Incident Response Playbook: Web Application & Internet-Facing Attacks

Team AnubisX

Version 1.0
October 2025

Document Control

Attribute	Value
Version	1.0
Status	Draft / Operational
Owner	AnubisX Security Team
Review Cycle	Quarterly or after major incident
Approver	SOC Manager / Head of IR

Contents

1 Introduction **3**
1.1 Purpose . . . 3
1.2 Scope . . . 3

2 Overview of the Category **3**
2.1 Definition . . . 3
2.2 Common Attack Chain . . . 3
2.3 Primary Risks & Business Impact . . . 3

3 Severity Level Assessment & MTTD / MTTR **4**

4 Tools & Preparation (Recommended) **4**

5 Incident Response Phases **4**
5.1 Identification & Triage . . . 4
5.2 Containment (Immediate / Short-term) . . . 5
5.3 Investigation & Forensic Triage . . . 5
5.4 Eradication . . . 5
5.5 Recovery . . . 6
5.6 Post-Incident Activities . . . 6

6 MITRE ATT&CK Framework Mapping **6**

7 Key Telemetry & Logs to Collect **6**

8 Subcategory Scenarios (Realistic) **6**

9 Appendices **9**
9.1 Appendix A — Useful SIEM / Investigation Queries . . . 9
9.2 Appendix B — Forensic Artifact Locations . . . 10
9.3 Appendix C — Incident Report Template (Summary) . . . 10

1 Introduction

1.1 Purpose

This playbook provides operational procedures to detect, triage, contain, investigate, eradicate, and recover from incidents categorized as **Web Application & Internet-Facing Attacks**. It is intended for SOC analysts, application security engineers, incident responders, web operations, and leadership. The playbook focuses on exposures and attacks against internet-facing assets including web servers, application servers, APIs, and web-based management interfaces.

1.2 Scope

Applies to web applications (on-prem and cloud), APIs (REST/GraphQL), web servers (IIS, Apache, Nginx), WAFs, CDN, load balancers, reverse proxies, containerized web workloads, and supporting telemetry (web logs, WAF logs, application logs, database logs, SIEM, network captures).

2 Overview of the Category

2.1 Definition

Web Application & Internet-Facing Attacks covers adversary techniques that exploit vulnerabilities in web applications or their hosting stacks to achieve code execution, data access, persistence or service disruption. Common instances include web shell deployment, SQL injection (SQLi), cross-site scripting (XSS), insecure deserialization, authentication bypass, and exposed admin interfaces.

2.2 Common Attack Chain

1. **Reconnaissance:** automated scanning, directory enumeration, probing for vulnerable parameters/endpoints.
2. **Exploitation:** SQLi, file upload flaws, insecure deserialization, command injection, XSS.
3. **Post-Exploitation:** web shell upload, execution of commands, creation of persistence, lateral pivoting to backend systems.
4. **Data Access / Exfiltration:** database dumps, API data extraction, file retrieval from webroot or mounted volumes.
5. **Impact:** data breach, unauthorized access to internal systems, ransomware staging, or service disruption (DoS).

2.3 Primary Risks & Business Impact

- Exposure of sensitive customer or business data via databases or APIs.
- Unauthorized admin access leading to full application takeover.
- Supply-chain or downstream impact when backend systems are reachable.
- Regulatory, financial, and reputational damage following a breach.

3 Severity Level Assessment & MTTD / MTTR

Level	Description / Criteria	Example	MTTD	MTTR
Critical	Confirmed remote code execution or web shell on internet-facing critical application exposing PII/financial systems, or automated mass-exfiltration discovered.	Web shell on customer portal extracting DB backups.	$\leq$ 15 min	Contain within 4 hrs; recovery 24–72 hrs.
High	Confirmed SQL injection allowing data exfiltration from production DB or authenticated XSS against high-privilege users.	SQLi leading to sensitive table dump.	$\leq$ 1 hr	6–24 hrs.
Medium	Attempted exploits or suspicious payloads detected (probing, single successful injection point in staging).	Scanner detected payloads with occasional successful parameter tampering.	$\leq$ 2 hrs	12–48 hrs.
Low	Reconnaissance or single probes that are blocked by WAF/controls and show no exploitation.	Single probe to admin.php blocked by WAF.	$\leq$ 4 hrs	Monitor / minor remediation within 24–72 hrs.

