Android Debug Bridge: High-Interaction Honeypot

Presented by Dea Llazo, Tsvetomir Hristov,
Velyan Kolev, and Vissarion Moutafis

Under the supervision of Harm Griffioen
12/04/2024

Content Overview

I

ADB

Android Debug Bridge
and Security Concerns

II

Honeypots

What are they?
Different Types
Why we use them

III

Our Solution

High-Interaction
ADB Honeypot

IV

Demo

Short recorded demo
Live demo

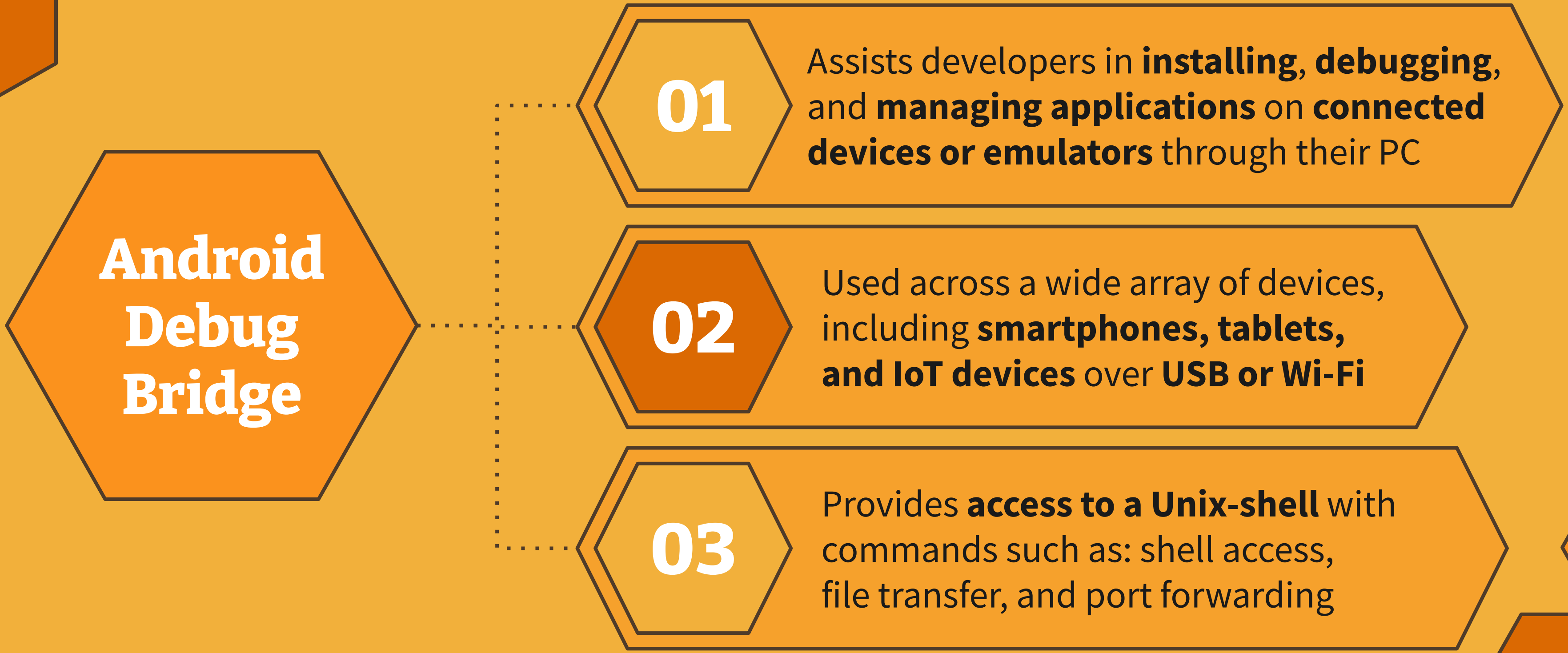
V

Discussion

Limitations and
Improvements

I

ADB: What is it?



