# Analysis Results

# bsincomevalidation-main.zip

Report Date
2026-02-11 07:10:50
Report Author
admin
Classification Method
By severity
Product Version
10.16
Rules Version
10.127506

# Confidentiality Note

This report is intended only for the person(s) or entity to which it is addressed and contains confidential and privileged information. If you are not the intended recipient, you must not view, use, copy, disclose, or otherwise disseminate this report or any part of it. Doing so is strictly prohibited, and may result in legal proceedings. If you received this in error, please notify the sender immediately and destroy any copies of this information.

# 01 Project Information

Project Name

bsincomevalidation-main.zip

UUID

5d4514db-99fe-4290-b09d-1c72eb57cf53

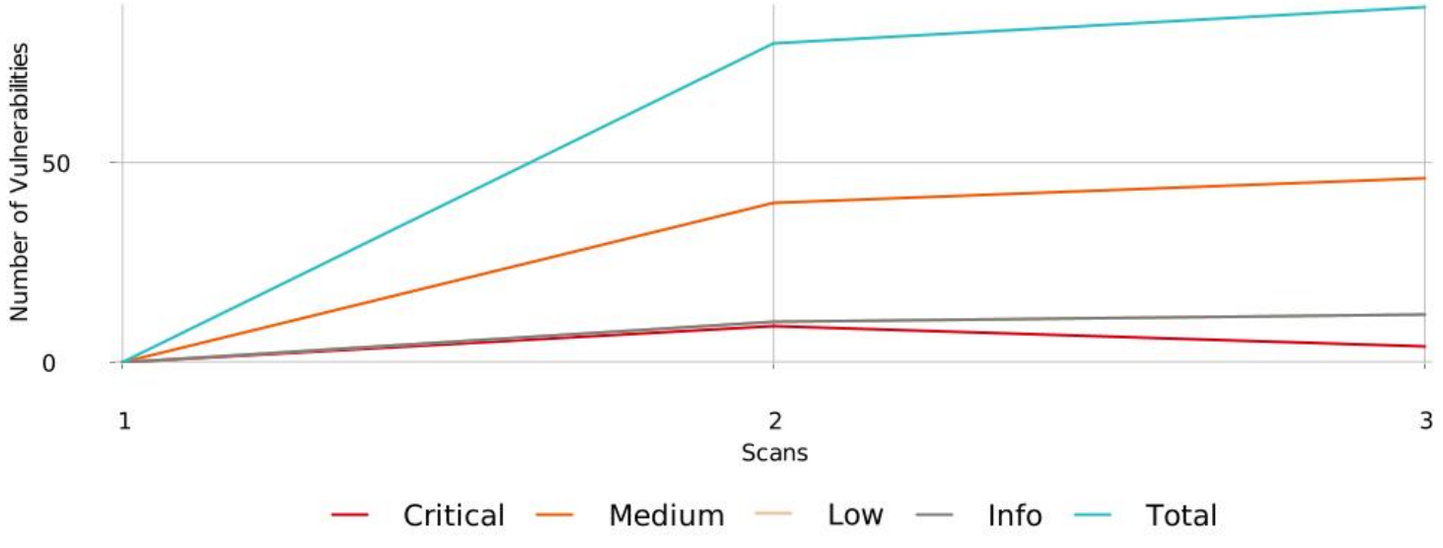
Project in DerScanner

# Dynamics by vulnerabilities

Vulnerabilities are divided by severity level: critical, medium, low and info.

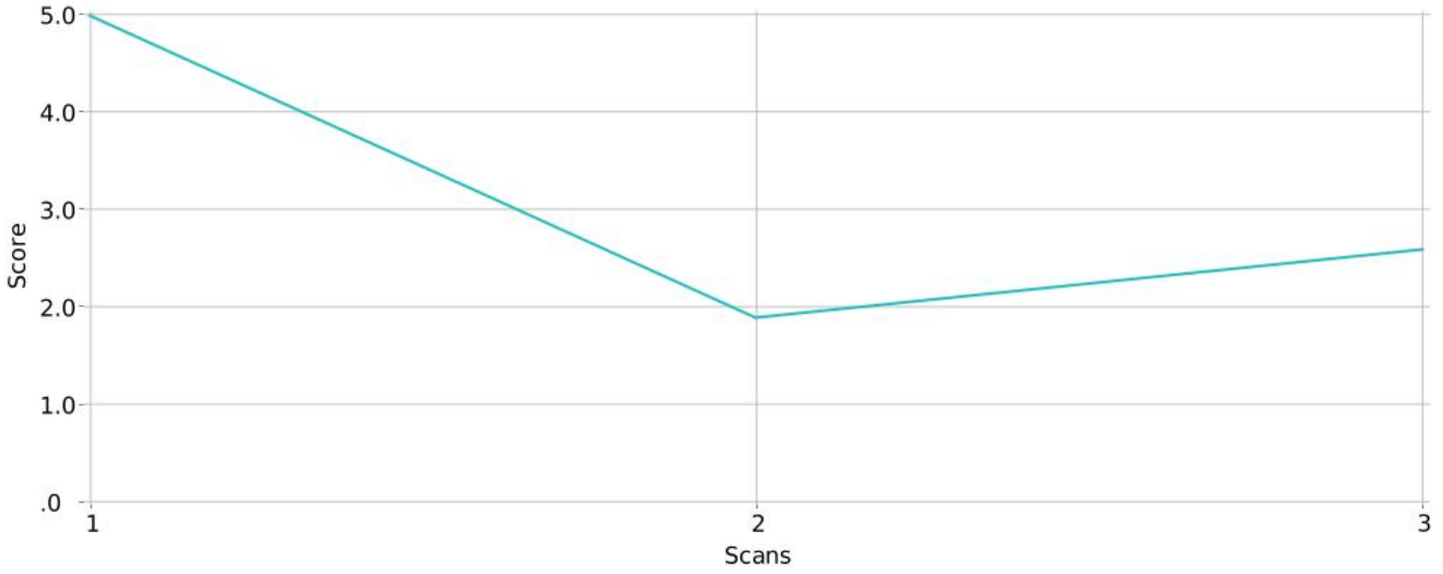
| CRITICAL LEVEL | MEDIUM LEVEL | LOW LEVEL | INFORMATION |
|---|---|---|---|
| Likely to lead to compromise confidential data and violation of the integrity of the system. | May be less likely to lead to compromising confidential data and violating the integrity of the system, or are less serious security | Can become a potential security risk. | Signal a violation of good programming practice. |

First of all, pay attention to vulnerabilities of critical and medium levels.



# Dynamics by rating

The app score is calculated on a scale from 0 to 5. Score is calculated based on the number of critical and medium level vulnerabilities. The impact of critical vulnerabilities is greater than that of medium level vulnerabilities, and does not take into account the amount of code. Medium level vulnerabilities are taken into account based on their frequency and total number of source code lines.



# Scan History

| | Date and Time | Status | Languages | Lines of Code | Number of Vulnerabilities | | | | | Score |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Critical | Medium | Low | Info | Total | |
| 3/3 | 2026-02-11 07:01:59 | completed | Config files, Python, PL/SQL, T-SQL, JavaScript, HTML5 | 34 496 | 4 | 46 | 12 | 27 | 89 | 2.6/5.0 |
| 2/3 | 2026-02-10 10:04:05 | completed | Config files, Python, PL/SQL, T-SQL, JavaScript, HTML5 | 33 841 | 9 | 40 | 10 | 21 | 80 | 1.9/5.0 |
| 1/3 | 2026-01-12 07:43:23 | completed | Config files | 4 | 0 | 0 | 0 | 0 | 0 | 5.0/5.0 |