Table 1: Severity Matrix - Web Application & Internet-Facing Attacks

4 Tools & Preparation (Recommended)

- **Web Application Firewall (WAF):** Managed or inline WAF with custom rules and logging (AWS WAF, Cloudflare, ModSecurity).
- **Runtime App Security:** RASP, WAF + EDR integration, application-layer logging and tracing.
- **SIEM / Log Aggregation:** Centralize web server logs, WAF logs, API gateway logs, DB logs and JVM/.NET logs.
- **Vulnerability Scanning / SCA:** Burp, ZAP, Snyk, and SCA for dependency/third-party vulnerabilities.
- **Forensics / IR Tools:** Web server access logs, database audit logs, packet captures at ingress, forensic copies of webroot and backups.
- **Development Controls:** Secure coding training, parameterized queries/ORMs, content security policy (CSP), input validation and output encoding.

5 Incident Response Phases

5.1 Identification & Triage

Signals/Detections:

- WAF alerts: blocked SQLi/XSS patterns, unusual POSTs with file uploads.

- Web server logs: anomalous POSTs to upload endpoints, long query strings, suspicious user agents.
- Database logs: unexpected queries (SELECT FROM ...) or large dumps, unusual account usage.
- IDS/Network: large outbound connections from web tier to external IPs, unusual SMB/DB connections.

Quick actions:

- Validate the alert, capture the raw request (full URL, headers, body), and classify severity.
- Preserve web logs, WAF logs, and take filesystem snapshot of webroot/uploads and associated timestamps.
- If possible, sandbox the suspicious payload to analyze behavior without executing on production.

5.2 Containment (Immediate / Short-term)

- Block or rate-limit offending IPs and user-agents at WAF/proxy; apply temporary rules to drop malicious payload patterns.
- Disable vulnerable endpoints or maintenance mode for affected application if immediate risk to data is high.
- Rotate credentials for service accounts used by the application and restrict DB access from web tier until remediated.
- Quarantine affected hosts and isolate from backend databases if web shell or RCE is confirmed.

5.3 Investigation & Forensic Triage

- Collect full HTTP request/response pairs, server logs, WAF logs, database query logs, and any uploaded files.
- Take forensic copies of webroot, application containers, and database backups (preserve chain-of-custody).
- Perform static/dynamic analysis of uploaded files (web shells) in a controlled sandbox and compute hashes for IOCs.
- Map data accessed or exported (which tables, objects, or files) and timeline of access.

5.4 Eradication

- Remove web shells, malicious uploads, and artifacts (after evidence capture). Replace compromised binaries/app builds with clean images.
- Patch the underlying vulnerability (fix input validation, parameterize queries, disable dangerous functions) and redeploy.
- Rotate database credentials, service account keys, and any affected API keys; revoke tokens used during the incident.
- Harden WAF rules and apply runtime protections to prevent the same exploitation vector.

5.5 Recovery

- Restore services in a staged manner from validated, clean builds; test endpoints thoroughly.
- Validate data integrity and restore any corrupted or exfiltrated data from trusted backups if required.
- Increase monitoring for the remediated endpoints and watch for indicators of re-compromise for 60–90 days.

5.6 Post-Incident Activities

- Full root-cause analysis, update secure coding guidelines and application threat models.
- Update WAF rules, SIEM detections (Sigma), and automated tests in CI to prevent regression.
- Run focused code reviews, dependency scans and a dedicated penetration test on the remediated application.
- Notify stakeholders, regulatory bodies (if required), and share IOCs with threat intel communities.