Android Debug Bridge

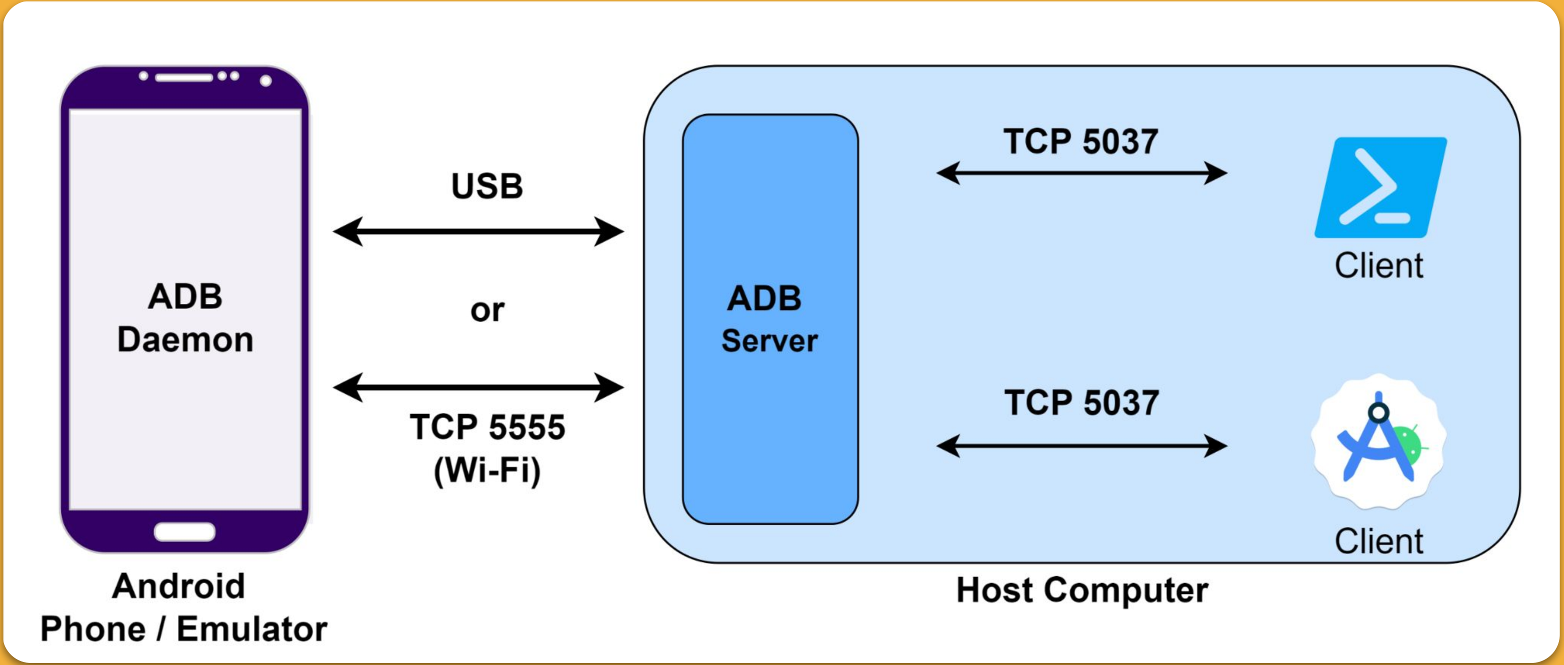
01 Assists developers in **installing**, **debugging**, and **managing applications** on **connected devices or emulators** through their PC

02 Used across a wide array of devices, including **smartphones, tablets, and IoT devices** over **USB or Wi-Fi**

03 Provides **access to a Unix-shell** with commands such as: shell access, file transfer, and port forwarding

I

ADB: How does it work?



https://emteria.com/learn/android-debug-bridge

ADB: Security Risks

01 Malicious actors can set up their **ADB servers to scan for vulnerable devices** and gain unauthorized access.

02 Android devices with **enabled debugging** and **connected to the same Wi-Fi network** as ADB server are at risk

03 **Known attacks** include MitM attacks[1], ADB.miner malware[2], Satori Variant[3]

04 **Possible security risks** include unauthorized access, malicious app installations, and data leakage

