

Golfing with Dragons

Building Secure Environments for CTFs

Jared Stroud – Lead Security Engineer

Dan Szafran – Security Engineer

May 2023



Agenda

- **What's a CTF?**
- **Threat Modeling for CTFs**
- **Identifying & Implementing Controls**
- **Key Lessons Learned**



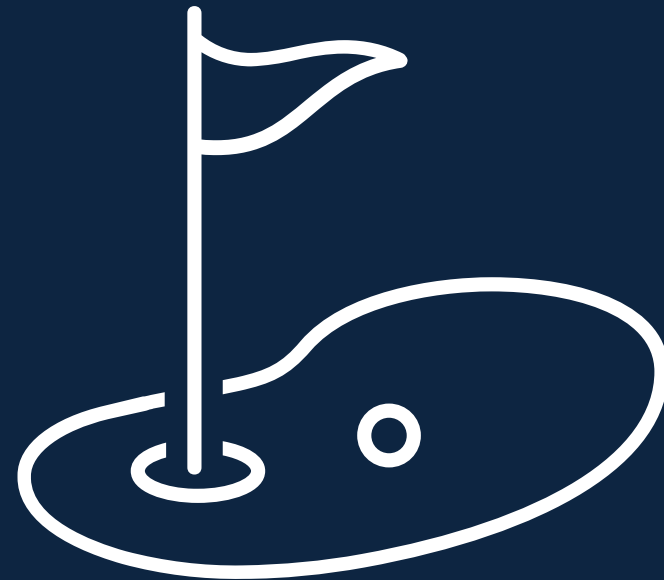
DevSecOps & Capture The Flags Competitions

- Capture The Flag (**CTF**) competitions are prominent events in the Cyber Security field for contestants to demonstrate their technical skills for prizes, jobs and notoriety.
- These events range in variety, but popular categories include:
 - **Jeopardy**: Competitors download and solve a stand-alone challenge for points.
 - **Attack/Defend**: Competitors defend infrastructure while attacking others.
 - **King of The Hill**: Competitors attack infrastructure to place their “flag”, a unique identifier, in a specific file to score points.



Ghidra Golf – A Spin on Jeopardy Style CTF

- Ghidra Golf is **Reverse Engineering** CTF event with focus on Ghidra Script development.
 - Ghidra Scripts are **Java/Python Applications**.
- The contestant's goal is to develop Ghidra Scripts to identify, parse, decrypt/decode or otherwise accomplish a specific reverse engineering task.
- Contestants are provided with a binary to download, reverse engineer and to test their code against, before submitting their Ghidra Script for **automated evaluation**.



Ghidra Golf – A Spin on Jeopardy Style CTF

- *“Contestants are provided with a binary to download, reverse engineer and to test their code against, before submitting their Ghidra Script for **automated evaluation.**”*
- Is a fancy way of saying....
 - Executing **arbitrary Python/Java applications** from strangers on the internet

Threat Modeling: Identifying risks to the competition

Threat Modeling **YOUR** Competition



Rogue Competitor

How do we minimize arbitrary code execution?



Infrastructure **Availability**

How do we recover from disaster?
How do we make our challenges available globally?

```
Program received signal SIGSEGV, Segmentation fault.
0x0000000000401116 in main ()
Missing separate debuginfos, use: dnf debuginfo-install
(gdb) disassemble
Dump of assembler code for function main:
   0x0000000000401105 <+0>:   push   %rbp
   0x0000000000401107 <+1>:   mov    %rsp,%rbp
   0x000000000040110a <+4>:   movq   $0,-0x8(%rbp)
   0x0000000000401112 <+12>:  mov   -0x8(%rbp),%rax
=> 0x0000000000401116 <+16>:  movl  0x6c6f66c,(%rax)
   0x000000000040111c <+22>:  mov   $0,%eax
   0x0000000000401121 <+27>:  pop   %rbp
   0x0000000000401122 <+28>:  ret
```

Broken Challenges

How do we Identify and recover from bugs?

