT device is the power source and a CAT5e/CAT6.



Kali

Kali Linux is a free, open-source operating system that is designed for penetration testing, security auditing, and other cybersecurity tasks. If the user would like to complete a task that is beyond the scope of the KNIGHT UI, they are encouraged to use the terminal and pre-installed packages on Kali Linux.



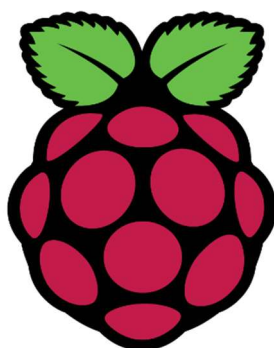
Flask

Flask is a Python-based web framework that helps developers build web applications. The KNIGHT team is hosting the KNIGHT UI via flask and the creation of a python virtual environment. Flask also allows for easier JSON and python scripting compatibility. This scripting is essential for many components in the KNIGHT UI.



NMAP

The KNIGHT UI works in tandem with the already-built scanning tool: NMAP. The KNIGHT device guides users through the process of configuring NMAP scans that will be run and then subsequently outputted on the KNIGHT UI.



Raspberry Pi 4 Model B

The KNIGHT device is a Raspberry Pi 4 Model B. This relatively small and inconspicuous computer is perfect as it is primarily geared toward educational penetration testing in a controlled environment. In such an environment, users can simulate “leaving the device behind” to conduct a stealth penetration test.

The Raspberry Pi 4 Model B uses 8 GB of RAM and has a 64 GB Micro SD Card for storage. Taking into account installed programs, the Pi should be left with about 40 GB of free storage. RAM and Storage on the device should be sufficient for most operations the KNIGHT can perform. Monitors for CPU usage and device storage space have been added to the top right of the Kali Desktop page. Ensure that the device does not exceed its storage or its CPU capacity.



USB Type C Power Supply

The Raspberry Pi uses a USB Type C for the power supply.



Ethernet Port

The device will need to be physically plugged into the network with an ethernet cable in order to connect to the internet. Security protocols that force devices to authenticate before connecting to the network such as Extensible Authentication Protocol (EAP) may prevent the device from connecting to the internet and connecting back to your attack machine. Wi-Fi adapters could be used to connect the KNIGHT to the internet; however, the KNIGHT programs have not been officially tested to work over a wireless connection.

Installing from an Image

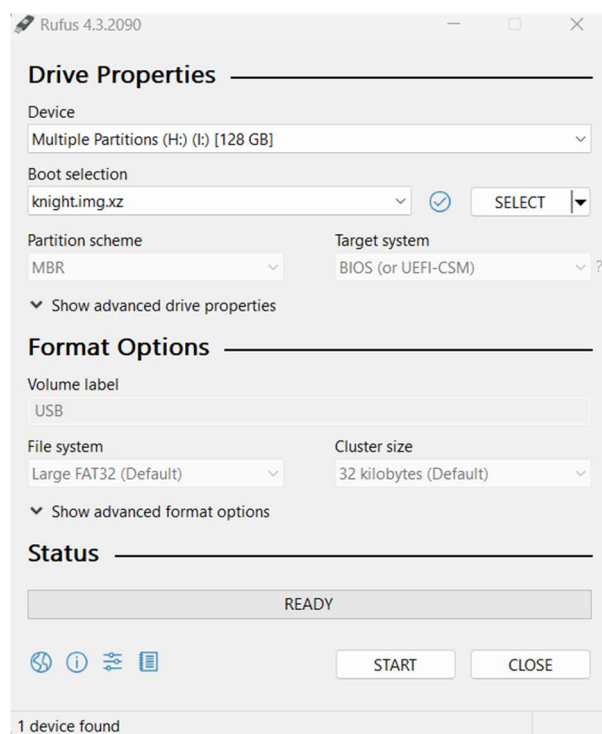
Before installing KNIGHT onto your Raspberry pi, you will need a few things:

- knight.img.xz (Modified Kali ARM image)
- MicroSD card (64gb recommended)
- MicroSD card reader
- A computer

1. Download Rufus or Balena Etcher (<https://rufus.ie/en/> OR <https://etcher.balena.io/>)

- We will be using Rufus for this guide, but the process should be similar

2. Open Rufus and select the MicroSD card and knight.img.xz



3. Click START

- This will flash the image to the MicroSD card.

