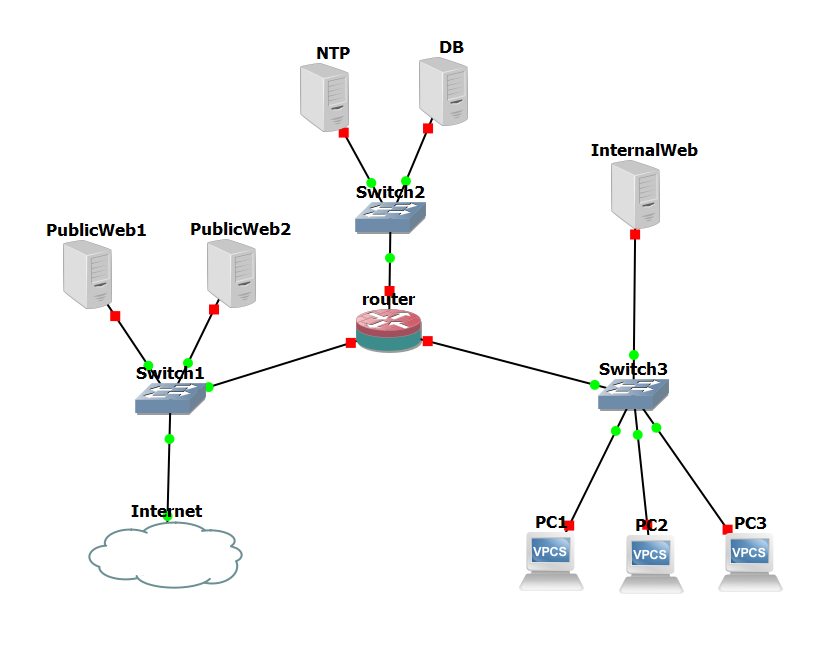
### Motivational Example



TODO (Adam):

* redefine graph - user hosts = public network with 2 web servers; no firewall or router nodes; two enterprise servers (NTP & DB); one operational server (WEB); three operational hosts (linux user systems)
* Draw a new attack graph over this network (with more than three nodes for more complex attacks on some machines)
* Decide on hard vulnerability

A web server

A database server

A user machine

linux/windows

#### Story

*Background information about the vulnerabilities, past exploits, and real-world impact*