Rogue Competitors – How to *safely* enable arbitrary code execution

- How could a competitor disrupt the environment for other competitors?
 - How do we limit this potential disruption?
- How do we scan submitted code to prevent undesired execution?
 - How do we become notified of “flagged” code?

Infrastructure **Availability**

- Most CTFs are run by volunteers in their free time.
 - Not everyone has a cloud budget.
 - How to enable remote access to volunteers?
- How do you leverage existing hardware to maximize reach to your competition?
- How do you monitor for issues and respond accordingly?

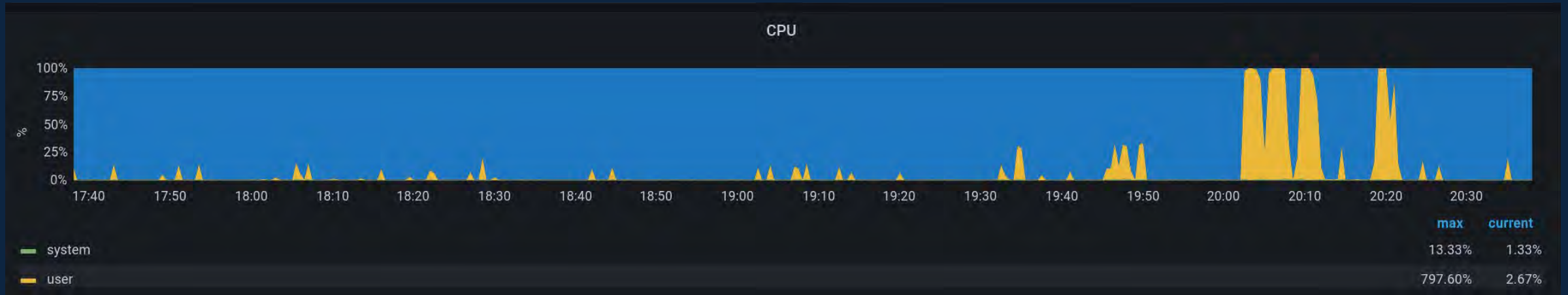


Resolving **Broken** Challenges

- Software has **bugs**, CTF challenges have **bugs**.
- A competitor is spending their valuable conference/free time with you.
 - They are **our customers**.
 - Ensuring they have a positive experience is critical to continuing running these events
- Identifying ways to resolve, redeploy and reengage competitors is critical to a positive experience.

Implementing **Controls** to **Reduce Risk**

Infrastructure **Monitoring** – Identifying Broken Challenge



Implementing **Controls** Throughout the Competition to Reduce Rogue Competitor Impact



Application

DAST/SAST
Scanning



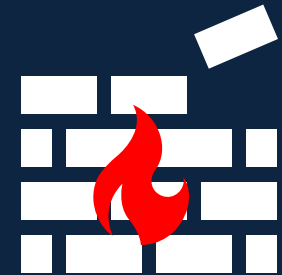
Container

Image Scanning
&
Runtime Hardening



Host

Vulnerability Scanning
&
Hardening



Network

Firewalls/Isolation

Implementing **Controls** Throughout the Competition



Source Code Application Scanning to identify low hanging fruit.

Examples:

- YARA Rules, custom implementation
- Bandit, Open Stack's vulnerability Scanner for Python
- Integration opportunities:
 - CI/CD Pipelines
 - Git commit hooks
 - Ad hoc scans