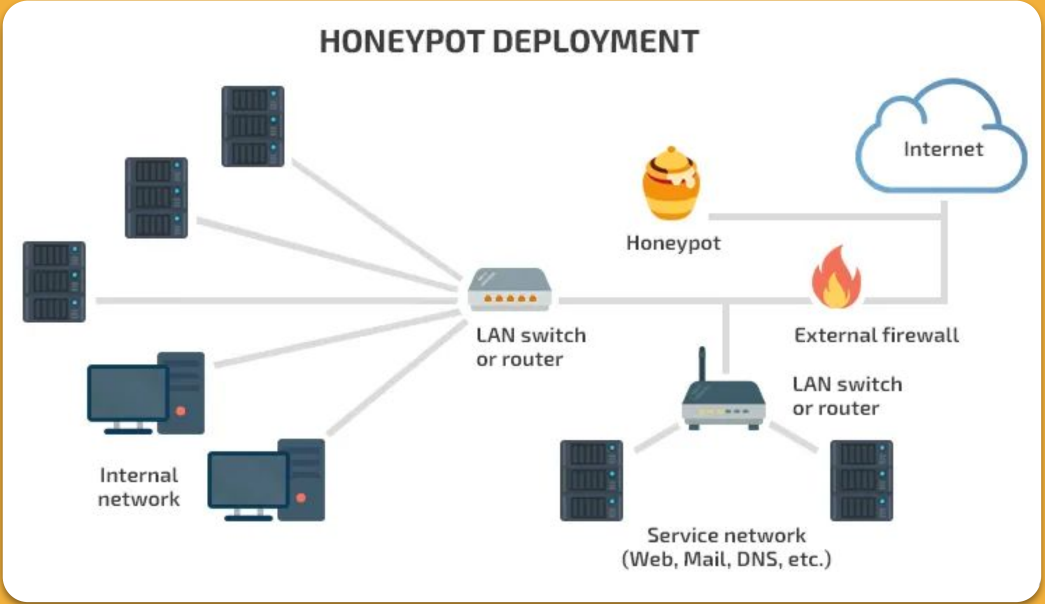
II

Project Objective

Create a **realistic yet controlled environment** for observing and **learning from real-life attack strategies** on ADB

Our Solution...
A Honeypot

- A **controlled, secure environment** designed to be intentionally vulnerable
- Acts as **bait for malicious actors**, to monitor and analyze their tactics
- Provides **insights into attack patterns**, enabling proactive security enhancements



https://www.apriorit.com/dev-blog/619-web-cybersecurity-honeypots-in-kubernetes

II

Types of Honeypots

Low-Interaction

Emulates specific services
or parts of protocols,
aiming at low surface attacks

Safe, low risk of harm,
limited resources needed

Limited engagement, less realistic
view on the attack surface

VS.

High-Interaction

Simulates a fully functional
environment that closely mirrors a
legitimate system

Enabling extensive interaction,
attracting sophisticated attacks

More complex to manage and
ensure safety

III

Our Solution: Goal

Expose several Android machines to any outside traffic **and capture requests** made to them through Android Debug Bridge