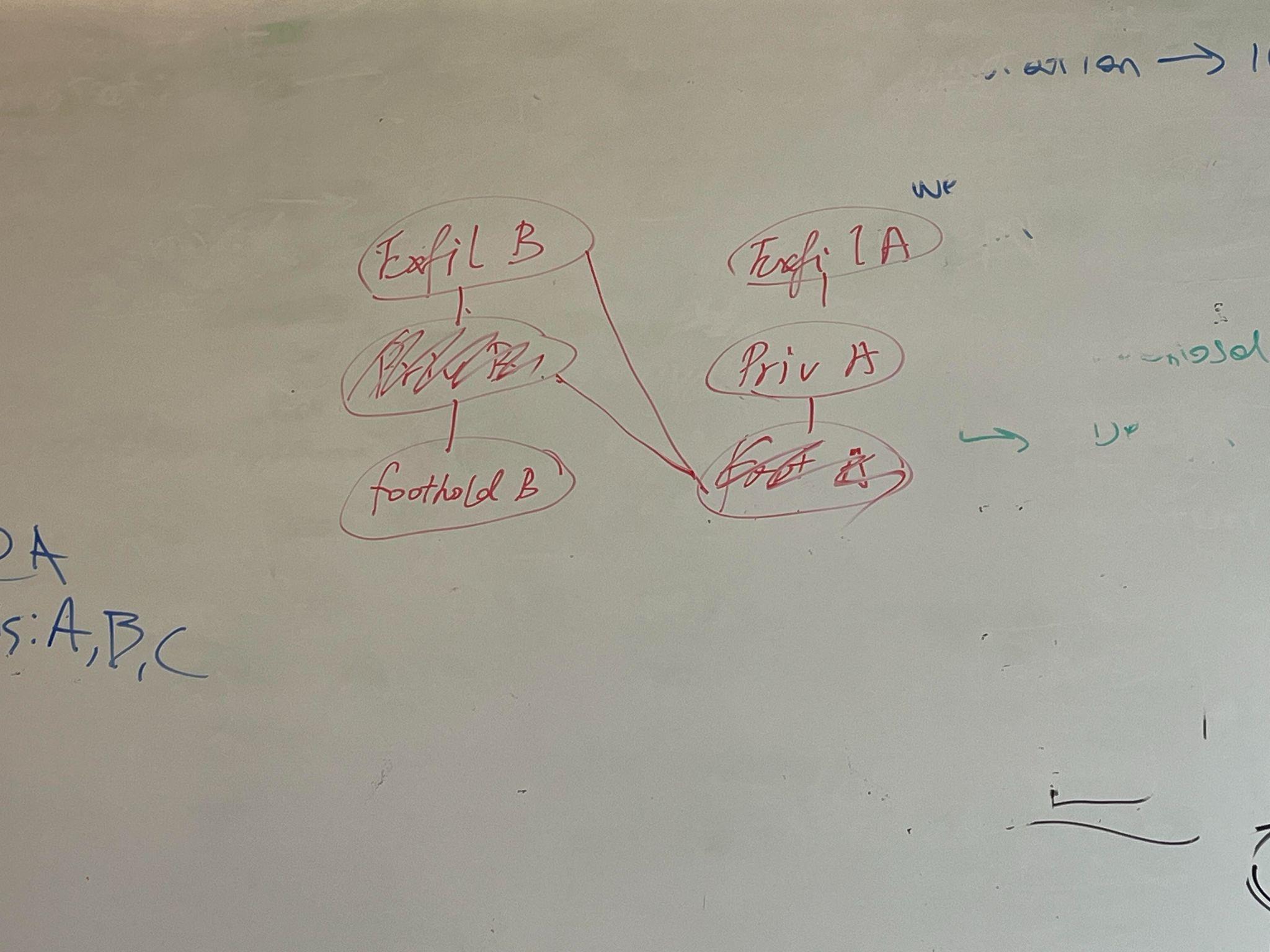
#### Vulnerabilities

*4 or more sets of vulnerabilities, including public facing ones for the attacker to establish a initial foothold, easy-medium-hard, noisy-quiet*

*Order by difficulty, footprint*

* Foothold
  + Wordpress Backup Migration <= 1.3.7 - Unauthenticated Remote Code Execution (public facing) [poc: <https://github.com/Chocapikk/CVE-2023-6553>] - Easy & Noisy
  + Weak logon credentials for SSH (local network only) - Easy & Noisy
  + PostgreSQL UDF RCE (<https://book.hacktricks.xyz/pentesting-web/sql-injection/postgresql-injection/rce-with-postgresql-extensions>) *medium difficulty and noise*
  + Cayin CMS NTP Server RCE (use exploit/linux/http/cayin\_cms\_ntp) - easy & medium noise
* Privilege Escalation
  + SUID executable - easy & quiet
  + Cron job executed as root running world writable script - easy & quiet
  + Vulnerable sudo version (<https://exploit-notes.hdks.org/exploit/linux/privilege-escalation/sudo/> <=1.28 POC: sudo -u#-1 /bin/bash) - easy & quiet
* Potentially Challenging - TBD
  + <https://www.exploit-db.com/exploits/48052>
  + <https://www.exploit-db.com/exploits/47673>
  + <https://www.exploit-db.com/exploits/46215>
  + Other: Search "buffer overflow" in <https://www.exploit-db.com/?platform=linux>
* Data stored unencrypted (exfiltration) - easy & quiet

#### Attack graph of each node with vulnerabilities



### Graph Definition (𝑁 , 𝐸, 𝑉 , 𝑃)

Node (N):  *low priv, high priv, data exfiltrated (3 nodes per machine)*

Edge (E): *foothold, privilege escalation, data exfiltration*

Value (V): *value of exfiltrated data*

Probability (P): *Given beforehand to the defender and learned by the attacker based on its scans.*

### 

### State

### 

### Attacker’s Observation (Oa)

| Abstract Information on the Attack Graph | Potential Implementations (Information about the network/defender and reconnaissance techniques to collect them) | Example |
| --- | --- | --- |
| Attacker Visible edges | 🦶Open ports (port scan) |  |
|  | 🦶Service detection (banner grab; manual interaction) |  |
|  | 🦶Network hosts (host scan) |  |
|  | ⬆️/📨 Files on host (ls) |  |
|  | ⬆️ Running processes (ps) |  |
|  | ⬆️ System configs (various) |  |