Example – Bandit Scans

```
>> Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b113_request_without_timeout.html
Location: GhidraGolf.py:168:16
167 # post data to CTFd endpoint
168 requests.post(self.ctfd_endpoint,
169               json=submission_json)
170 logging.critical(f"YARA rule hit: {yaretval} for submission {gsObj.sub_id}")
```

```
>> Issue: [B603:subprocess_without_shell_equals_true] subprocess call - check for execution of untrusted input.
Severity: Low Confidence: High
CWE: CWE-78 (https://cwe.mitre.org/data/definitions/78.html)
More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b603_subprocess_without_shell_equals_true.html
Location: GhidraGolf.py:224:8
223 # Danger here!!! building execution script from user arguments!
224 subprocess.call(["/bin/bash", "-c", ghidra_runner])
225 return True
```

```
>> Issue: [B113:request_without_timeout] Requests call without timeout
Severity: Medium Confidence: Low
CWE: CWE-400 (https://cwe.mitre.org/data/definitions/400.html)
More Info: https://bandit.readthedocs.io/en/1.7.5/plugins/b113_request_without_timeout.html
Location: GhidraGolf.py:246:26
245 # post data to CTFd endpoint
246 results = requests.post(self.ctfd_endpoint,
247                        json=submission_json)
248 logging.info(results.text)
```

Example – YARA Rule Implementation

```
rule cmdrules {
  meta:
    description = "detect submission script attempting to use command line tools"

  strings:
    // Shells
    $bash = "/bin/bash"
    $sh = "/bin/sh"
    $zsh = "/usr/bin/zsh"
    $chsh = "/usr/bin/csh"
    $sysexec = "system.exec"
    //$python = "python" // Should cover "python3" as well

    // Third Party Network Commands
    $curl = "curl "
    $wget = "wget "
    $ftp = "ftp "
    $netcat = "netcat "
    $ssh = "ssh "
    $scp = "scp "
    $telnet = "telnet "
    $nmap = "nmap "
```

https://github.com/ghidragolf/ctfd-ghidragolf/blob/main/conf/consumer/yara/rules/cmd_rules.yar

Implementing **Controls** Throughout the Competition



Image Scanning & Capability Enforcement

Examples:

- Trivy, Open-Source vulnerability scanner for containers
- Drop all capabilities
 - add only those that are needed
- OWASP Container Auditing

Example – Old Redis Images in CTFd’s docker-compose

```
→ - trivy image redis:4
2023-05-02T14:04:18.428-0400 [INFO] Need to update DB
2023-05-02T14:04:18.429-0400 [INFO] DB Repository: ghcr.io/aquasecurity/trivy-db
2023-05-02T14:04:18.429-0400 [INFO] Downloading DB...
36.57 MiB / 36.57 MiB [-----] 100.00% 10.26
2023-05-02T14:04:23.007-0400 [INFO] Vulnerability scanning is enabled
2023-05-02T14:04:23.007-0400 [INFO] Secret scanning is enabled
2023-05-02T14:04:23.007-0400 [INFO] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2023-05-02T14:04:23.007-0400 [INFO] Please see also https://aquasecurity.github.io/trivy/v0.41/docs/secret/scanning/#recommendation for faster secret detection
2023-05-02T14:04:24.985-0400 [INFO] Detected OS: debian
2023-05-02T14:04:24.985-0400 [INFO] Detecting Debian vulnerabilities...
2023-05-02T14:04:24.997-0400 [INFO] Number of language-specific files: 0

redis:4 (debian 10.3)

Total: 188 (UNKNOWN: 5, LOW: 83, MEDIUM: 28, HIGH: 57, CRITICAL: 15)
```

Example – Hardening Runtime

```
deployments > ! path3.yaml
1  version: '3'
2  name: path3
3  services:
4    # Gateway service
5    refresh:
6      hostname: refresh
7      build: challenges/refresh
8      restart: unless-stopped
9      user: user
10     cap_drop:
11       - ALL
12     volumes:
13       - .flags/refresh:/flags
14     networks:
15       - public
16       - private
17
18   # Hidden service
19   dejavu:
20     hostname: dejavu
21     build: challenges/dejavu
22     restart: unless-stopped
23     user: user
24     cap_drop:
25       - ALL
26     volumes:
27       - .flags/dejavu:/flags
28       - .keys:/home/user/.ssh:ro
29     extra_hosts:
30       - "disgusting-bovine.battle:192.168.129.200"
31       - "poke-it.battle:192.168.129.201"
32     # Only accessible from the first service
33     networks:
34       vagrant:
35         private:
36           aliases:
37             - dejavu.${TLD}
```

Drop Extraneous Capabilities

Non-root user

Unique flag directory Per challenge

Read only mounts to prevent challenge tampering

Implementing **Controls** Throughout the Competition



OS Scanning & Control Verification!

