AI Honeypot

Live Attack Demonstration Guide

Step-by-Step Demo of All 7 OWASP Attacks

PRE-DEMO SETUP

1. **Start the server**

```
python app.py
```

2. **Open the Demo Controller (Primary Method)**

```
http://localhost:8000/controller
```

For a smooth demo, use this controller to execute attacks with one click.

3. **Open Dashboard (Can do from Controller)**

```
http://localhost:8000/demo
```

4. **Position windows side-by-side**

Left: Demo Controller | Right: Live Dashboard showing real-time updates

5. **Have this PDF ready as a script/backup**

DEMO FLOW (7-10 minutes)

You will demonstrate 7 attack types in sequence.

Primary Method: Click buttons on the Demo Controller (Recommended)

Backup Method: Copy-paste URLs from this PDF if controller fails

For each attack:

• Click attack button (or navigate to URL)

• Point out the detected attack on the dashboard

• Highlight the new **LLM Reasoning** steps

• Show the **Attacker Profile** updating

• Explain the **Threat Intelligence** score

ATTACK 1: SQL INJECTION

Step 1: Execute the attack

```
http://localhost:8000/search?q=' OR 1=1--
```

Step 2: What to say to judges

"This is a classic SQL injection attack. The payload ' OR 1=1-- attempts to bypass authentication by injecting SQL logic. Notice the single quote, OR condition, and comment marker."

Step 3: Point out dashboard changes

✓ ***LLM Reasoning:*** *Shows step-by-step analysis of the payload*

✓ ***Attacker Profile:*** *Shows 'NOVICE' skill level and 'SQLMap' tool detection*

✓ ***Threat Intelligence:*** *Updates threat score (likely >60)*

✓ ***Intelligence Analysis:*** *Maps to MITRE T1190*

Step 4: Highlight the LLM Analysis

"Notice the 'LLM Reasoning Process' panel. It's not just logging the attack; it's explaining WHY it's an attack: 'Detected SQL metacharacters', 'Analyzing intent', and 'Recommended response'."

Step 5: Show the response

"The honeypot returns fake database results to keep the attacker engaged. Notice it looks realistic - fake usernames, emails, passwords."

ATTACK 2: CROSS-SITE SCRIPTING (XSS)

Step 1: Execute the attack

```
http://localhost:8000/search?q=<script>alert('XSS')</script>
```

Step 2: What to say to judges

"This is a reflected XSS attack. The attacker injects JavaScript that would execute in the victim's browser. In a real application, this could steal cookies or hijack sessions."

Step 3: Point out dashboard changes

✓ *Attack type changes to 'XSS'*

✓ *Attack counter: 2*

✓ *Behavioral analysis: Attacker using multiple vectors*

✓ *Skill level may upgrade to INTERMEDIATE*

Step 4: Show attack sequence

"Notice the timeline now shows: SQL Injection → XSS. The system is tracking the attack progression and building an attacker profile."

ATTACK 3: PATH TRAVERSAL

Step 1: Execute the attack

```
http://localhost:8000/search?q=../../etc/passwd
```

Step 2: What to say to judges

"This is a path traversal attack trying to access /etc/passwd. The ../ sequences attempt to navigate up the directory tree to read sensitive files."

Step 3: Point out dashboard changes

✓ *Attack type: 'PATH_TRAVERSAL'*

✓ *Attack counter: 3*

✓ *Attack stage may escalate to PRIVILEGE_ESCALATION*

✓ *Threat level increases (MEDIUM → HIGH)*

Step 4: Show the fake response

"The honeypot returns a realistic-looking /etc/passwd file with fake user accounts. This keeps the attacker engaged while we study their techniques."

ATTACK 4: COMMAND INJECTION

Step 1: Execute the attack

```
http://localhost:8000/search?q=test;ls -la
```

Step 2: What to say to judges

"This is OS command injection. The semicolon terminates the intended command and executes 'ls -la' to list directory contents. This could lead to full system compromise."

Step 3: Point out dashboard changes

✓ *Attack type: 'CMD_INJECTION'*

✓ *Attack counter: 4*

✓ *MITRE technique: T1059 (Command and Scripting Interpreter)*

✓ *Threat level: HIGH (attacker attempting system access)*

Step 4: Show AI prediction update

"After 4 attacks, our ML model has learned this attacker's pattern. It now predicts they'll likely attempt data exfiltration or admin access next."

ATTACK 5: SERVER-SIDE REQUEST FORGERY (SSRF)

Step 1: Execute the attack

```
http://localhost:8000/search?q=http://localhost/admin
```

Step 2: What to say to judges

"This is SSRF - the attacker tries to make our server request internal resources. They're targeting http://localhost/admin to access internal admin panels."

Step 3: Point out dashboard changes

✓ *Attack type: 'SSRF'*

✓ *Attack counter: 5*

✓ *Attack stage: PRIVILEGE_ESCALATION*

✓ *Skill level may upgrade to ADVANCED*

Step 4: Highlight the sophistication

"SSRF is a more advanced attack. The system recognizes this and upgrades the attacker's skill level. This affects our threat assessment and response strategy."