02

# Scan Information

3/3 2026-02-11 07:01:59
10.127506

## Scan Statistics

Status
completed

Score
2.6/5.0

Duration
0:03:30

Lines of Code
34 496

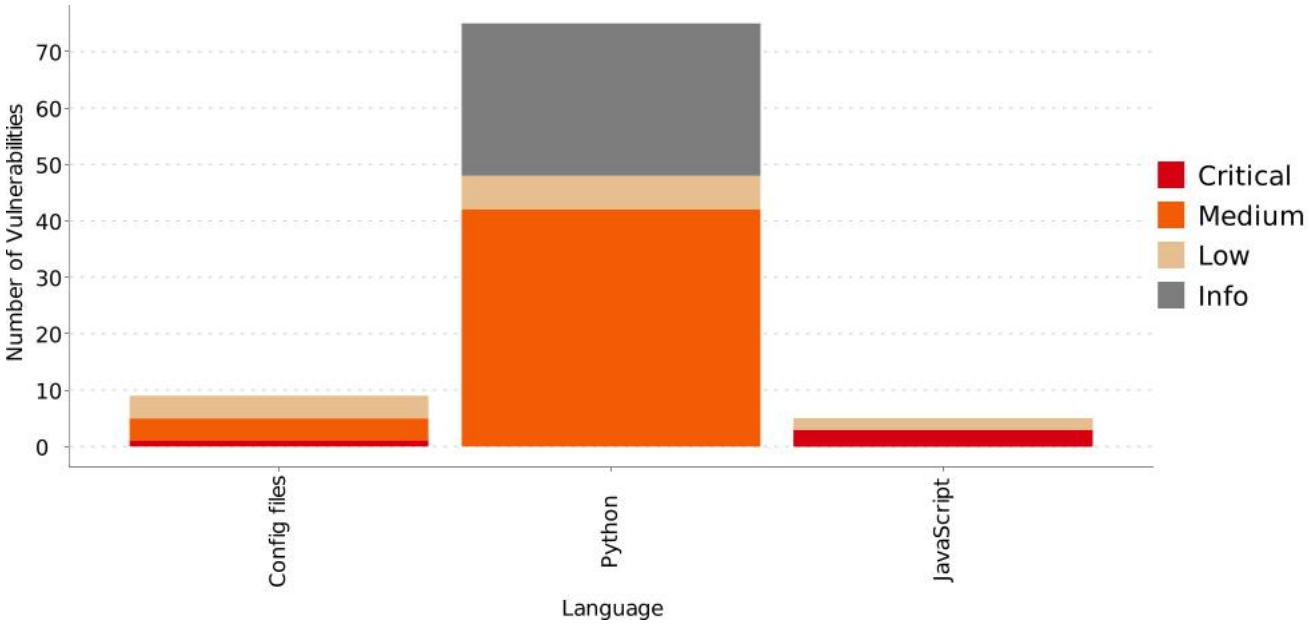
Vulnerabilities

| 4 | 46 | 12 | 27 |
|---|---|---|---|
| Critical | Medium | Low | Info |

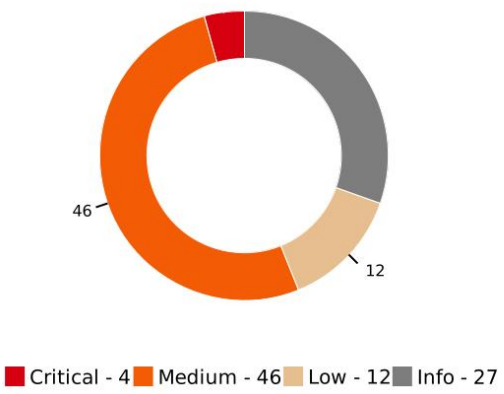
89
Total

| Language | Status | Duration | Lines of Code | Number of Vulnerabilities | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | Critical | Medium | Low | Info | Total |
| Config files | completed | 0:00:02 | 20 452 | 1 | 4 | 4 | 0 | 9 |
| Python | completed | 0:03:21 | 13 267 | 0 | 42 | 6 | 27 | 75 |
| PL/SQL | completed | 0:00:02 | 257 | 0 | 0 | 0 | 0 | 0 |
| T-SQL | completed | 0:00:01 | 256 | 0 | 0 | 0 | 0 | 0 |
| JavaScript | completed | 0:00:02 | 212 | 3 | 0 | 2 | 0 | 5 |
| HTML5 | completed | 0:00:00 | 52 | 0 | 0 | 0 | 0 | 0 |

## Language Statistics



## Diagram of identified vulnerabilities




## Vulnerable Dependencies

# Vulnerability List

Vulnerabilities are displayed accordingly to export settings: **50 selected**

Actual: **50 of 89**

| | | |
|---|---|---|
| Critical vulnerabilities | | **4** |
| Hardcoded encryption key in config | Config files | 1 |
| Dockerfile:15 | | Not processed |
| Persistent XSS | JavaScript | 3 |
| src/ui/db_viewer.html:408 | | Not processed |
| src/ui/db_viewer.html:450 | | Not processed |
| src/ui/db_viewer.html:483 | | Not processed |
| Medium-level vulnerabilities | | **46** |
| Persistent authentication | Config files | 3 |
| src/ui/taxpayer_search_app.py:29 | | Not processed |
| src/ui/taxpayer_search_app.py:244 | | Not processed |
| src/ui/upload_app.py:43 | | Not processed |
| Wrong user settings | Config files | 1 |
| Dockerfile:1 | | Not processed |
| DOS attack via regular expressions possible | Python | 1 |
| src/halyk_ind/transactions.py:79 | | Not processed |
| Information leak | Python | 12 |
| run_all_services.py:153 | | Not processed |

* Rejected vulnerabilities are not taken into account

## Medium-level vulnerabilities

| Information leak | Python | |
|---|---|---|
| run_all_services.py:186 | | Not processed |
| run_all_services.py:215 | | Not processed |
| src/bcc/footer.py:211#215 | | Not processed |
| src/bcc_ind/parser.py:341 | | Not processed |
| src/halyk_business/footer.py:188 | | Not processed |
| src/halyk_ind/parser.py:319 | | Not processed |
| src/halyk_ind/parser.py:322 | | Not processed |
| src/halyk_ind/parser.py:326 | | Not processed |
| src/utils/income_calc.py:296#300 | | Not processed |
| src/utils/income_calc.py:309#313 | | Not processed |
| src/utils/income_calc.py:322#326 | | Not processed |
| **Resource injection** | **Python** | **26** |
| src/alatau_city_bank/batch_parse.py:98 | | Not processed |
| src/alatau_city_bank/batch_parse.py:137 | | Not processed |
| src/api/app.py:844 | | Not processed |
| src/api/storage.py:134 | | Not processed |
| src/api/storage.py:149 | | Not processed |
| src/api/storage.py:157 | | Not processed |
| src/api/storage.py:167 | | Not processed |
| src/bcc/batch_parse.py:81 | | Not processed |
| src/bcc/batch_parse.py:176 | | Not processed |
| src/bcc/footer.py:189 | | Not processed |
| src/bcc/parser.py:40 | | Not processed |
| src/eurasian_bank/batch_parse.py:77 | | Not processed |
| src/eurasian_bank/batch_parse.py:144 | | Not processed |

## Medium-level vulnerabilities

| Resource injection | Python | |
|---|---|---|
| src/forte_bank/batch_parse.py:113 | | Not processed |
| src/forte_bank/batch_parse.py:188 | | Not processed |
| src/forte_bank/footer.py:194 | | Not processed |
| src/freedom_bank/batch_parse.py:105 | | Not processed |
| src/freedom_bank/batch_parse.py:185 | | Not processed |
| src/halyk_business/footer.py:140 | | Not processed |
| src/halyk_business/footer.py:153 | | Not processed |
| src/halyk_business/footer.py:193 | | Not processed |
| src/halyk_ind/parser.py:53 | | Not processed |
| src/kaspi_pay/parser.py:40 | | Not processed |
| src/utils/convert_pdf_json_page.py:201 | | Not processed |
| src/utils/convert_pdf_json_pages.py:61 | | Not processed |
| src/utils/path_security.py:95 | | Not processed |
| SQL injection | Python | 3 |
| src/api/app.py:812 | | Not processed |
| src/ui/upload_app.py:1191 | | Not processed |
| src/ui/upload_app.py:1226 | | Not processed |

## Low-level vulnerabilities 0

## Info-level vulnerabilities 0

# Analysis Results

## Hardcoded encryption key in config (Config files)

### Description

Encryption key is hardcoded in a configuration file. This may lead to an application data compromise.
Eliminating security risks related to hardcoded passwords or keys is extremely difficult. This data is available at least to every developer of the application. Moreover, after the application is installed, removing password or key from its code is possible only via an update. Constant strings are easily extracted from the compiled application by decompilers. Therefore, an attacker does not necessarily need to have an access to the source code to find out the value of the key.
Insufficient Cryptography vulnerabilities take the fifth place in the “OWASP Top 10 2016” mobile application vulnerabilities ranking.