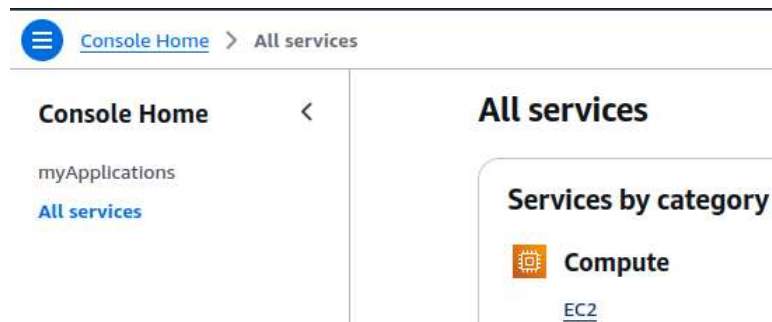
4. When finished, install the MicroSD card into the Raspberry Pi and start it up. After this you're finished and can now access the KNIGHT UI.

AWS and VPN Setup

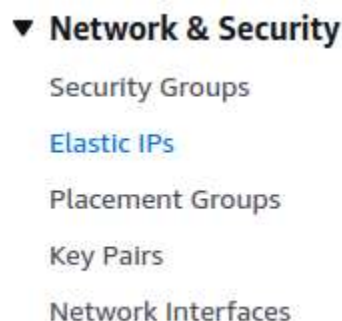
Setting up AWS

Before the VPN can be set up, a virtual machine will be needed to host it on. For these purposes, AWS will be used, however, any cloud provider will work.

1. After you create an account with AWS, navigate to the EC2 tab



- a.
2. In the left menu find Elastic IPs



- a.
3. In the top right click “Allocate Elastic IP address” and click “Allocate” on the next page. This will be used in a later step to provide a static IP address for use with the VPN.



- a.
4. Go back to the EC2 page and click the “Launch Instance” button.

Launch a virtual server


To get started, launch an Amazon EC2 Instance, which is a virtual server in the cloud.



- a.

5. Listed below are multiple Quick Start options and the option to set a name for the VM. In this case, the Ubuntu Quick Start option should be chosen.

a.

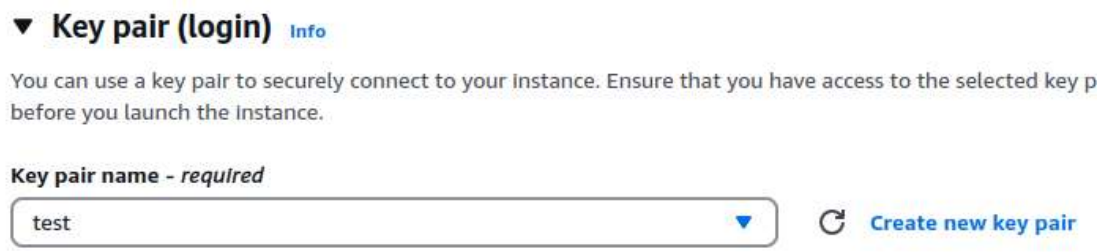


The screenshot shows the 'Quick Start' tab in the AWS console. It displays six options: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, and SUSE Linux. The 'Ubuntu' option is highlighted with a blue border and the Ubuntu logo. Below the options, the 'Name and tags' section is visible, with the 'Name' field containing 'VPNTEST'.

b.