Our Solution: Overview

01 Virtual Machine

Sandbox for running services

Additional layer of separation

Easily deploy infrastructure on external servers

02 IPS

Capture all traffic

Capable of blocking certain traffic

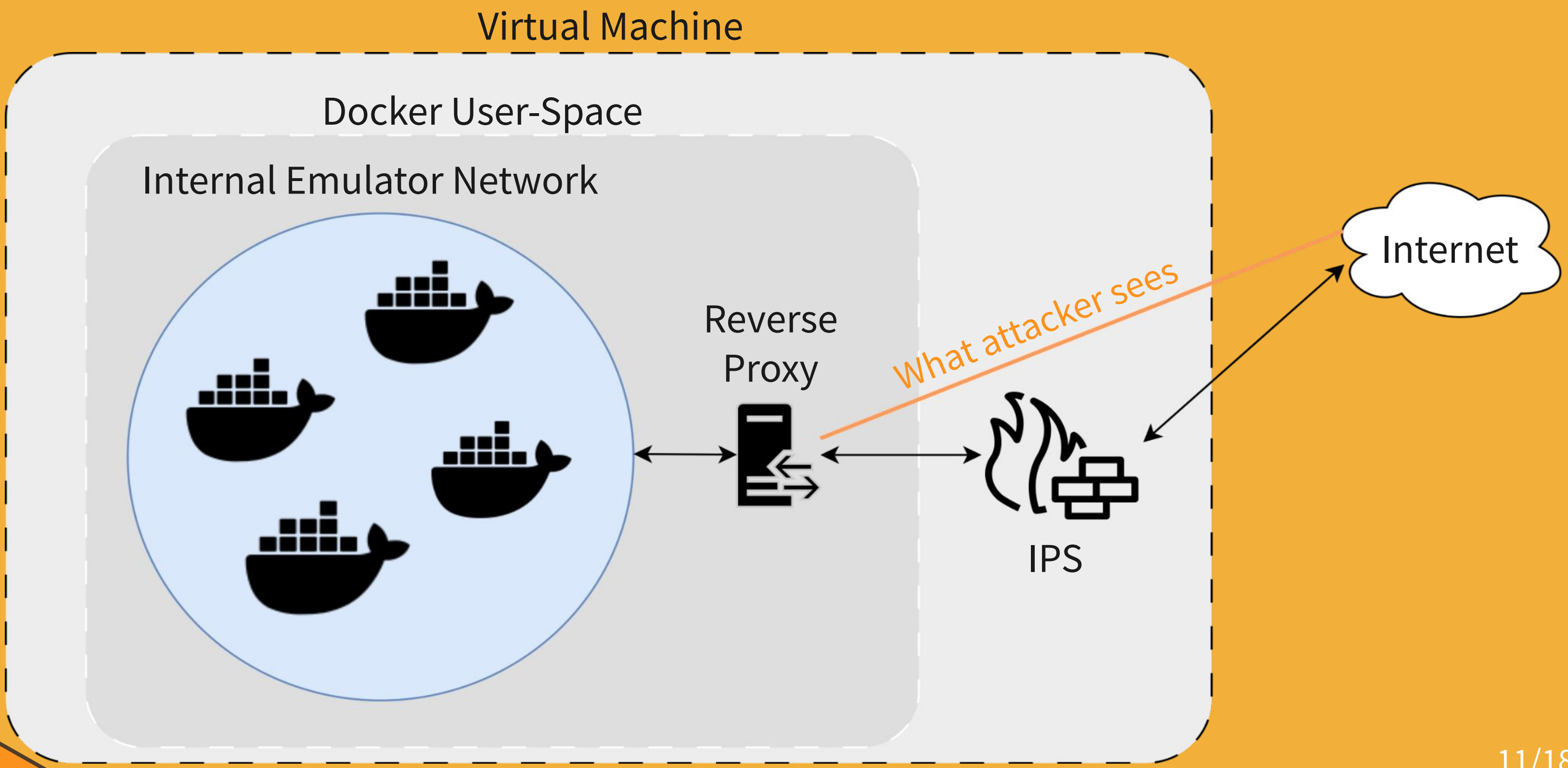
03 Reverse Proxy

Forward ADB traffic

04 Emulator

Running Android phone(s) which have ADB

Our Solution: Overview



III

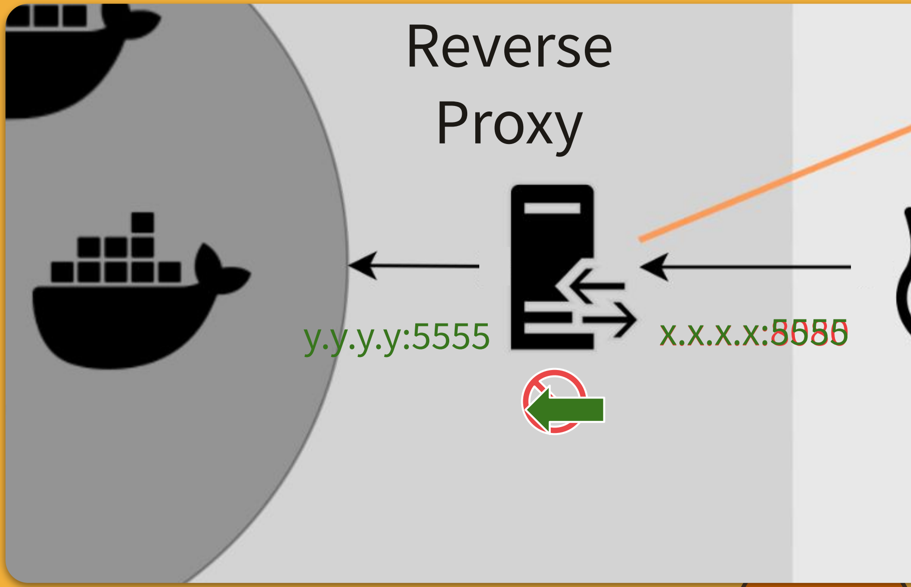
Our Solution: Android Emulators

- Why?
  - Simulate ADB’s interaction and commands so that
  - Attackers are fooled into connecting with it
- How?
  - Docker containers using official Google images of Android Devices
  - Trivial solution of a working protocol
  - Can be scaled by spawning more containers
- Ethical Concern
  - Problem: Running machines made to be compromised whose resources can be used for malicious purposes
  - Solution: Block all outgoing traffic by containing the emulators in an internal network
  - Consequence: Emulators can only be accessed by other hosts in the internal network