### Example

In the following example, the encryption key plist is hardcoded in the file plist:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist>
<dict>
   <key>cryptoKey</key>

<string>1ac96ba5217fe15c4b76b8c5191583b6a24488581545844d2e67fa83c537
f0b3</string>
</dict>
</plist>
```

### Recommendations

- Do not use hardcoded encryption keys.
- To generate the keys, it is recommended to use a cryptographically secure PRNG (pseudorandom number generators).
- To store keys, you need to use special solutions, such as Hardware Security Modules.
- For security reasons, you need to regularly update your keys.

## Links

1. Use of hard-coded password
2. CWE-321: Use of Hard-coded Cryptographic Key
3. Mobile Top 10 2014-M6-Broken Cryptography
4. OWASP Top 10 2013-A5-Security Misconfiguration
5. OWASP Top 10 2013-A6-Sensitive Data Exposure
6. RNCryptor README - github.com
7. Mobile Top 10 2016-M5-Insufficient Cryptography
8. CWE-798: Use of Hard-coded Credentials

## Vulnerability Entries

### Dockerfile:15

Level Critical

Status Not processed

```
12
13 RUN pip install --no-cache-dir --upgrade pip && \
14     pip install --no-cache-dir $(grep -v "pipreqs" requirements.txt) && \
15     pip install --no-cache-dir uvicorn fastapi python-multipart
16
17 COPY . .
18
```

## Persistent XSS (JavaScript)

## Description

Persistent XSS or server XSS attack is possible.
Cross-site scripting is one of the most common types of attacks on web applications. XSS attacks take seventh place in the “OWASP Top 10 2017” list of ten most significant vulnerabilities in web applications.
The main phase of any XSS attack is an imperceptible for the victim execution of a malicious code in the context of the vulnerable application. For this purpose, the functionality of the client application (browser) is used that allows to automatically execute scripts embedded in web page code. In most cases, these malicious scripts are implemented in JavaScript.
Consequences of an XSS attack vary from violations of application functionality to complete loss of user data confidentiality. The malicious code during the XSS attack can steal user HTTP-cookie, which gives an attacker the ability to make requests to the server on behalf of the user.

OWASP suggests the following classification of XSS attacks:

- Server type XSS occurs when data from an untrusted source is included in the response returned by the server. The source of such data can be both user input and server database (where it had been previously injected by an attacker who exploited vulnerabilities in the server-side application).
- Client type XSS occurs when the raw data from the user input contains code that changes the Document Object Model (DOM) of the web page received from the server. The source of such data can be both the DOM and the data received from the server (e.g., in response to an AJAX request).

Typical server type attack scenario:

1. Unvalidated data, usually from a HTTP request, gets into the server part of the application.
2. The server dynamically generates a web page that contains the unvalidated data.
3. In the process of generating a web page, server does not prevent the inclusion of an executable code that can be executed in the client (browser), such as JavaScript code language, HTML-tags, HTML-attributes, Flash, ActiveX, etc., in the page code.
4. The victim's client application displays the web page that contains the malicious code injected via data from an untrusted source.
5. Since malicious code is injected in the web page coming from the known server, the client part of the application (browser) executes it with the rights set for the application.
6. This violates the same-origin policy, according to which the code from the one source must not get an access to resources from another source.

Client type attacks are executed in a similar way with the only difference that the malicious code is injected during the phase of the client application work with the document object model received from the server.

## Example

In the following example, the application obtains user identifier (name) from the URL and displays it:

```
var name = localStorage.getItem("name");
document.write(name);
```

The code operates correctly if name does not contain special characters. If name contains code it will be executed in the victim's web browser.

If the name values do not contain special characters, the code behaves correctly. But if the name value is derived from data from an untrusted source (e.g., user input), the attacker can store the malicious code in the database. Such attacks are particularly dangerous because they may affect a large number of users.

## Recommendations

- Implement a validation mechanism. Whitelist is more secure but less flexible than the blacklist. The blacklist must at least include the characters "&", "<", ">", and quotation marks.
- Many web application servers provide their own mechanisms of protection against XSS, but they may not be considered sufficient. There is no guarantee that the application will

run in conjunction with the server that updates these mechanisms timely and completely.

## Links

1. OWASP: Cross-site Scripting (XSS)
2. CWE-79: Improper Neutralization of Input During Web Page Generation
3. Types of Cross-Site Scripting - OWASP
4. OWASP: XSS Prevention Cheat Sheet
5. OWASP Top 10-2017 A7-Cross-Site Scripting (XSS)
6. CWE CATEGORY: OWASP Top Ten 2017 Category A3 - Sensitive Data Exposure
7. CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
8. CWE-81: Improper Neutralization of Script in an Error Message Web Page
9. CWE-83: Improper Neutralization of Script in Attributes in a Web Page
10. Cross-site Scripting (XSS) Affecting jquery-mobile package

## Vulnerability Entries

### src/ui/db_viewer.html:408

Level Critical

Status Not processed

```
405    });
406
407    html += '</tbody></table>';

408    document.getElementById('statementsContainer').innerHTML = html;

409 } catch (error) {
410    showError('Ошибка загрузки выписок: ' + error.message);
411 }
```

Trace

row

src/ui/db_viewer.html:394

```
391 html += '</tr></thead><tbody>';
392
393 data.forEach(row => {

394    const period = `${escapeHtml(row.period_from || 'Н/Д')} по ${escapeHtml(row.
```

```
period_to || 'Н/Д')}` ;
395     const created = new Date(row.created_at).toLocaleDateString();
396     html += `<tr>
397         <td>${escapeHtml(row.bank || '')}</td>
```

html

src/ui/db_viewer.html:408

```
405     });
406
407     html += '</tbody></table>';
408     document.getElementById('statementsContainer').innerHTML = html;
409 } catch (error) {
410     showError('Ошибка загрузки выписок: ' + error.message);
411 }
```

## src/ui/db_viewer.html:450

Level Critical

Status Not processed

```
447     });
448
449     html += '</tbody></table>';
450     document.getElementById('transactionsContainer').innerHTML = html;
451 } catch (error) {
452     showError('Ошибка загрузки транзакций: ' + error.message);
453 }
```

Trace

row

src/ui/db_viewer.html:434

```
431 html += '</tr></thead><tbody>';
432
433 data.forEach(row => {
434     const date = escapeHtml(row.operation_date || 'Н/Д');
435     const debit = row.debit_amount ? `<span class="amount-cell negative">${escapeHtml(row.debit_amount.toLocaleString())}</span>` : '-';
436     const credit = row.credit_amount ? `<span class="amount-cell positive">${escapeHtml(row.credit_amount.toLocaleString())}</span>` : '-';
437
```

html

src/ui/db_viewer.html:450

```
447     });
448
449     html += '</tbody></table>';
450     document.getElementById('transactionsContainer').innerHTML = html;
451 } catch (error) {
452     showError('Ошибка загрузки транзакций: ' + error.message);
453 }
```

## src/ui/db_viewer.html:483

Level Critical

Status Not processed

```
480     });
481
482     html += '</tbody></table>';
```

```
483     document.getElementById('clientsContainer').innerHTML = html;
484 } catch (error) {
485     showError('Ошибка загрузки клиентов: ' + error.message);
486 }
```

Trace

row

src/ui/db_viewer.html:471

```
468 html += '</tr></thead><tbody>';
469
470 data.forEach(row => {
471     const created = new Date(row.created_at).toLocaleDateString();
472     html += `<tr>
473         <td><strong>${escapeHtml(row.iin_bin || '')}</strong></td>
474         <td>${escapeHtml(row.full_name || '')}</td>
```

html

src/ui/db_viewer.html:483