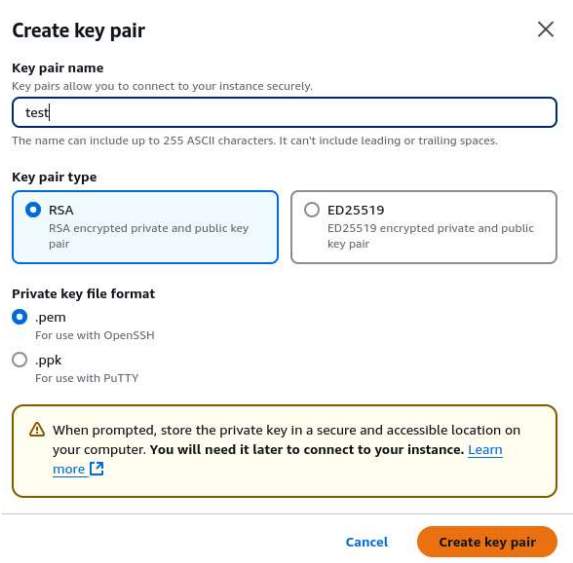
6. Next, create a Key pair for logging in. The KNIGHT Team is not using it, but it is recommended to make one.

a.



The screenshot shows the 'Key pair (login)' section in the AWS console. It includes a description: 'You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.' Below this, the 'Key pair name - required' field is set to 'test'. To the right is a 'Create new key pair' button.

b.



The screenshot shows the 'Create key pair' dialog box. It has a title bar with a close button. The 'Key pair name' field is set to 'test'. Below it, the 'Key pair type' section shows 'RSA' selected, with a description: 'RSA encrypted private and public key pair'. The 'Private key file format' section shows '.pem' selected, with a description: 'For use with OpenSSH'. At the bottom, there is a warning box: 'When prompted, store the private key in a secure and accessible location on your computer. You will need it later to connect to your instance. Learn more'. At the bottom right are 'Cancel' and 'Create key pair' buttons.

7. Everything else will stay as default, once done click "Launch Instance"
8. Go back to the Elastic IP menu as shown in Step 2 and click on the Elastic IP we created.

<input type="checkbox"/>	Name	Allocated IPv4 addr...	Type
<input type="checkbox"/>	-	18.216.104.44	Public IP

- a.
9. Click on “Associate Elastic IP Address”, select the instance that was created and its private IP address. When done, click on “Associate”

Associate Elastic IP address

a.

Instance

Private IP address

The private IP address with which to ass

b.

Associate

c.

10. Click on Instances in the left menu

▼ Instances

[Instances](#)

a.

11. Click the Instance we created and go to the security section

☒ VPNTTEST [i-0a1e31e082493b5b1](#)

a.

Details | **Status and alarms** | **Monitoring** | **Security**

▼ Security details

IAM Role

-

Security groups

☐ [sg-0a91c87123cf3a781 \(launch-wizard-2\)](#)

b.

12. Click the security group listed and click “Edit inbound rules”



Manage tags

Edit inbound rules

a.

13. Add two rules for port 51820, one TCP and one UDP. Add another rule for port 22 so we can access the VM. When done, click “Save rules”

allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

a.

14. NOTES:

- It is recommended to specify an IP address for SSH.
- Once finished setting up the VPN, ensure the SSH security rule is removed.

Setting up VPN

To set up Wireguard, the PiVPN script will be used. It is found at <https://www.pivpn.io/>

- Navigate to the EC2 Instance we created and click “Connect”



a.

- Leave everything at default on the next page and click “Connect”

- A terminal window will now be in your browser.

