

# 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."

## ADVANCED FEATURES DEMONSTRATION

### Feature 1: AI Prediction

#### Navigate to:

```
http://localhost:8000/api/prediction/[attacker_id]
```

"This shows our Markov chain predictions. Based on the attack sequence, it predicts the next likely attack with probability scores."

### Feature 2: MITRE ATT&CK; Mapping

#### Navigate to:

```
http://localhost:8000/api/mitre/[attacker_id]
```

"Every attack is automatically mapped to MITRE ATT&CK; techniques. This shows tactics, techniques, and even matches to known APT groups."

### Feature 3: Forensic Timeline

#### Navigate to:

```
http://localhost:8000/api/timeline/[attacker_id]
```

"This is a complete forensic timeline with timestamps, attack types, and payloads. We can even generate a replay script to recreate the attack."

### Feature 4: Incident Playbook

#### Navigate to:

```
http://localhost:8000/api/playbook/SQL%20Injection
```

"For each attack type, we auto-generate an incident response playbook. This includes detection rules, containment steps, and remediation actions."

### Feature 5: STIX Export

#### Navigate to:

```
http://localhost:8000/api/threat-intel/[attacker_id]/stix
```

"We export threat intelligence in STIX 2.1 format - the industry standard. This can be imported into any SIEM or shared with other organizations."

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

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