```
480     });
481
482     html += '</tbody></table>';
483     document.getElementById('clientsContainer').innerHTML = html;
484 } catch (error) {
485     showError('Ошибка загрузки клиентов: ' + error.message);
486 }
```

## Persistent authentication (Config files)

## Description

The application uses a permanent Bearer authentication token. At the end of the life of the token, the application may not work correctly. If the token has a long lifetime, after getting it, any subject will be able to authorize into the service.

## Example

In the following example, the application uses a persistent Bearer token:

```
{
  "key": "Authorization",
  "value": "Bearer ...Token..."
}
```

## Recommendations

- Dynamically add token to request header.
- Limit the validity of the authentication token.

## Links

1. FormsAuthentication.RedirectFromLoginPage Method - msdn.microsoft.com
2. OWASP Top 10 2017 A2-Broken Authentication
3. CWE CATEGORY: OWASP Top Ten 2017 Category A2 - Broken Authentication

## Vulnerability Entries

### src/ui/taxpayer_search_app.py:29

Level Medium

Status Not processed

```
26 # Константы
27 # SECURITY: Use environment variables instead of hardcoded values
28 import os

29 DEFAULT_PORTAL_TOKEN = os.environ.get("TAXPAYER_API_PORTAL_TOKEN", "")

30
31
32 def init_session_state() -> None:
```

## src/ui/taxpayer_search_app.py:244

Level Medium

Status Not processed

```
241 try:
242     client = TaxpayerAPIClient(
243         portal_host=st.session_state.portal_host,
244         portal_token=st.session_state.portal_token
245     )
246
247     taxpayer_type_enum = TaxpayerType[taxpayer_type]
```

## src/ui/upload_app.py:43

Level Medium

Status Not processed

```
40     "TAXPAYER_API_PORTAL_HOST",
41     "https://portal.kgd.gov.kz"  # Default, should be overridden via env var
42 )
43 TAXPAYER_API_PORTAL_TOKEN = os.environ.get(
44     "TAXPAYER_API_PORTAL_TOKEN",
45     ""  # Must be set via environment variable in production
46 )
```

## Wrong user settings (Config files)

## Description

A microservice is run by default with root privileges in the container. While there's still some default protection left (Linux capabilities, either AppArmor or SELinux profiles) it removes one layer of protection. This extra layer broadens the attack surface. It also violates the least privilege principle and from the OWASP perspective is an insecure default.
For privileged containers (with privileged flag), a microservice breakout into the container is almost comparable to running without any container. Privileged containers endanger your whole host and all other containers.

## Example

The following are instructions for creating a user inside a container:
FROM alpine
RUN groupadd -r myuser && useradd -r -g myuser myuser
<HERE DO WHAT YOU HAVE TO DO AS A ROOT USER LIKE INSTALLING PACKAGES ETC.>
USER myuser
Only commands that require root privileges can be run as root.

## Recommendations

Configure the container to use an unprivileged user.

## Links

1. CWE-250: Execution with Unnecessary Privileges
2. CWE-657: Violation of Secure Design Principles
3. OWASP Top 10 2017 A5:2017-Broken Access Control
4. OWASP Docker Security Cheat Sheet

## Vulnerability Entries

### Dockerfile:1

Level Medium

Status Not processed

```
1 FROM python:3.12-slim

2 RUN apt-get update && apt-get install -y \
3     ghostscript \
```

```
4    python3-tk \
```

## DOS attack via regular expressions possible (Python)

### Description

The regexp used is from an unreliable source, which can be computationally intensive for some inputs. Regular expression denial of service (ReDOS) attack is possible.
Regular expressions are widely used in applications to validate the user-supplied data. Expressions containing structures like (( )+)+ cause execution of a significant amount of iterations. By inputting a certain type of string an attacker can disrupt the application operation. All implementations of regular expressions have such vulnerabilities.

### Example

The following regular expression performs a cycle of 32,768 iterations on the input string aaaaaaaaaaaaaaaX:

```
^(a+)+$
```

### Recommendations

- Do not use the data that is obtained from an untrusted source has not passed validation in regular expressions.
- Avoid regular expressions containing nested groups of repeating characters.
- Check the regular expression for the possibility of ReDOS attack using safeRegex.

### Links

1. OWASP: Regular expression Denial of Service
2. Runaway Regular Expressions: Catastrophic Backtracking – regular-expressions.info
3. saferegex – Tool for testing regular expressions for ReDoS vulnerabilities
4. re - Regular expression operations
5. CWE-400

## Vulnerability Entries

### src/halyk_ind/transactions.py:79

Level Medium

Status Not processed

```
76    "fee":          r"\bкомисси[яи]\b",
77    "account":      r"\b№\s*карточки/счета\b|\bкарточки/счета\b",
78 }

79 RX = {k: re.compile(v, re.I) for k, v in ANCHORS.items()}

80
81 def clean_spaces(s: str) -> str:
82     return re.sub(r"[ \t\r\n\u00A0\u202F]+", " ", s or "").strip()
```

## Information leak (Python)

## Description

System configuration information leak is possible. This can help an attacker to create a plan of an attack. Debug information and error messages can be written to the log, displayed to the console, or sent to the user depending on the system settings. In some cases, an attacker can make a conclusion about the system vulnerabilities from the error message. For example, a database error can indicate insecurity against SQL injection attacks. Information about the version of the operating system, server applications and system configurations can also be of value to the attacker.

## Example

In the following example, the application turns on debug mode:
from django.conf import settings
settings.configure(DEBUG=True)

## Recommendations

- Exclude detailed information about the system and its configuration from the error messages.

## Links

1. OWASP Top 10 2013-A5-Security Misconfiguration
2. CWE-497: Exposure of System Data to an Unauthorized Control Sphere
3. OWASP Top 10 2017-A3-Sensitive Data Exposure
4. OWASP Top 10 2017-A6-Security Misconfiguration
5. CWE CATEGORY: OWASP Top Ten 2017 Category A5 - Broken Access Control
6. CWE CATEGORY: OWASP Top Ten 2017 Category A6 - Security Misconfiguration
7. CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
8. CWE-209: Generation of Error Message Containing Sensitive Information
9. CWE-489

## Vulnerability Entries

### run_all_services.py:153

Level Medium

Status Not processed

```
150         stdout, _ = proc.communicate(timeout=1)
151         if stdout:
152             print(f"\n  Процесс {i} завершился с ошибкой:")
153             print(stdout[-500:])  # Последние 500 символов
154     except:
155         pass
156
```

### run_all_services.py:186

Level Medium

Status Not processed

```
183             try:
184                 stdout, _ = proc.communicate(timeout=1)
185                 if stdout:
186                     print(stdout[-500:])
187             except:
```

```
188                 pass
189 except KeyboardInterrupt:
```

## run_all_services.py:215

Level Medium

Status Not processed

```
212         stdout, stderr = proc.communicate(timeout=1)
213         if stderr:
214             print(f"\nОшибки из процесса {i}:")
215             print(stderr)
216     except:
217         pass
218
```

## src/bcc/footer.py:211#215

Level Medium

Status Not processed

```
208 DEBUG_MODE = os.environ.get("DEBUG_PARSER", "false").lower() == "true"
209
210 if DEBUG_MODE:
211     print({
212         "total_debit": res["total_debit"],
213         "total_credit": res["total_credit"],
214         "closing_balance": res["closing_balance"],
215     })
216 else:
217     print(f"Summary: debit={res['total_debit']}, credit={res['total_credit']}, balance={res
['closing_balance']}")
218 })
```

## src/bcc_ind/parser.py:341

Level Medium

Status Not processed

```
338 DEBUG_MODE = os.environ.get("DEBUG_PARSER", "false").lower() == "true"
339 if DEBUG_MODE:
340     with pd.option_context("display.max_colwidth", None):
341         print(df.head(3)[["Описание операции"]].to_string(index=False))
342 else:
343     print(f"Parsed {len(df)} rows (operation details hidden)")
344
```

Trace

Path

src/bcc_ind/parser.py:312