ATTACK 6: AUTHENTICATION BYPASS

Step 1: Execute the attack

```
http://localhost:8000/admin?user=admin&pass;=admin:admin
```

Step 2: What to say to judges

"This is a credential stuffing attack using default credentials admin:admin. Attackers often try common username/password combinations."

Step 3: Point out dashboard changes

✓ *Attack type: 'Authentication Bypass'*

✓ *Attack counter: 6*

✓ *Endpoint: /admin (high-value target)*

✓ *MITRE technique: T1078 (Valid Accounts)*

✓ *Threat level: HIGH or CRITICAL*

Step 4: Show the complete attack chain

"Look at the timeline: SQL Injection → XSS → Path Traversal → Command Injection → SSRF → Auth Bypass. This is a sophisticated multi-stage attack. Our system has mapped it to MITRE ATT&CK; tactics."

ATTACK 7: INSECURE DESERIALIZATION

Step 1: Execute the attack

```
http://localhost:8000/search?q=__reduce__
```

Step 2: What to say to judges

"This targets Python's pickle deserialization. The __reduce__ method can execute arbitrary code during unpickling, leading to remote code execution."

Step 3: Point out dashboard changes

✓ *Attack type: 'Insecure Deserialization'*

✓ *Attack counter: 7*

✓ *All 7 OWASP attack types demonstrated!*

✓ *Threat level: CRITICAL (RCE attempt)*

Step 4: Show the complete picture

"We've now demonstrated all 7 OWASP vulnerabilities our honeypot detects. The dashboard shows the complete attack timeline, MITRE mapping, threat level, and AI predictions."

PART 2: POST-EXPLOITATION ANALYSIS (USING ATTACKER ID)

By this point, we have successfully tracked the attacker's unique ID across the session. Now we use this ID to generate deep forensic insights:

Feature 1: AI Prediction (Probabilistic Modeling)

Click Button: ■ AI Prediction

"Using the captured Attacker ID, our Markov chain model analyzes their specific sequence. It predicts the next likely attack vector (e.g., 'Admin Access') with calculated probability."

Feature 2: MITRE ATT&CK; Matrix

Click Button: ■ MITRE Mapping

"We map this specific attacker's actions to the MITRE ATT&CK; framework. You can see the Tactics (e.g., Initial Access) and Techniques (e.g., T1190) used in this session."

Feature 3: Forensic Timeline

Click Button: ■ Forensic Timeline

"This generates a complete chronological timeline for this Attacker ID. Every payload, timestamp, and response is logged for forensic reconstruction."

Feature 4: Incident Playbook Generation

Click Button: ■ Incident Playbook

"Based on the dominant attack type (e.g., SQL Injection), the system auto-generates an actionable incident response playbook with detection rules and containment steps."

Feature 5: STIX 2.1 Threat Intel Export

Click Button: ■ STIX Export

"We package all intelligence about this Attacker ID into a standardized STIX 2.1 JSON bundle. This is ready for immediate sharing with other SOCs or import into a SIEM."

Feature 6: Canary Token Analytics

Click Button: ■■ Canary Analytics

"If the attacker accessed our honeytokens (fake credentials/files), this dashboard shows exactly where and how those tokens were used, revealing their post-compromise movement."

CLOSING THE DEMO

Summary Points

"Let me summarize what we just demonstrated:"

1. **Real-time detection** of 7 OWASP attack types with instant dashboard updates
2. **AI-powered prediction** using Markov chains to forecast next attacks
3. **Automatic MITRE mapping** to industry-standard ATT&CK; framework
4. **Forensic timeline** with complete attack reconstruction
5. **Auto-generated playbooks** for instant incident response
6. **STIX 2.1 export** for threat intelligence sharing
7. **Production-ready security** with rate limiting, signed cookies, and resource protection

Key Differentiators

"What makes this different from traditional honeypots:"

- **Proactive vs Reactive:** We predict attacks, not just log them
- **Automated Intelligence:** MITRE mapping and playbooks happen automatically
- **Real-time Analytics:** Sub-second dashboard updates via WebSockets
- **Production Ready:** Enterprise-grade security with all vulnerabilities fixed
- **Integration Ready:** STIX export, API access, SIEM compatibility

TROUBLESHOOTING

Dashboard not updating?

- Refresh the browser (F5)
- Check WebSocket connection in browser console
- Restart the server: Ctrl+C, then python app.py

Attack not detected?

- Verify the URL is correct (copy-paste from this PDF)
- Check server console for errors
- Ensure special characters are properly encoded

Server crashed?

- Don't panic! Restart with: python app.py
- Previous attacks are logged in attacks.json
- Continue demo from where you left off

QUICK REFERENCE - ALL ATTACK URLS

Attack Type	URL

SQL Injection	http://localhost:8000/search?q=' OR 1=1--
XSS	http://localhost:8000/search?q=<script>alert('XSS')</script>
Path Traversal	http://localhost:8000/search?q=../../etc/passwd
Command Injection	http://localhost:8000/search?q=test;ls -la
SSRF	http://localhost:8000/search?q=http://localhost/admin
Auth Bypass	http://localhost:8000/admin?user=admin&pass=admin:admin
Deserialization	http://localhost:8000/search?q=__reduce__

Pro Tip: Keep this PDF open during your demo. Copy-paste URLs directly to avoid typos!