6 MITRE ATT&CK Framework Mapping

Web Application & Internet-Facing Attacks - ATT&CK Mapping

- **Initial Access / Exploitation:** T1190 (Exploit Public-Facing Application), T1505 (Server Software Component)
- **Execution / Persistence:** T1059 (Command and Scripting Interpreter), T1505.003 (Web Shell)
- **Impact / Exfiltration:** T1486 (Data Encrypted for Impact) via web-tier, T1041 (Exfiltration Over C2 Channel)
- **Defense Evasion:** T1027 (Obfuscated Files or Information), T1070 (Indicator Removal on Host)

7 Key Telemetry & Logs to Collect

- Web server access and error logs with full request/response bodies where allowed.
- WAF logs (detections, blocked requests, rule IDs), application logs, and API gateway logs.
- Database audit logs (slow queries, large selects, dump exports), and filesystem change logs for webroot/uploads.
- Container/orchestration logs, host process lists and EDR traces for web server processes.
- Network PCAPs at ingress/egress points during timeframe of suspicious activity.

8 Subcategory Scenarios (Realistic)

Note: Each scenario below is an operational SOC/IR narrative — includes detection, investigative steps, containment actions, eradication, recovery and lessons learned.

Scenario A: Web Shell Upload & Execution — Arbitrary File Upload to Admin Panel

Summary: An attacker exploited a weak file upload validation on an e-commerce admin panel, uploading a PHP web shell disguised as an image. The web shell was executed via a crafted request, providing the attacker interactive command execution in the web server context and access to the database connection string.

Detection:

- WAF: blocked and logged suspicious file uploads with unusual content-type and execution attempts.
- Web logs: POST to upload endpoint with multipart content and later GET requests to the uploaded PHP file.
- EDR/Host: web server process spawning shell commands and making outbound connections to unfamiliar IPs.

Investigation & Actions:

1. **Triage & classification:** Classified as *Critical* due to confirmed RCE and exposure of DB credentials.
2. **Immediate containment:** Block the attacker IPs at WAF/firewall and place the application in maintenance mode; remove the uploaded file (after collecting a forensic copy).
3. **Forensic collection:** Preserve web logs, WAF logs, the uploaded file, process memory of the web server, and DB access logs for the timeframe.
4. **Hunt:** Search for other uploads with similar file characteristics and scan the webroot and backups for other shells.

Containment & Eradication:

- Remove web shell artifacts, clean or reimage compromised host, and rotate any credentials found in configuration files.
- Patch upload validation (restrict extensions, validate magic bytes, store uploads outside webroot), and deploy WAF rules to block similar payloads.
- Rebuild the webserver from trusted images if persistence cannot be ruled out.

Recovery:

- Restore application from a clean build and verify configuration hardening; validate DB access and rotate DB credentials.
- Monitor the application and outbound connections for at least 60 days for signs of recompromise.

Outcome & Lessons:

- Root cause: insufficient file validation and uploads served from webroot. Mitigations: implement safe upload handling, content-type validation, and store uploads off-webroot; enhance WAF rules and runtime monitoring.

Scenario B: SQL Injection (SQLi) — Authenticated Search Parameter Leads to Data Exfil

Summary: A parameter in a search endpoint failed to use parameterized queries. An authenticated attacker crafted a payload that enumerated table names and exfiltrated rows containing customer PII via timed blind SQLi and concatenated responses into HTTP responses.

Detection:

- WAF: alerts for SQLi signatures and unusual SQL-like payloads in request parameters.
- DB logs: repeated slow queries with unusual patterns originating from the web application account.
- Application logs: spikes in response times for the search endpoint coinciding with unusual query strings.

Investigation & Actions:

1. **Triage & classification:** Classified as *High/Critical* depending on data volume accessed.
2. **Immediate containment:** Block the attacker IP and WAF rule to drop the malicious parameter patterns; temporarily disable the vulnerable endpoint if necessary.
3. **Forensic collection:** Capture request logs, full DB query logs, and export potential exfiltrated data hashes; snapshot DB for forensic analysis.
4. **Hunt:** Search for similar payloads across other endpoints and identify other accounts that may have been used.

Containment & Eradication:

- Fix the vulnerable code path: parameterize queries/ORM, apply input validation and least-privilege DB accounts.
- Rotate DB credentials that the web app uses and audit for unexpected privilege grants.
- Deploy WAF signatures and detection rules tuned to the observed exploitation technique.