```
309 ap.add_argument("--calib-pages", default="1,2,3", help="Pages to try for calibration (e.g. '1,2,3').")
310 args = ap.parse_args()
311
312 pdf = Path(args.pdf)
313 if not pdf.exists():
314     raise SystemExit(f"File not found: {pdf}")
315
```

print

src/bcc_ind/parser.py:341

```
338 DEBUG_MODE = os.environ.get("DEBUG_PARSER", "false").lower() == "true"
339 if DEBUG_MODE:
340     with pd.option_context("display.max_colwidth", None):
341         print(df.head(3)[["Описание операции"]].to_string(index=False))
342 else:
```

```
343     print(f"Parsed {len(df)} rows (operation details hidden)")
344
```

## src/halyk_business/footer.py:188

Level Medium

Status Not processed

```
185 import os
186 DEBUG_MODE = os.environ.get("DEBUG_PARSER", "false").lower() == "true"
187 if DEBUG_MODE:
188     print(df.to_string(index=False))
189 else:
190     print(f"Summary: {len(df)} rows parsed (details hidden)")
191
```

## src/halyk_ind/parser.py:319

Level Medium

Status Not processed

```
316
317 if DEBUG_MODE:
318     print("=== HEADER ===")
319     print(header_df.to_string(index=False))
320
321     print("\n=== TX (first 20) ===")
322     print(tx_df.head(20).to_string(index=False))
```

## src/halyk_ind/parser.py:322

Level Medium

Status Not processed

```
319 print(header_df.to_string(index=False))
320
321 print("\n=== TX (first 20) ===")
322 print(tx_df.head(20).to_string(index=False))
323
324 print("\n=== FOOTER ===")
325 if not footer_df.empty:
```

## src/halyk_ind/parser.py:326

Level Medium

Status Not processed

```
323
324     print("\n=== FOOTER ===")
325     if not footer_df.empty:
326         print(footer_df.to_string(index=False))
327     else:
328         print("<empty>")
329 else:
```

## src/utils/income_calc.py:296#300

Level Medium

Status Not processed

```
293 if DEBUG_MODE:
294     print("\n[income_calc] examples excluded by KNP:")
295     # Only show non-sensitive columns in debug mode
296     print(
```

```
297       df.loc[df["ip_is_non_business_by_knp"],
298           [col_op_date, col_knp, col_credit]]
299        .head(max_examples)
300    )
```

```
301 else:
302    print(f"\n[income_calc] {n_knp} transactions excluded by KNP (details hidden)")
303
```

## src/utils/income_calc.py:309#313

Level Medium

Status Not processed

```
306 if DEBUG_MODE:
307    print("\n[income_calc] examples excluded by keywords:")
308    # Only show non-sensitive columns in debug mode
```

```
309    print(
310       df.loc[df["ip_is_non_business_by_keywords"],
311           [col_op_date, col_knp, col_credit]]
312        .head(max_examples)
313    )
```

```
314 else:
315    print(f"\n[income_calc] {n_kw} transactions excluded by keywords (details hidden)")
316
```

## src/utils/income_calc.py:322#326

Level Medium

Status Not processed

```
319 if DEBUG_MODE:
320    print("\n[income_calc] examples KEPT due to KNP=099 + 'возмещение/гарант':")
321    # Only show non-sensitive columns in debug mode
```

```
322    print(
323       df.loc[override_keep_mask,
```

```
324             [col_op_date, col_knp, col_credit]]
325         .head(max_examples)
326     )
```

```
327 else:
328     print(f"\n[income_calc] {n_override} transactions kept due to override rules (details hidden)")
329
```

## Resource injection (Python)

### Description

An attacker can gain access to reading and changing protected system resources of the application has the ability to change the resource identifier.
Resource injection occurs when an atacker can specify identifier that will be used to access the system resource (for example, the port number to connect to the network resource). This allows him, in particular, to transfer valuable data to thied-party server.

### Example

In the following example, the program executes a HTTP request in order to find out the price of tickets:

```
host_name = request.GET['host_example']
dbc = db.connect(host = host_name, port = 1235, dbname = ticket_data_base)
c = dbc.cursor()
###
result = c.execute('SELECT * FROM price_list')
###
```

If the request contains special characters, the attacker can receive confidential data.

### Recommendations

- Create a whitelist of valid resource IDs and allow a user to select from this list and not to set his/her own value.
- If maintaining a whitelist is too difficult because of the large number of valid IDs, create a whitelist of characters allowed in identifiers. Blacklist in this case is ineffective, as it is likely to initially be incomplete, or sooner or later cease to be relevant.
- If nonetheless blacklist is chosen as a validation mechanism, make sure that it takes into account all the possible encodings and special character values (different for different

operating systems). Changing the list should be simple when changing the requirements for validation.

## Links

1. OWASP Top 10 2017-A1-Injection
2. OWASP Top 10 2013-A1-Injection
3. OWASP Top 10 2013-A4-Insecure Direct Object References
4. CWE-99: Improper Control of Resource Identifiers (‘Resource Injection’)
5. Dangerous Python Functions, Part 2
6. Create, use, and remove temporary files securely
7. CWE CATEGORY: OWASP Top Ten 2017 Category A1 - Injection
8. CWE-1030

## Vulnerability Entries

### src/alatau_city_bank/batch_parse.py:98

Level Medium

Status Not processed

```
95          xref = {"error": f"{type(e).__name__}: {e}"}
96      out["XRef"] = xref
97
98 with open(out_path, "w", encoding="utf-8") as f:
99      json.dump(out, f, ensure_ascii=False, indent=2)
100
101 print(f"[json] Dumped PDF internals to {out_path}")
```

### src/alatau_city_bank/batch_parse.py:137

Level Medium

Status Not processed

```
134 if not json_path.exists() or not json_path.is_file():
135     raise ValueError(f"Invalid JSON path: {json_path}")
136
```

```
137 with open(json_path, "r", encoding="utf-8") as f:
138     pdf_json = json.load(f)
139
140 closing_date = header_df.iloc[0].get("closing_balance_date")
```

## src/api/app.py:844

Level Medium

Status Not processed

```
841     resolved_path = ui_file.resolve()
842     # Ensure path is within project or expected locations
843     if resolved_path.exists() and resolved_path.is_file():
844         with open(resolved_path, 'r', encoding='utf-8') as f:
845             return Response(content=f.read(), media_type="text/html")
846 except (OSError, ValueError) as e:
847     # Skip invalid paths
```

## src/api/storage.py:134

Level Medium

Status Not processed

```
131
132 # Save project metadata
133 project_file = self._get_project_file(project_id)
134 with open(project_file, 'w', encoding='utf-8') as f:
135     json.dump(project.to_dict(), f, ensure_ascii=False, indent=2)
136
137 # Create directory for project files
```

## src/api/storage.py:149

Level Medium

Status Not processed

```
146 if not project_file.exists():
147     return None
148
149 with open(project_file, 'r', encoding='utf-8') as f:
150     data = json.load(f)
151
152 return Project.from_dict(data)
```

## src/api/storage.py:157

Level Medium

Status Not processed

```
154 def update_project(self, project: Project):
155     """Update existing project"""
156     project_file = self._get_project_file(project.project_id)
157     with open(project_file, 'w', encoding='utf-8') as f:
158         json.dump(project.to_dict(), f, ensure_ascii=False, indent=2)
159
160 def get_projects_by_iin(self, iin: str) -> List[Project]:
```

## src/api/storage.py:167

Level Medium

Status Not processed

```
164 # Scan all project files
165 for project_file in self.base_dir.glob("project_*.json"):
166     try:
167         with open(project_file, 'r', encoding='utf-8') as f:
168             data = json.load(f)
169 
170         if data.get('iin') == iin:
```

## src/bcc/batch_parse.py:81

Level Medium

Status Not processed

```
78     out.update(dump_pages(pdf, **kw))
79     # we skip XRef for now – not needed for metadata validation
80 
81 with open(json_path, "w", encoding="utf-8") as f:
82     json.dump(out, f, ensure_ascii=False, indent=2)
83 
84 return json_path
```

## src/bcc/batch_parse.py:176

Level Medium

Status Not processed

```
173 # Security: Validate path before opening
174 from src.utils.path_security import validate_path
175 validated_path = validate_path(meta_json_path, pdf_meta_dir)
176 with open(validated_path, "r", encoding="utf-8") as f:
177     pdf_json = json.load(f)
178 
```