Examples:

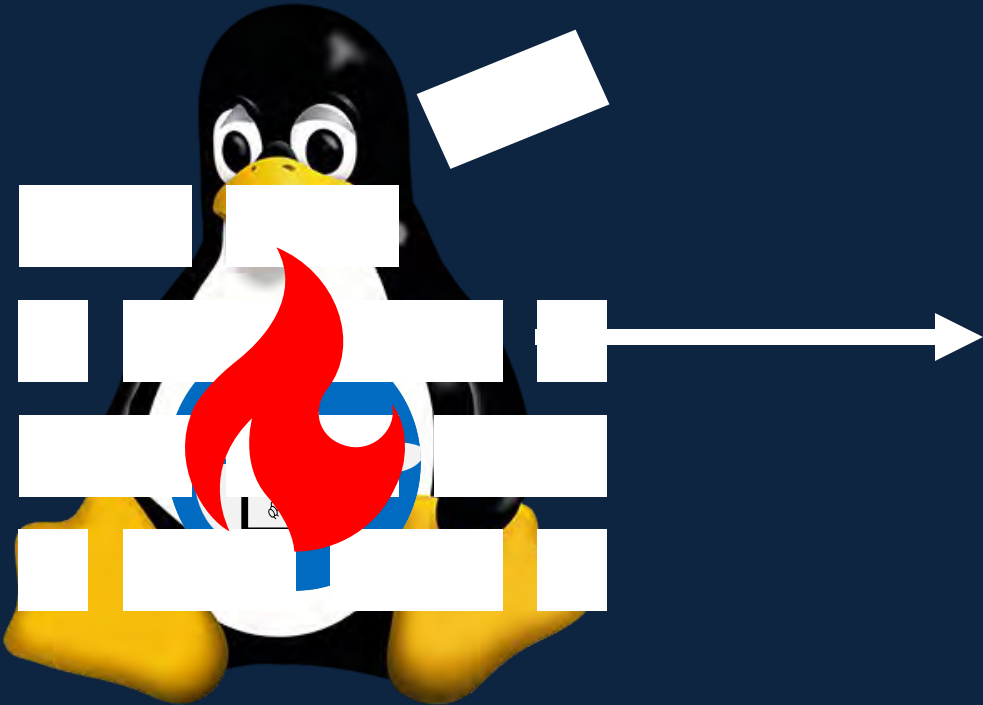
- **LinPeas**, Open-Source reconnaissance script that audits common vulnerabilities/suggests breakouts.
- **OpenScap**: Open-Source vulnerability scanner with reasonable defaults!

Example – Privilege Escalation Opportunities in Images

```
Container & breakout enumeration
https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-breakout
Container ID ..... 16d7cf7556a8 || Seccomp enabled? .....
AppArmor profile? ..... system_u:system_r:container_t:s0:c233,c423 || U:
0      1000      1
1      524288   65536
Vulnerable to CVE-2019-5021 .... No

Breakout via mounts
https://book.hacktricks.xyz/linux-hardening/privilege-escalation/docker-breakout,
sitive-mounts
/proc mounted? ..... No
/dev mounted? ..... No
Run ushare ..... Yes
release_agent breakout 1..... No
release_agent breakout 2..... No
core_pattern breakout ..... No
binfmt_misc breakout ..... No
uevent_helper breakout ..... No
is modprobe present ..... No
DoS via panic_on_oom ..... No
DoS via panic_sys_fs ..... No
DoS via sysreq_trigger_dos ..... No
/proc/config.gz readable ..... No
/proc/sched_debug readable ..... No
/proc/*/mountinfo readable ..... Yes
/sys/kernel/security present ... Yes
/sys/kernel/security writable .. No
```