### Defender’s Observation (Od)

| Abstract Information on the Attack Graph | Potential Implementations (Information about the network/attacker and telemetry techniques to collect them) | Example |
| --- | --- | --- |
| Probabilistic position of attacker <I(attacker is on node A)> (matrix of timestamps over the network nodes (with privilege level/attack step) they have visited) | Traps (i.e. honeypots) + IDS | 100% knowledge of position when trap set off, diminishes until attacker sets off another trap |
| Nodes |  |  |
| (Edges)Transition Probabilities | Given ahead of time | We’ll do some research and try to come up with reasonable transition probabilities given the possible vulnerabilities in the network we build |
| Updated Nodes |  |  |
| Updated edges |  |  |
| Budgets <for each action> |  |  |

### Attacker’s Action Space (Aa)

Enumerate the combinations: cron job or SSH bruteforce can be run concurrently with anything else

List <(cron job, SSH, X), (cron job, X), (SSH, X), (cron job, sudo), (cron job, SUID), (cron job, Cayin), (cron job, postgreSQL), (cron job, Wordpress), (cron job, unencrypted exfil), (SSH, sudo), (SSH, SUID), (SSH, Cayin), (SSH, postgreSQL), (SSH, Wordpress), (SSH, unencrypted exfil), (cron job, SSH, sudo), (cron job, SSH, SUID), (cron job, SSH, Cayin), (cron job, SSH, postgreSQL), (cron job, SSH, Wordpress), (cron job, SSH, unencrypted exfil), (add potentially challenging attacks when chosen)>

| Abstract Action on the Attack Graph | Potential Implementations | Cost (#step [maybe 1 or 2]) | Example |
| --- | --- | --- | --- |
| Traverse edge | Gain foothold |  | Gets a low priv shell |
| Traverse edge? | privesc |  | Gets a high priv shell |
| Traverse edge? | exfil |  | Gets data off machine |
| Stay in same node | Do nothing |  |  |

### Defender’s Action Space (Ad)

List <(cmd, host)> // all systems will have valuable data except ntp, with DB having the most valuable data

| Abstract Action on the Attack Graph | Potential Implementations (Defensive or Deceptive Techniques) | Cost | Time delay | Deployment stage | Benefit of real-time deployment? | Example |
| --- | --- | --- | --- | --- | --- | --- |
| Adding a fake edge to a low priv node | Start honey service (foothold) 🕵️ | 0 if no SSH already  otherwise >0 | 3 seconds (TODO: need to measure) | before they have foothold | doesn't inhibit other SSH server lowers chance of detection | Cowrie SSH server |
| Adding a fake edge to a high priv node | Add fake privesc 🕵️ | 0 | <1 second | before privesc | lowers chance of detection | SUID on deceptive executable (e.g. cp) |
| Adding a fake edge to a data exfil node | Create fake data (exfil) 🕵️ | 0 | <1 second | before exfil | lowers chance of detection | Make a look-alike of whatever the real data is, but change the important values (and maybe add a mark that can be detected during exfil) |
| Ending the game | Block IP 🛡️ | 0 |  |  |  |  |
| Do nothing |  | 0 |  |  |  |  |

### 

### Reward Scheme for Attacker Ra(s)

(Potential objectives of attackers in the real world? Cost?)

Objectives:

1. Steal data (positive points)
2. Don’t touch trap (negative points)

Reward function:

### Reward Scheme for Defender Rd(s)

(Potential objectives of defenders in the real world? Cost?)

Objectives:

1. Have traps triggered (positive points)
2. Don’t lose data (negative points)

Reward function:

### Game tree

#### State + next player

* What is the entire state of the game?

#### Legal moves

* What moves are legal based on the game state?
  + Accessible nodes/ traversable edges from current position

Assume attacker and defender take turns and defender can choose to do nothing as an action

Draw a tree showing all *reasonable* decisions of attackers and defenders following some heuristic, layers alternating between attacker and defender turns.

#### Tree size - to get a sense of the state space and learning complexity

#### Score of game paths

Value we assign to each state, how many points will the defender lose at each step

### Heuristic attacker/defender

#### Now start coding

### Simulation Experiment Design

#### Algorithms

#### Train

#### Evaluation

### Sources to attract attackers

Attract (incentivization)

Payment