2024-2025

ELEC-H504 - Network Security

Deception & Honeypot for Attack Profiling

SUNDARESAN Sankara
CHOUGULE Gaurav
MESSAOUDI Leila
BOTTON David

Pr. Jean-Michel Dricot
Navid Ladner

June 2025

Contents

ABSTRACT

This paper shows an operational deployment of the `SSH` honeypot using `cowrie` on an Ubuntu `EC2` instance to capture attacker activity in real-world circumstances. By exposing a knowingly open SSH port on the Internet and securing legitimate access with a cryptographic key on a different port, the study observes and inspects adversary tactics, techniques, and procedures (TTPs). Key steps include isolating the honeypot space from production access using `fail2ban`, redirecting malicious traffic to `cowrie` via `iptables`, and customizing fake files to track attacker activity. A walkthrough of the settings is covered to demonstrate a complete implementation in `cowrie`, as this paper remains focused on the hands-on aspect of honeypot-based deception.

1 Introduction

With more advanced automated SSH-based attacks, empirical analysis of attacker processes has been crucial in strengthening defenses. This report describes the configuration of a cowrie honeypot in a production AWS environment that was designed to mimic realistic infrastructure but isolate malicious activity from legitimate administrative activity. Through routing *port 22* traffic to the honeypot and restricting production access to *port 2223* with key-based authentication, the study enables monitoring of attacker activities in fine granularity without compromising the security of systems. The fail2ban program is employed for counteracting brute-force attacks against hardened administrative interfaces, while rules based on iptables are implemented to route opponents into the honeypot infrastructure. This project's public git repository is available at that location.

1.1 Problem Statement

Public SSH services are most frequent targets of credential-stuffing attacks and post-compromise persistence methods such as SSH key injection and cronjob exploitation. Existing defensive measures have a tendency to lack the visibility to deeply analyze these malicious processes. This research addresses two significant challenges: (1) safe isolation of production access from honeypot bait to prevent collateral system compromise, and (2) the engineering of logging mechanisms that are able to capture attacker TTPs without affecting environmental integrity.

1.2 Research Questions

The study investigates three core questions: First, which credential pairs are most frequently exploited in SSH brute-force campaigns against internet-exposed systems? Second, what persistence mechanisms (e.g., unauthorized key additions, scheduled task manipulation) do attackers prioritize following initial compromise? Third, to what extent can strategically placed decoy files, such as fabricated /etc/shadow entries, extend attacker engagement to improve TTP profiling?

Security Disclaimer

This setup only serves academic purposes; adversarial activities are welcomed, but no offensive counteraction shall be taken in return. Additionally, cloud provider configurations (e.g., IAM, networking) are not included as implementation-specific and beyond the scope of this paper.

2 SSH Isolation & Fail2ban Hardening

2.1 Admin SSH Hardening

Restricts access to key-based auth on non-standard *port 2223* for legitimate contributors to the project, thus reducing attack surface.

```
# /etc/ssh/sshd_config
Port 2223
Protocol 2
HostKeyAlgorithms ssh-ed25519,rsa-sha2-512
KexAlgorithms curve25519-sha256
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com
```

```
MACs hmac-sha2-512-etm@openssh.com
PermitRootLogin no
PasswordAuthentication no
AllowUsers ubuntu
LoginGraceTime 30s
MaxAuthTries 2
PubkeyAuthentication yes
X11Forwarding no
```

Listing 1: Securing Legitimate Access

2.2 Fail2ban Configuration

Targets repeated public key failures, common in credential stuffing. Ignores password attempts (disabled in SSH).

```
 # /etc/fail2ban/jail.d/ssh-admin.conf
 [ssh-admin]
enabled  = true
port     = 2223
filter   = sshd
maxretry = 3
findtime = 10m
bantime  = 30m
```

Listing 2: Custom Jail Rules

```
 # /etc/fail2ban/filter.d/sshd.conf
failregex = %(cmnfailre)s
            <mdre-<mode>>
            ^%(__prefix_line)s(?: Received disconnect from <HOST> port \d+: Too many
    ↪ authentication failures | Disconnected from <HOST> port \d+ due to: Authentication
    ↪ failed for .* publickey )
            %(cfooterre)s
```

Listing 3: Regex Filter Against Key-Based Attacks

2.3 IPtables Redirection

Redirects all *port 22*'s traffic to `cowrie`, with rules persisting after reboot. Legitimate key-based accesses on *port 2223* are separated and preserved, we could additionally whitelist our group collaborators' IP addresses for a much tighter defense but will not do it here for ease of development. The `sshd` to `cowrie` (ports *22* → *2222*) redirection is a security design with multiple benefits: we avoid both processes to conflict with each other, we can restart them separately, binding to high ports does not require root privileges, also, `fail2ban` needs an explicit target to be effective. Compartmentalization is a key principle for any deceptive operation, we also ensure an appropriate foundation for clean and comprehensive post-attack analysis.

```
sudo iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
sudo ip6tables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
sudo apt install \texttt{iptables}-persistent netfilter-persistent
sudo netfilter-persistent save
```

Listing 4: Traffic Redirection to Cowrie

2.4 Validation Metrics

To ensure rigorous validation, we verify component functionality and isolation using cybersecurity tools. The following Annex provides exact commands for: Network isolation (nmap, `iptables`), `fail2ban` regex precision (ssh-key brute-forcing), `SSH` configuration enforcement and resilience regarding downgrade attacks, protocol validation and cryptographic compliance.

TODO: Validate that these commands work once cowrie is running on 2222

A Annex: Validation for SSH Isolation & Fail2ban Hardening

```
# Verify port 22 redirects to Cowrie (2222) and admin port (2223) is exclusive
sudo nmap -sV -Pn -p 22,2222,2223 $EC2_PUBLIC_IP

# Check iptables NAT rules for redirect (should show 22 to 2222)
sudo iptables -t nat -L PREROUTING -v -n

# Ensure no SSH service binds to port 22 (only Cowrie on 2222)
sudo ss -tulpn | grep -E ':22|:2222|:2223'
```

Listing 5: Network Isolation Verification

TODO: screenshot here

```
# Simulate key-based brute-forcing to trigger fail2ban
for i in {1..5}; do ssh -i /path/fake_key ubuntu@localhost -p 2223; done

# Verify fail2ban logged the bans (look for 'ssh-admin' jail)
sudo grep "ssh-admin" /var/log/fail2ban.log

# Check active bans (should list test IP)
sudo fail2ban-client status ssh-admin
```

Listing 6: Fail2ban Efficacy Testing

TODO: screenshot here

```
# 1. Verify active SSH configuration matches hardening intent (no fallback to weak
    ↪ protocols)
sudo sshd -T | grep -E '^ciphers|^kexalgorithms|^macs|^hostkeyalgorithms'

# 2. Test SSH service for protocol/cipher negotiation weaknesses
nmap -Pn -p 2223 --script ssh2-enum-algos $EC2_PUBLIC_IP | grep -A 10 "algorithm
    ↪ negotiation"

# 3. Confirm password authentication is globally disabled (even if bypass attempted)
ssh -o PubkeyAuthentication=no -o PreferredAuthentications=password ubuntu@$EC2_PUBLIC_IP
    ↪ -p 2223
```

Listing 7: SSH Service Hardening Validation

TODO: screenshot here

B Annex: LLM Usage in this Project

TODO: that annex