```
179 period_end = header_df.iloc[0].get("Период (конец)")
```

## src/bcc/footer.py:189

Level Medium

Status Not processed

```
186 validated_path = validate_path(jsonl_path)
187
188 pages: List[Dict[str, Any]] = []
189 with open(validated_path, "r", encoding="utf-8") as f:
190     for line in f:
191         line = line.strip()
192         if line:
```

## src/bcc/parser.py:40

Level Medium

Status Not processed

```
37
38 pages: List[Dict[str, Any]] = []
39
40 with open(validated_path, "r", encoding="utf-8") as f:
41     for line in f:
42         line = line.strip()
43         if not line:
```

## src/eurasian_bank/batch_parse.py:77

Level Medium

Status Not processed

```
74       )
75    )
76
77 with open(json_path, "w", encoding="utf-8") as f:
78     json.dump(out, f, ensure_ascii=False, indent=2)
79
80 return json_path
```

## src/eurasian_bank/batch_parse.py:144

Level Medium

Status Not processed

```
141
142 meta_json_path = ensure_pdf_meta_json(pdf_path, pdf_meta_dir)
143 try:
144     with open(meta_json_path, "r", encoding="utf-8") as f:
145         pdf_json = json.load(f)
146
147     period_end = header_df.iloc[0].get("period_end")
```

## src/forte_bank/batch_parse.py:113

Level Medium

Status Not processed

```
110     )
111
112 #  this is the important part
113 with open(json_path, "w", encoding="utf-8") as f:
114     json.dump(out, f, ensure_ascii=False, indent=2, default=str)
115
```

```
116 return json_path
```

## src/forte_bank/batch_parse.py:188

Level Medium

Status Not processed

```
185
186 meta_json_path = ensure_pdf_meta_json(pdf_path, pdf_meta_dir)
187 try:
188     with open(meta_json_path, "r", encoding="utf-8") as f:
189         pdf_json = json.load(f)
190
191     period_end = header_df.iloc[0].get("period_end")
```

## src/forte_bank/footer.py:194

Level Medium

Status Not processed

```
191 def parse_forte_footer(jsonl_path: str) -> Dict[str, Optional[float]]:
192     # Load pages
193     pages: List[Dict[str, Any]] = []
194     with open(jsonl_path, "r", encoding="utf-8") as f:
195         for line in f:
196             line = line.strip()
197             if line:
```

## src/freedom_bank/batch_parse.py:105

Level Medium

Status Not processed

```
102 # make sure there are no Decimal (or other non-JSON) types
103 out_clean = _json_safe(out)
104
105 with open(json_path, "w", encoding="utf-8") as f:
106     json.dump(out_clean, f, ensure_ascii=False, indent=2)
107
108 return json_path
```

## src/freedom_bank/batch_parse.py:185

Level Medium

Status Not processed

```
182
183 meta_json_path = ensure_pdf_meta_json(pdf_path, pdf_meta_dir)
184 try:
185     with open(meta_json_path, "r", encoding="utf-8") as f:
186         pdf_json = json.load(f)
187
188     period_end = header_df.iloc[0].get("period_end")
```

## src/halyk_business/footer.py:140

Level Medium

Status Not processed

```
137
138 if source_path.lower().endswith(".jsonl"):
139     last_obj: Dict[str, Any] = {}
140     with open(source_path, "r", encoding="utf-8") as f:
141         for line in f:
142             line = line.strip()
```

```
143         if not line:
```

## src/halyk_business/footer.py:153

Level Medium

Status Not processed

```
150         or last_obj.get("page_text")
151         or "")
152
153 with open(source_path, "r", encoding="utf-8") as f:
154     return f.read()
155
156
```

## src/halyk_business/footer.py:193

Level Medium

Status Not processed

```
190     print(f"Summary: {len(df)} rows parsed (details hidden)")
191
192 if args.out_json:
193     with open(args.out_json, "w", encoding="utf-8") as f:
194         json.dump(res, f, ensure_ascii=False, indent=2)
195     print(f" JSON saved → {args.out_json}")
196
```

## src/halyk_ind/parser.py:53

Level Medium

Status Not processed

```
50
51 def _read_pages_jsonl(jsonl_path: str) -> List[Dict[str, Any]]:
52     pages: List[Dict[str, Any]] = []
53     with open(jsonl_path, "r", encoding="utf-8") as f:
54         for line in f:
55             line = line.strip()
56             if not line:
```

## src/kaspi_pay/parser.py:40

Level Medium

Status Not processed

```
37 # ----------------------------------------------------------------------
38 def _read_pages_jsonl(path: str) -> List[Dict[str, Any]]:
39     pages: List[Dict[str, Any]] = []
40     with open(path, "r", encoding="utf-8") as f:
41         for line in f:
42             line = line.strip()
43             if not line:
```

## src/utils/convert_pdf_json_page.py:201

Level Medium

Status Not processed

```
198             xref = {"error": f"{type(e).__name__}: {e}"}
199         out["XRef"] = xref
200
201 with open(out_path, "w", encoding="utf-8") as f:
202     json.dump(out, f, ensure_ascii=False, indent=2)
203
```

```
204 print(f"Dumped to {out_path}")
```

## src/utils/convert_pdf_json_pages.py:61

Level Medium

Status Not processed

```
58      words_per_page.append(words)
59
60 # 2) Raw content streams (pikepdf)
61 with pikepdf.open(str(pdf_path)) as pdf, open(out_path, "w", encoding="utf-8") as out:
62     for i, page in enumerate(pdf.pages):
63         contents = page.get("/Contents", None)
64         raw_bytes = b""
```

## src/utils/path_security.py:95

Level Medium

Status Not processed

```
92     ValueError: If path is invalid or escapes base_dir
93 """
94 validated_path = validate_path(file_path, base_dir)
95 return open(validated_path, mode, **kwargs)
```

# SQL injection (Python)

## Description

An injection in an SQL query is possible. This can be exploited to bypass the authentication mechanism, access all database entries, or execute malicious code with application rights.
Client side code injection attacks take the first place in the “OWASP Top 10 2017” web application

vulnerabilities ranking and the seventh place in the “OWASP Mobile Top 10 2014” ranking. The level of potential damage from such an attack depends on the user input validation performance and file protection mechanisms.
SQL injections occur when a database query is based on data from an untrusted source (e.g., from a string entered by a user). In the absence of proper validation an attacker can modify the query to execute malicious SQL query.
The most common variants of SQL injection:

- Direct addition of malicious code into a string variable, based on which the SQL query is generated.
- Premature termination of the correct SQL command via the “– ” sequence of characters (interpreted as the beginning of a comment). The contents of the string after this sequence will be ignored during the execution of SQL command.
- Addition to the input string variable of the “;” character (interpreted as the end of the command), and other malicious commands (request splitting).
- Guessing the password through sequential execution of SQL queries.

## Example

Let the application substitute the user entered city name to the SQL query. For example, for the input string “London” we have:
SELECT * FROM Orders WHERE City = 'London'
An attacker can enter the following query:
London'; drop table Orders--
In this case, a query will be built that searches through the table and then deletes the table:
SELECT * FROM Orders WHERE City = 'London';drop table Orders-- '

## Recommendations

- Do not make assumptions about the type or amount of data entered by a user.
- Implement a mechanism of validation for data entered by a user.
- Escape special characters (“;”, “–”, “/*”, “*/”, “‘”; the exact list depends on the database type).
- Use stored procedures to validate user input along with the mechanism of parameters filtering.

## Links

1. OWASP Top 10 2017-A1-Injection
2. OWASP: SQL Injection
3. WASC-19: SQL Injection
4. CAPEC-66: SQL Injection
5. Understanding SQL Injection – cisco.com

6. CWE CATEGORY: OWASP Top Ten 2017 Category A1 - Injection
7. CWE-89

## Vulnerability Entries

### src/api/app.py:812

Level Medium

Status Not processed