Implementing **Controls** Throughout the Competition

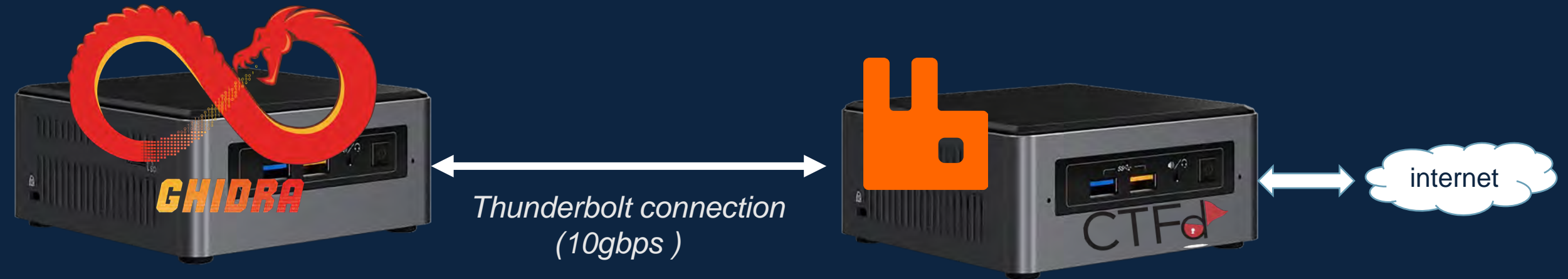


Isolation & “Zero Trust”

Examples:

- **Cloudflare Tunnels** – enable remote access behind unknown conference network
- **Isolated high-risk containers on host (Ghidra Runner) without direction internet connection**