Recovery:

- Restore from clean backups if data was tampered with; notify data owners and Legal/Compliance if PII was exposed.
- Conduct a full code review and penetration test for the application and similar endpoints.

Outcome & Lessons:

- Root cause: unsafe dynamic SQL and excessive DB privileges for the web account. Mitigations: parameterized queries, principle-of-least-privilege, and stronger DB auditing.

Scenario C: Cross-Site Scripting (XSS) — Reflected XSS Used to Steal Session Cookies

Summary: A reflected XSS vulnerability in a legacy support portal allowed an attacker to craft a URL that, when visited by support staff, executed JavaScript that exfiltrated session cookies to the attacker-controlled domain. One privileged support user's session was captured leading to unauthorized access to several customer support tickets.

Detection:

- WAF: alerts for XSS payload patterns and JavaScript content in query parameters.
- SIEM: unusual outbound requests from support user's browser to known attacker domain shortly after visiting portal.
- Application logs: GET requests with long query strings containing script tags.

Investigation & Actions:

1. **Triage & classification:** Classified as *High* due to privileged user session compromise.
2. **Immediate containment:** Invalidate sessions for affected users, block the attacker domain, and remove cached pages that contained the reflected payload if possible.
3. **Forensic collection:** Preserve web logs, capture the malicious URL, and collect browser telemetry (if available) from affected user devices.
4. **Hunt:** Look for other users who may have visited the malicious URL and scan for similar vulnerable endpoints.

Containment & Eradication:

- Patch the vulnerability by output encoding/escaping, enforce Content Security Policy (CSP), and sanitize user-supplied input.
- Reissue session tokens and require MFA re-validation for affected users.
- Add WAF rules to block common XSS payload patterns and suspicious referrers.

Recovery:

- Restore normal operations after verifying fixes and monitoring; conduct user awareness for support staff on suspicious links.
- Run automated scanners for XSS across the application and include XSS tests in CI pipelines.

Outcome & Lessons:

- Root cause: lack of output encoding and legacy code paths not covered by modern frameworks. Mitigations: implement CSP, escape outputs, and ensure automated security tests in CI.

9 Appendices

9.1 Appendix A — Useful SIEM / Investigation Queries

Splunk: find suspicious POSTs to upload endpoints

```
index=web sourcetype=access_combined
| search uri_path="/admin/upload" OR uri_path="/upload"
| table _time clientip user agent uri_path status bytes
```

Splunk: detect SQLi patterns (heuristic)

```
index=web (uri_query="*' OR *--*" OR uri_query="*UNION*SELECT*" OR uri_query="*/*
    select*/*")
| stats count by clientip, uri_path, uri_query
```

Kusto / Sentinel: detect long encoded strings in DNS or URLs

```
AppRequests
| where TimeGenerated > ago(7d)
| where strlen(RequestUri) > 200 or RequestUri contains "base64"
| project TimeGenerated, ClientIP, RequestUri
```

MySQL: find large result sets or slow queries

```
SELECT event_time, user_host, argument
FROM mysql_slow_log
WHERE query_time > 5 OR argument LIKE '%UNION%' OR argument LIKE '%SELECT%FROM%';
```

9.2 Appendix B — Forensic Artifact Locations

- Web server access/error logs (IIS, Apache, Nginx) — preserve raw request bodies where allowed.
- WAF logs and rule IDs for blocked requests.
- Application logs (application-level traces, stack traces), upload directories and temp folders.
- Database audit logs and binary logs (MySQL binlog, MSSQL audit).
- Container images, orchestration logs (Kubernetes events, pod logs), and host EDR captures.

9.3 Appendix C — Incident Report Template (Summary)

- Incident ID / Detection timestamp / Severity / Playbook invoked.
- Affected application(s), hosts, database(s), and scope of data accessed.
- Evidence preserved (web logs, WAF logs, DB snapshots, file hashes) and IOCs (IPs, URIs, file hashes).
- Actions taken (containment, eradication, recovery) with timestamps and owners.
- Root cause analysis, mitigation actions, and long-term security improvements.
- Notifications performed (internal/external), regulatory reporting, and lessons learned.