- Paste “curl -L https://install.pivpn.io | bash” into the terminal. (This command is found on the pivpn website)

```

Welcome to Ubuntu 24.04.2 LTS (GNU/Linux 6.8.0-1024-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Mon Mar 31 16:14:53 UTC 2025

System load:  0.00      Processes:    109
Usage of /:   33.5% of 6.71GB   Users logged in:  0
Memory usage: 28%          IPv6 address for enx0: 172.31.29.146
Swap usage:   0%

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.
   https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.
24 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

*** System restart required ***

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-29-146:~$

```

i-0a1e31e082493b5b1 [VPNTST]
PublicIP: 18.216.154.44 PrivateIP: 172.31.29.146

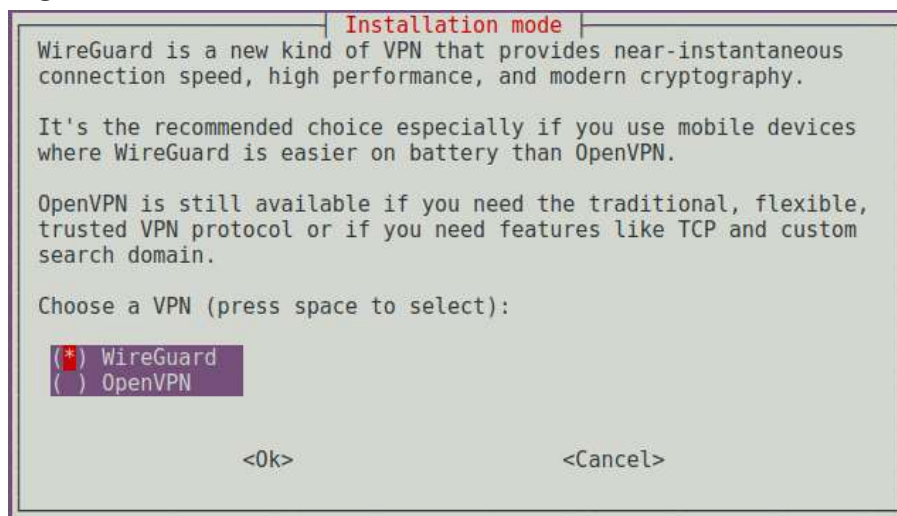
b.

- This will bring the device into an automated installer, when here press enter for “Ok”



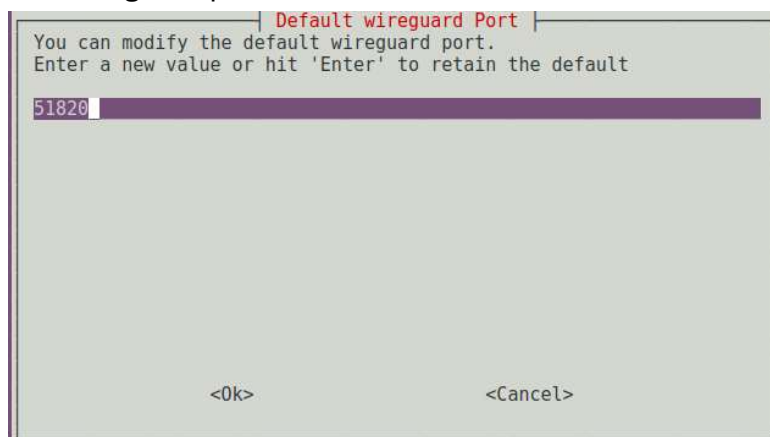
a.