Isolated Host– Container Overlay Networks



Example – Github Accounts for SSO



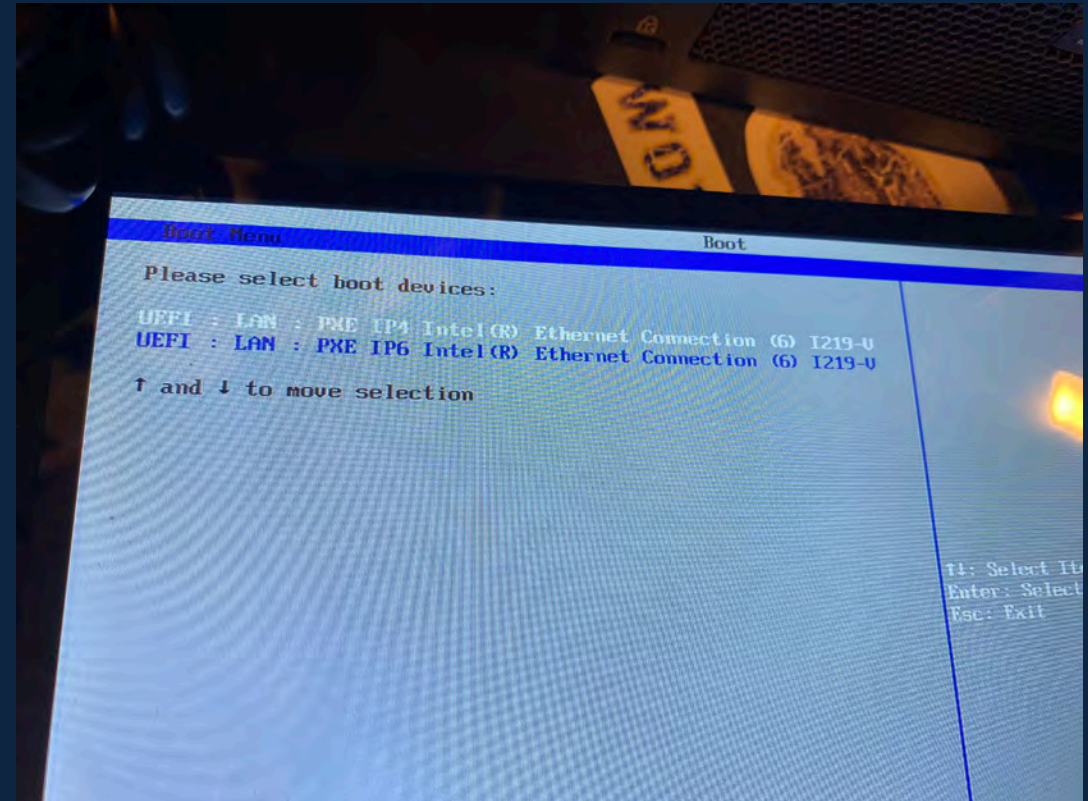
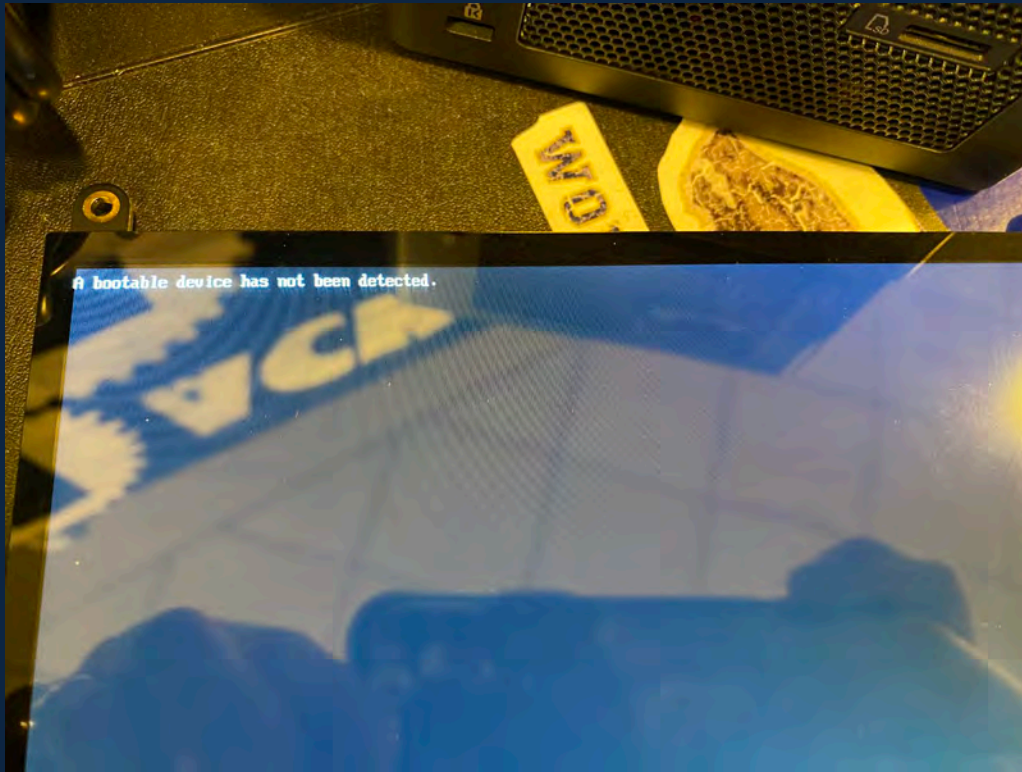
Clear **Communication &** Responsibilities



Identify **who** is responsible for each layer



Identify **communication platform** for incidents



Have a **plan for failure**

Interested in Learning More?

- <https://ghidra.golf>
- <https://battleofthebots.net>

- <https://github.com/ghidragolf>
- <https://github.com/battleofthebots>

- Code available on Github!

