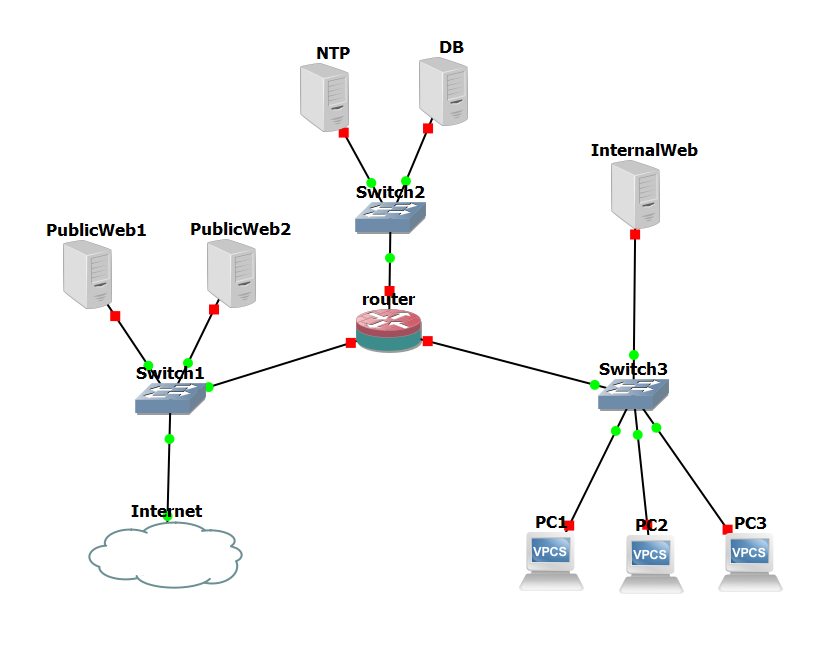
## Network Spec



* Determine what are the internet gateways, what hosts/servers will be public and which ones will be private
* Firewall rules
* host/server OS (Ubuntu, Windows…)
* Who needs access to the network as a dev
* How many standby servers/subnets are needed (for example for honeypots)
* How many IPs we need to reserve (Elastic IPs)
* From subnet1 you can only access subnet2 but not subnet 3, and vice versa. So to go from 1 to 3 or 3 to 1 you have to go through 2
* Only allow already established TCP connections to defender, allow all outgoing traffic from

## Vulnerability Distribution

* Create install scripts?
* Manual input (i.e. weak login credentials)
* Specified in the routing tables and network ACL

## Telemetry

* Intrusion detection system
  + Requirement
    - Traffic encryption
    - Custom rules
    - Configurable alarms
* File monitoring

## Admin Interface

* Deploy/remove instances
* Add/remove user access
* Modify network ACL and Security Group rules (AWS Firewall and network topology)
* Direct SSH access to the instances and ability to modify their state (Add vulnerabilities, code modification…)
* Access to traffic logs (monitoring)

## Defender Interface

* Defender IAM user
* No full knowledge of vulnerabilities in the network
* Monitoring: automate network information through terraform scripts using VPC Flow Logs or CloudTrail API and analysis of those logs with AWS CloudWatch

## Code Structure

* Software needed: AWS CLI, Terraform, YAML
* Automating honeypot deployment: <https://docs.aws.amazon.com/opsworks/latest/userguide/workingcookbook-extend-cron.html>

Todo:

* Bullet points on overleaf document for defender and attacker interface
* Terraform scripts
* Cron Jobs for Honeypot deployment
* ~~NAT Gateway~~
* ~~Install c2 code and make sure that 17737 port is open. Controller is the client, the rest are servers~~
* ~~Set up NTP server~~
* Slides
* ~~Look into vulnerability exploits~~
* ~~DB~~
  + ~~Open postgres~~
  + ~~Install postgres (password: postgres)~~
  + ~~Add user and open an ssh with that user~~
* ~~Create a separate NTP and DB Security Groups~~
  + ~~NTP: NTP port, 22 ssh, two tcp ports 17737, 80~~
* ~~Sanitize ports and make a table to show the state of ports so that Adam can check it~~
* ~~Autorecon for network scan (autorecon.py)~~
* ~~linPEAS finds vulns for privilege escalation~~
* Add web honeypot and setup again
* Figure out bastion host for internal WEB subnet and CONTROLLER (if needed?)
* Redo DB so that it’s in the right subnet
* Upgrade public server instances

Deployment Notes:

* Firewalls implemented only through security groups, NACLs left untouched (allow all inbound and outbound traffic)
* Pre-established TCP connections for controller by setting no Inbound Rules, so only the controller can start TCP connections
* Firewalls:
  + Public:
    - Inbound traffic, allow all
    - Outbound traffic, allow all
  + NTP/DB:
    - Inbound traffic, from Public, WEB, Controller
    - Outbound traffic, allow all
  + WEB:
    - Inbound traffic, from Controller, NTP/DB
    - Outbound traffic, to Controller, NTP/DB
  + Controller:
    - Inbound traffic, deny all
    - Outbound traffic, allow all
* Look into terraformer to automate terraform scripts from deployment
* Use yaml files to auto generate attack graph
* Public NACL and Security Groups for deployment