```
809 ]
810
811 for table in tables:
812     result = db.execute_query(f"SELECT COUNT(*) as count FROM {table}")
813     summary[table] = result[0]['count'] if result else 0
814
815 db.disconnect()
```

### src/ui/upload_app.py:1191

Level Medium

Status Not processed

```
1188     'transactions': 'Транзакции'
1189 }
1190 for table in ['clients', 'accounts', 'statements', 'transactions']:
1191     cursor.execute(f"SELECT COUNT(*) FROM {table};")
1192     counts[table_names_ru[table]] = cursor.fetchone()[0]
1193
1194 cursor.close()
```

### src/ui/upload_app.py:1226

Level Medium

Status Not processed

```
1223
1224 for table in tables:
1225     try:
1226         cursor.execute(f"SELECT COUNT(*) FROM {table};")
1227         stats[table_names_ru[table]] = cursor.fetchone()[0]
1228     except:
1229         stats[table_names_ru[table]] = "Н/Д"
```

# WAF Setup Instructions

## Persistent XSS

### Description

Persistent XSS or server XSS attack is possible.
Cross-site scripting is one of the most common types of attacks on web applications. XSS attacks take seventh place in the “OWASP Top 10 2017” list of ten most significant vulnerabilities in web applications.
The main phase of any XSS attack is an imperceptible for the victim execution of a malicious code in the context of the vulnerable application. For this purpose, the functionality of the client application (browser) is used that allows to automatically execute scripts embedded in web page code. In most cases, these malicious scripts are implemented in JavaScript.
Consequences of an XSS attack vary from violations of application functionality to complete loss of user data confidentiality. The malicious code during the XSS attack can steal user HTTP-cookie, which gives an attacker the ability to make requests to the server on behalf of the user.
OWASP suggests the following classification of XSS attacks:

- Server type XSS occurs when data from an untrusted source is included in the response returned by the server. The source of such data can be both user input and server database (where it had been previously injected by an attacker who exploited vulnerabilities in the server-side application).
- Client type XSS occurs when the raw data from the user input contains code that changes the Document Object Model (DOM) of the web page received from the server. The source of such data can be both the DOM and the data received from the server (e.g., in response to an AJAX request).

Typical server type attack scenario:

1. Unvalidated data, usually from a HTTP request, gets into the server part of the application.
2. The server dynamically generates a web page that contains the unvalidated data.
3. In the process of generating a web page, server does not prevent the inclusion of an executable code that can be executed in the client (browser), such as JavaScript code language, HTML-tags, HTML-attributes, Flash, ActiveX, etc., in the page code.
4. The victim’s client application displays the web page that contains the malicious code injected via data from an untrusted source.
5. Since malicious code is injected in the web page coming from the known server, the client part of the application (browser) executes it with the rights set for the application.
6. This violates the same-origin policy, according to which the code from the one source must not get an access to resources from another source.

Client type attacks are executed in a similar way with the only difference that the malicious code is injected during the phase of the client application work with the document object model received from the server.

### Vulnerability Entries

1. src/ui/db_viewer.html:408
2. src/ui/db_viewer.html:450
3. src/ui/db_viewer.html:483