> *“Best way to emulate a protocol is to run the protocol” - No one ever (but it sounds cool)*

Our Solution: Reverse Proxy

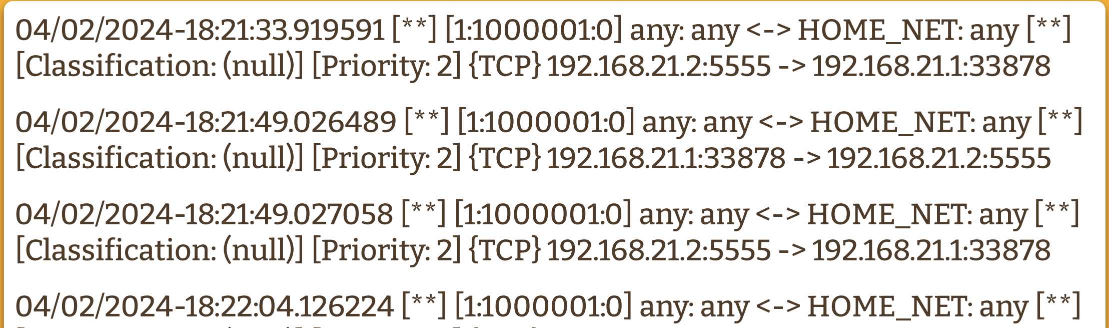
- Why?
  - Forward traffic on port 5555 (default ADB port) to Emulators
- How?
  - NGINX Docker container
  - Running on both the internal and external network
  - Forwarding all traffic destined to port 5555 to one of the Emulators
  - Same source
    = same assigned container
    = persistent changes
    = consistent view for attackers



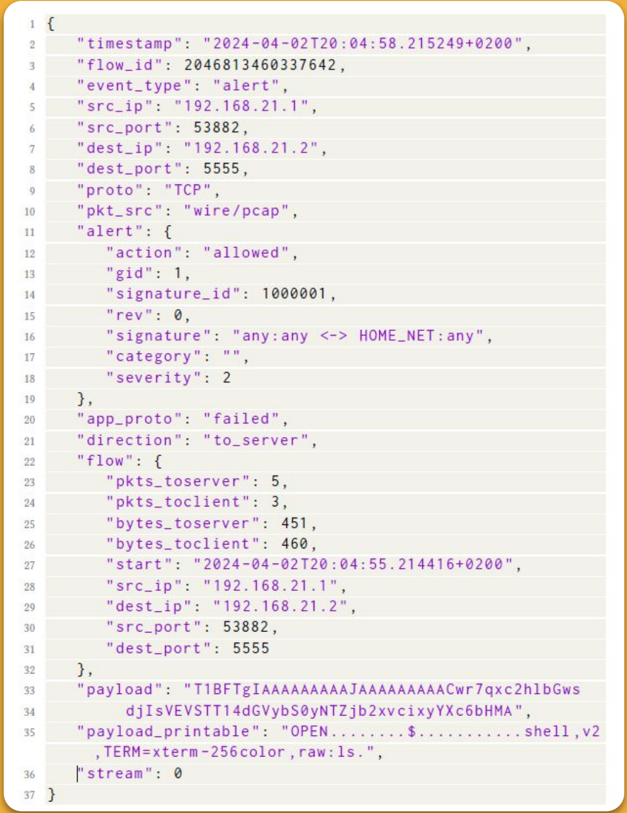
III

Our Solution: IPS

- Why?
  - Monitor all traffic related to Emulators
  - In the future, block certain traffic if it relates to undesired attacks
- What?
  - Suricata configured in IPS mode
  - Capture all incoming and outgoing traffic in network
  - Provides several types of logs