4. Accept the default configuration until the Installation Mode screen. Here, ensure that Wireguard is selected.



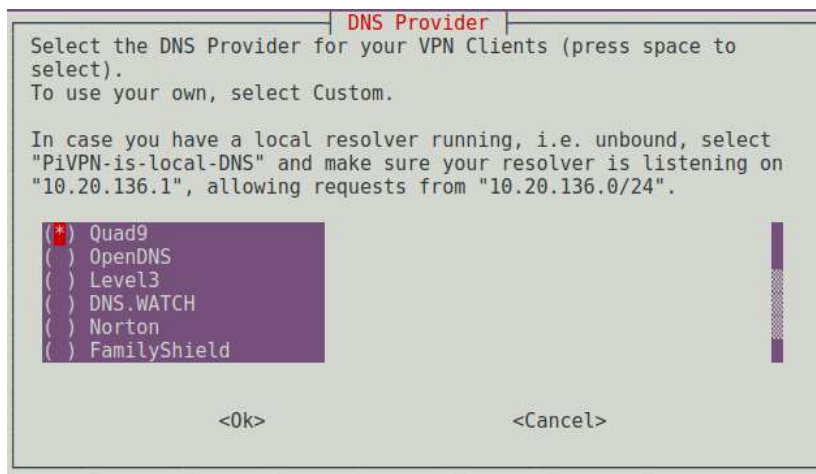
a.

5. Ensure the wireguard port is set to 51820 and click enter to continue.



a.

6. Select a DNS provider. The default will be fine.



a.

7. From here on out, continue with the default options until you reach the “Installation Complete!” screen. When asked to reboot, select yes.



a.

8. Once the system is rebooted and it has been reconnected into, run the command “pivpn add”. Pivpn will walk through the steps of creating a configuration file. Once this is done, the VPN can now be connected into.

```
ubuntu@ip-172-31-29-146:~$ pivpn add
Enter the Client IP from range 10.20.136.2 - 10.20.136.254 (optional): 10.20.136.2
Enter a Name for the Client: TestClient
::: Client Keys generated
::: Client config generated
::: Updated server config
::: WireGuard reloaded

=====
::: Done! TestClient.conf successfully created!
::: TestClient.conf was copied to /home/ubuntu/configs for easytransfer.
::: Please use this profile only on one device and create additional
::: profiles for other devices. You can also use pivpn -qr
::: to generate a QR Code you can scan with the mobile app.
=====
```

a.

9. NOTES:

- Config files used to connect to the VPN are stored in /home/ubuntu/configs
- For use with the KNIGHT, at least two config files will need to be created: One for the KNIGHT and one for the client device. A single config file cannot be in use by two separate devices at the same time.

File System Overview

Updating With Git

Updated versions of the KNIGHT UI can be pulled from the Git repository

(<https://github.com/NathanHelmer/KNIGHT>) by running the command `git pull` while in the KNIGHT directory.

Python Packages and Dependencies

The KNIGHT UI requires Python 3 to be installed on the device. A complete list of required Python packages can be found under KNIGHT/requirements.txt. The UI program runs Python in a virtual environment. This is set to run by default on the KNIGHT device, however if you need to modify or reconstruct a virtual environment on the device, follow these steps:

1. Create the virtual environment run these commands in the home directory:

```
python -m venv venv
```

2. Activate the virtual environment:

```
source venv/bin/activate
```

3. Install required packages:

```
source venv/bin/activate
```

```
pip install -r requirements.txt
```

If pip was used to install additional packages in your virtual environment, you can update requirements.txt by running:

```
pip freeze > requirements.txt
```

Manually Starting the UI

While the KNIGHT device will run its UI program on start-up, it may be necessary to manually start the UI using Python. Web services are managed by Flask through

KNIGHT/scripts/app.py. While in the KNIGHT/scripts/ directory, run the command to start the service:

```
Python3 app.py
```

Check Python's output to ensure that Flask has started successfully. The UI will by default run on the local host on port 5000 (127.0.0.1:5000) which can be accessed on your web browser.