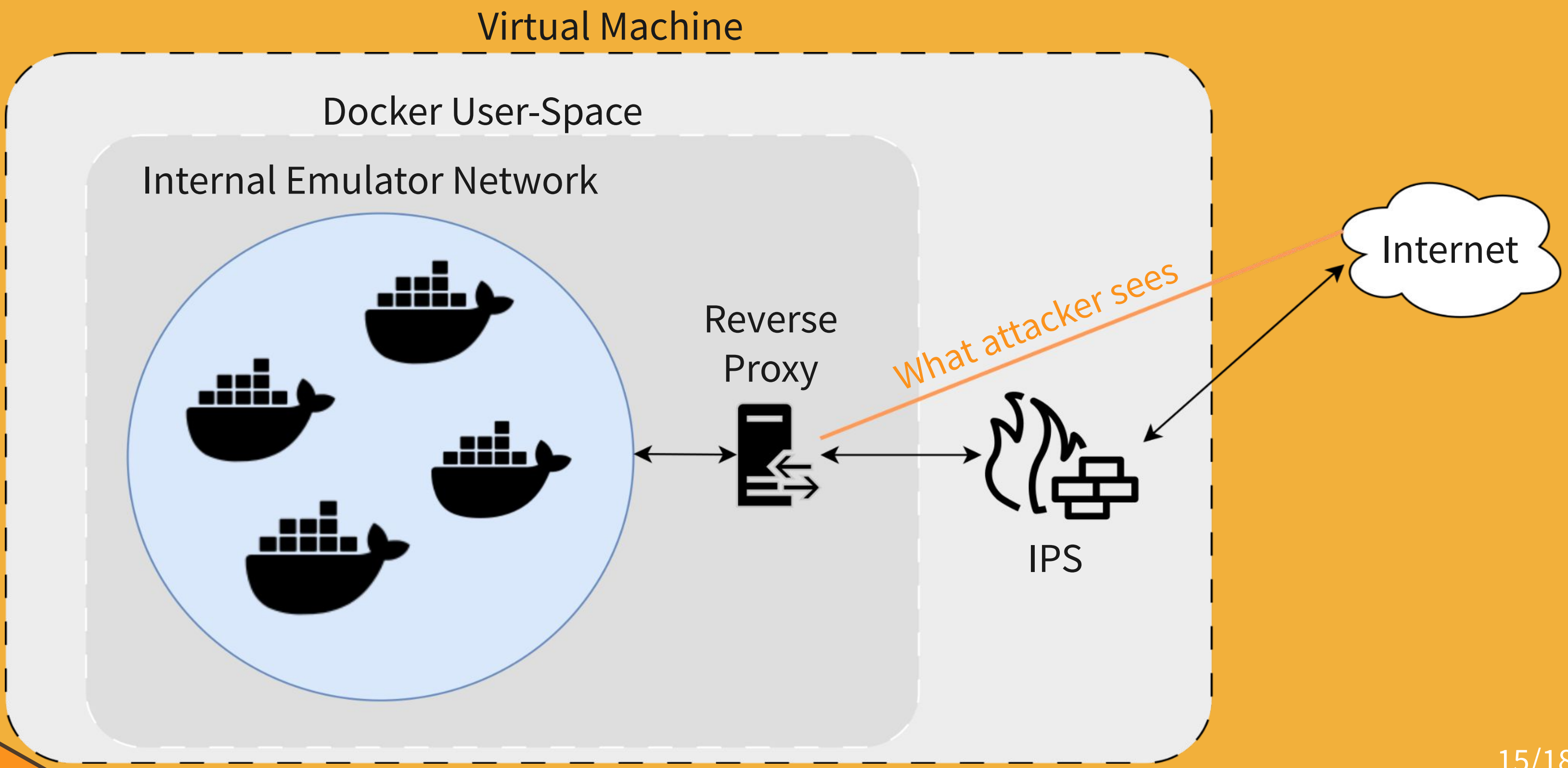


```
04/02/2024-18:21:33.919591 [**] [1:1000001:0] any: any <-> HOME_NET: any [**]
[Classification: (null)] [Priority: 2] {TCP} 192.168.21.2:5555 -> 192.168.21.1:33878

04/02/2024-18:21:49.026489 [**] [1:1000001:0] any: any <-> HOME_NET: any [**]
[Classification: (null)] [Priority: 2] {TCP} 192.168.21.1:33878 -> 192.168.21.2:5555

04/02/2024-18:21:49.027058 [**] [1:1000001:0] any: any <-> HOME_NET: any [**]
[Classification: (null)] [Priority: 2] {TCP} 192.168.21.2:5555 -> 192.168.21.1:33878

04/02/2024-18:22:04.126224 [**] [1:1000001:0] any: any <-> HOME_NET: any [**]
```

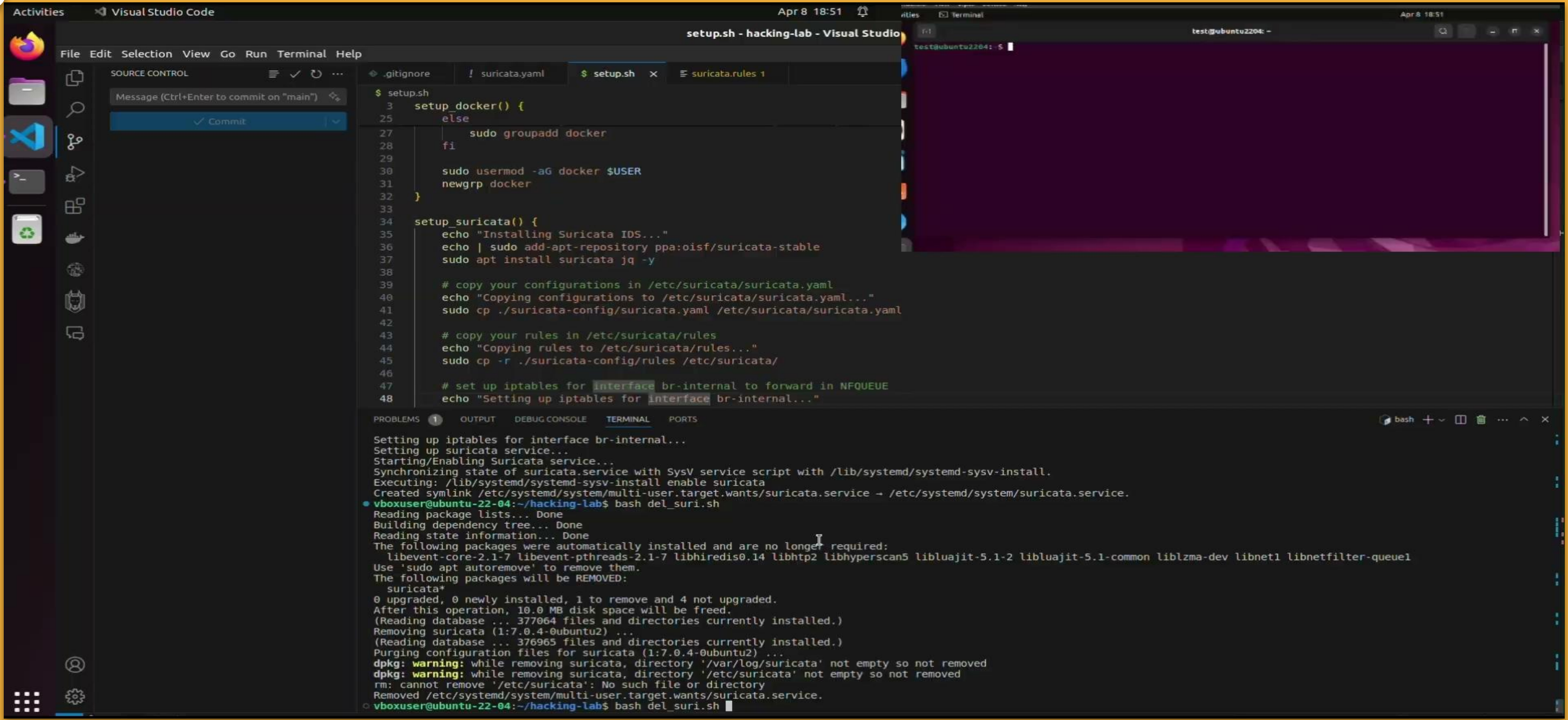


```
{
   "timestamp": "2024-04-02T20:04:58.215249+0200",
   "flow_id": 2046813460337642,
   "event_type": "alert",
   "src_ip": "192.168.21.1",
   "src_port": 53882,
   "dest_ip": "192.168.21.2",
   "dest_port": 5555,
   "proto": "TCP",
   "pkt_src": "wire/pcap",
   "alert": {
      "action": "allowed",
      "gid": 1,
      "signature_id": 1000001,
      "rev": 0,
      "signature": "any:any <-> HOME_NET:any",
      "category": "",
      "severity": 2
   },
   "app_proto": "failed",
   "direction": "to_server",
   "flow": {
      "pkts_toserver": 5,
      "pkts_toclient": 3,
      "bytes_toserver": 451,
      "bytes_toclient": 460,
      "start": "2024-04-02T20:04:55.214416+0200",
      "src_ip": "192.168.21.1",
      "dest_ip": "192.168.21.2",
      "src_port": 53882,
      "dest_port": 5555
   },
   "payload": "T1BFTgIAAAAAAAAAJAAAAAAAAACwr7qxc2hlbGws
        djIsVEVSTT14dGVybS0yNTZjb2xvcixyYXc6bHMA",
   "payload_printable": "OPEN........$..........shell,v2
     ,TERM=xterm-256color,raw:ls.",
   "stream": 0
}
```