The app.py file will render html pages and make function requests when a specific URL path is reached. For instance, the home directory at 127.0.0.1:5000/ will render the index.html file. Results for Nmap scanning and vulnerability scanning will be made using the /run-nmap-scan/ and /run-vuln-scan/ paths respectively. If you need to troubleshoot scanning results for Nmap and vulnerability scanning, access these paths in your browser to view the raw Json results of the program. Example paths for Nmap scanning and vulnerability scanning from the KNIGHT device are given below:

Nmap Scan URL Path:

```
127.0.0.1:5000/run-nmap-scan/?ip=scanme.nmap.org&ports=1-1000&cmd=nmap+-p+1-1000+-sV
```

Vulnerability Scan URL Path:

```
127.0.0.1:5000/run-vuln-scan/?ip=scanme.nmap.org&ports=1-1000
```

Connecting to Metasploit

Vulnerability scanning uses pymetasploit3 and MsfRpcClient to connect to Metasploit modules. MsfRpcClient should already be automatically running on the device. If problems are encountered with connecting to Metasploit, the MsfRpcClient may need to be restarted. Use the following steps:

1. Ensure that the msgrpc module is loaded by opening Metasploit with ``msfconsole``.
2. In the Metasploit console, run the command ``load msgrpc Pass=knight``. This will load the service with the KNIGHT's default password for MsfRpcClient ('knight').

3. Start the service using the command ``msfrpcd -P knight``. This should start the service on port 55553.

If problems continue to occur, it may be helpful to check the error output from the Python code. Note that pymetasploit3 requires msgpack v. 0.6.2 and is not verified to work with other versions of msgpack. Other msgpack modules or versions may conflict with pymetasploit3 and should not be installed. Documentation for the pymetasploit3 Python module can be found at <https://pypi.org/project/pymetasploit3/>.

Appendix 1: Description of Python Files and Functions

app.py

Description: runs Flask applications at the routes specified in '@app.route()'

vuln_scan.py

Description: Runs an nmap vulnerability scan using a script and returns the results.

`vuln_scanner(ipaddr, ports='1-1000')`

Precondition: ipaddr is a string of an IP address or IP address range.

Postcondition: returns an nmap scan object of scan results on ipaddr.

net_scan.py

Description: Runs a Nmap scan after taking an IP address and port range as input. Must be run with administrator privileges.

`port_scan(ipaddr, port_nums='1-1000', scan_flags='', path='')`

Precondition: ipaddr is a string for the IP address of the form '255.255.255.255' and port_nums is a string for the port range of the form 'x-y'

Postcondition: writes nmap scan results to nmap_results.txt

`get_flags(line, ipaddr, ports)`

Precondition: line is a string for the command line command, ipaddr is a string of the target IP range, and ports is a string of the port range

Postcondition: returns a string of the optional scan flags from the command

`meta.py`

Description: Handles connections to Metasploit through pymetasploit3 using MsfRpcClient.

`connect_meta()`

Precondition: The MsfRpcClient service is running on the device.

Postcondition: Returns an object for the MsfRpcClient connection or null if the connection fails.

`search_exploit(cve)`

Precondition: cve is a string for the cve name to be searched.

Postcondition: returns a list of exploit objects for matching exploits.

`test.py`

Description: Automated testing for the KNIGHT UI backend.

`run_scans()`

Precondition: none

Postcondition: runs sample nmap scans

`run_vulns()`

Precondition: none

Postcondition: runs sample vulnerability scans

osdetection.py

Description: returns file paths given the host's OS. Note: when running the program in linux, run it from ~/PATHTOFLASK --app ~/PATHTOAPP.PY run

nmap_results_path()

Precondition: none

Postcondition: returns the file path for Nmap scan results based on the host OS (Windows or Linux)

vuln_results_path()

Precondition: none

Postcondition: returns the file path for vulnerability scan results based on the host OS (Windows or Linux)

nmap_logs_path()

Precondition: none

Postcondition: returns the file path for Nmap logs directory based on the host OS (Windows or Linux)

vuln_logs_path()

Precondition: none

Postcondition: returns the file path for Vuln logs directory based on the host OS (Windows or Linux)

results_path()

Precondition: none

Postcondition: returns the file path for results directory based on the host OS (Windows or Linux)

file_zip_path()

Precondition: none

Postcondition: returns the file path for all_files.zip based on the host OS (Windows or Linux)