Our Solution: Overview



Set-Up: 2 ubuntu-laptops (honeypot and *attacker*) on the same network (Hotspot)



```
$ setup.sh
setup_docker() {
    else
        sudo groupadd docker
    fi

    sudo usermod -aG docker $USER
    newgrp docker
}

setup_suricata() {
    echo "Installing Suricata IDS..."
    echo | sudo add-apt-repository ppa:oisf/suricata-stable
    sudo apt install suricata jq -y

    # copy your configurations in /etc/suricata/suricata.yaml
    echo "Copying configurations to /etc/suricata/suricata.yaml..."
    sudo cp ./suricata-config/suricata.yaml /etc/suricata/suricata.yaml

    # copy your rules in /etc/suricata/rules
    echo "Copying rules to /etc/suricata/rules..."
    sudo cp -r ./suricata-config/rules /etc/suricata/

    # set up iptables for interface br-internal to forward in NFQUEUE
    echo "Setting up iptables for interface br-internal..."
```

```
Setting up iptables for interface br-internal...
Setting up suricata service...
Starting/Enabling Suricata service...
Synchronizing state of suricata.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
Created symlink /etc/systemd/system/multi-user.target.wants/suricata.service → /etc/systemd/system/suricata.service.
vboxuser@ubuntu-22-04:~/hacking-lab$ bash del_suri.sh
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libevent-core-2.1-7 libevent-pthreads-2.1-7 libhiredis0.14 libhtp2 libhyperscan5 libluajit-5.1-2 libluajit-5.1-common liblzma-dev libnet1 libnetfilter-queue1
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  suricata*
0 upgraded, 0 newly installed, 1 to remove and 4 not upgraded.
After this operation, 10.0 MB disk space will be freed.
(Reading database ... 377064 files and directories currently installed.)
Removing suricata (1:7.0.4-0ubuntu2) ...
(Reading database ... 376965 files and directories currently installed.)
Purging configuration files for suricata (1:7.0.4-0ubuntu2) ...
dpkg: warning: while removing suricata, directory '/var/log/suricata' not empty so not removed
dpkg: warning: while removing suricata, directory '/etc/suricata' not empty so not removed
rm: cannot remove '/etc/suricata': No such file or directory
Removed /etc/systemd/system/multi-user.target.wants/suricata.service.
vboxuser@ubuntu-22-04:~/hacking-lab$ bash del_suri.sh
```

Discussion

Our Solution

- **Flexible setup** incl. changeable service
- **Several stateful emulators** for advanced attacks
- **Full ADB functionality** alongside isolation
- **Configurable monitoring**

Limitations

- All **outgoing traffic** is **blocked** (no internet access)
- ADB ‘shell’ commands logs are **hard to reconstruct**
- IPS configuration does **not monitor internal emulator activity**

Improvements

- Configure IPS rules to **allow legitimate outbound traffic**
- Implement **in-device logging** for non-network activities
- Variable number of emulators spawned for **better load balancing**

Questions?

Thank You!

REFERENCES

01

MitM attacks
https://www.sciencedirect.com/science/article/pii/S016740481831023X

02

ADB.miner malware
https://www.radware.com/security/ddos-threats-attacks/threat-advisories-attack-reports/adb-miner/

03

Satori Variant
https://wccftech.com/attackers-exploit-adb-satori-botnet/

04

Slidesgo Slides Template
http://bit.ly/2PfT4lq