## Information leak

### Description

System configuration information leak is possible. This can help an attacker to create a plan of an attack. Debug information and error messages can be written to the log, displayed to the console, or sent to the user depending on the system settings. In some cases, an attacker can make a conclusion about the system vulnerabilities from the error message. For example, a database error can indicate insecurity against SQL injection attacks. Information about the version of the operating system, server applications and system configurations can also be of value to the attacker.

### Vulnerability Entries

1. run_all_services.py:153
2. run_all_services.py:186
3. run_all_services.py:215
4. src/bcc/footer.py:211#215
5. src/bcc_ind/parser.py:341
6. src/halyk_business/footer.py:188
7. src/halyk_ind/parser.py:319
8. src/halyk_ind/parser.py:322
9. src/halyk_ind/parser.py:326
10. src/utils/income_calc.py:296#300
11. src/utils/income_calc.py:309#313
12. src/utils/income_calc.py:322#326

## Resource injection

### Description

An attacker can gain access to reading and changing protected system resources of the application has the ability to change the resource identifier.
Resource injection occurs when an atacker can specify identifier that will be used to access the system

resource (for example, the port number to connect to the network resource). This allows him, in particular, to transfer valuable data to thied-party server.

## Vulnerability Entries

1. src/alatau_city_bank/batch_parse.py:98
2. src/alatau_city_bank/batch_parse.py:137
3. src/api/app.py:844
4. src/api/storage.py:134
5. src/api/storage.py:149
6. src/api/storage.py:157
7. src/api/storage.py:167
8. src/bcc/batch_parse.py:81
9. src/bcc/batch_parse.py:176
10. src/bcc/footer.py:189
11. src/bcc/parser.py:40
12. src/eurasian_bank/batch_parse.py:77
13. src/eurasian_bank/batch_parse.py:144
14. src/forte_bank/batch_parse.py:113
15. src/forte_bank/batch_parse.py:188
16. src/forte_bank/footer.py:194
17. src/freedom_bank/batch_parse.py:105
18. src/freedom_bank/batch_parse.py:185
19. src/halyk_business/footer.py:140
20. src/halyk_business/footer.py:153
21. src/halyk_business/footer.py:193
22. src/halyk_ind/parser.py:53
23. src/kaspi_pay/parser.py:40
24. src/utils/convert_pdf_json_page.py:201
25. src/utils/convert_pdf_json_pages.py:61
26. src/utils/path_security.py:95

# SQL injection

## Description

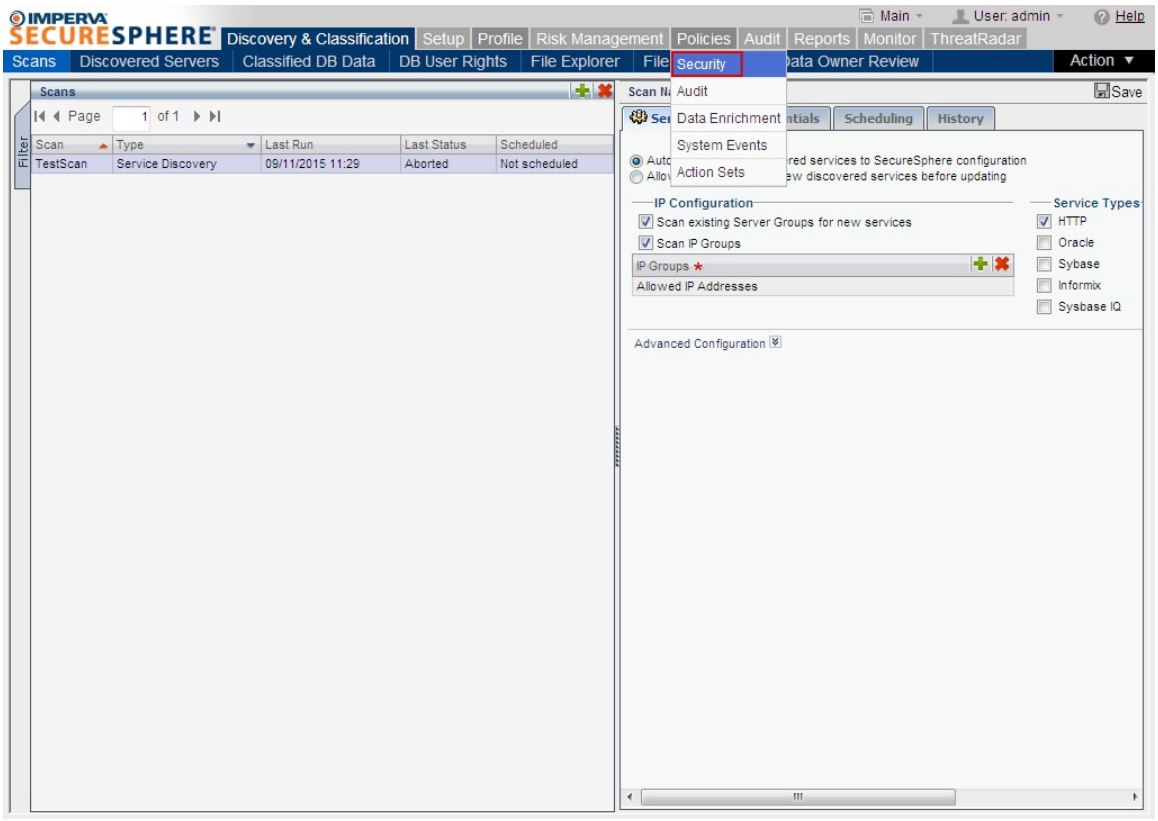
An injection in an SQL query is possible. This can be exploited to bypass the authentication mechanism, access all database entries, or execute malicious code with application rights.
Client side code injection attacks take the first place in the “OWASP Top 10 2017” web application vulnerabilities ranking and the seventh place in the “OWASP Mobile Top 10 2014” ranking. The level of potential damage from such an attack depends on the user input validation performance and file protection mechanisms.
SQL injections occur when a database query is based on data from an untrusted source (e.g., from a string entered by a user). In the absence of proper validation an attacker can modify the query to execute malicious SQL query.
The most common variants of SQL injection:

- Direct addition of malicious code into a string variable, based on which the SQL query is generated.
- Premature termination of the correct SQL command via the “– ” sequence of characters (interpreted as the beginning of a comment). The contents of the string after this sequence will be ignored during the execution of SQL command.
- Addition to the input string variable of the “;” character (interpreted as the end of the command), and other malicious commands (request splitting).
- Guessing the password through sequential execution of SQL queries.

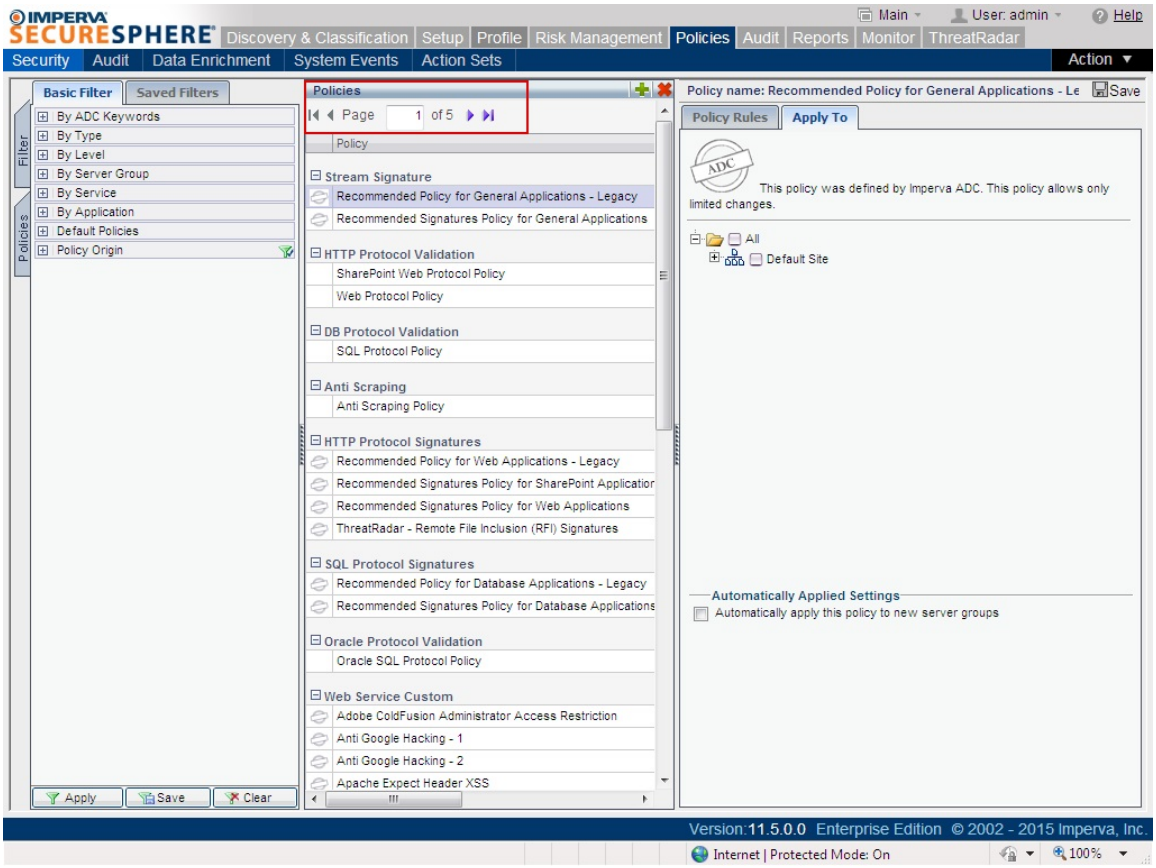
## Imperva SecureSphere

To turn on attack tracking/blocking mode in the Imperva SecureSphere for this attack type, follow these steps:

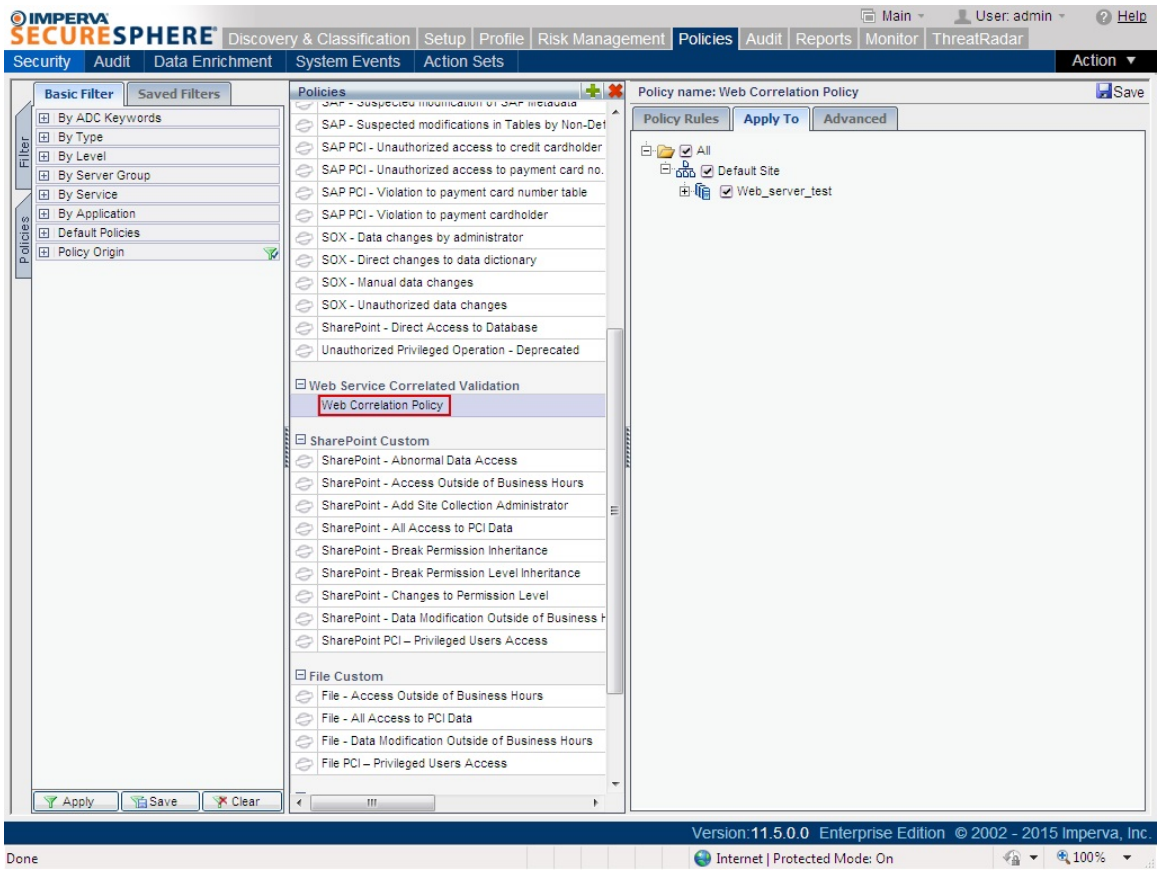
1. Go to “Policies → Security” in Imperva SecureSphere Web interface.



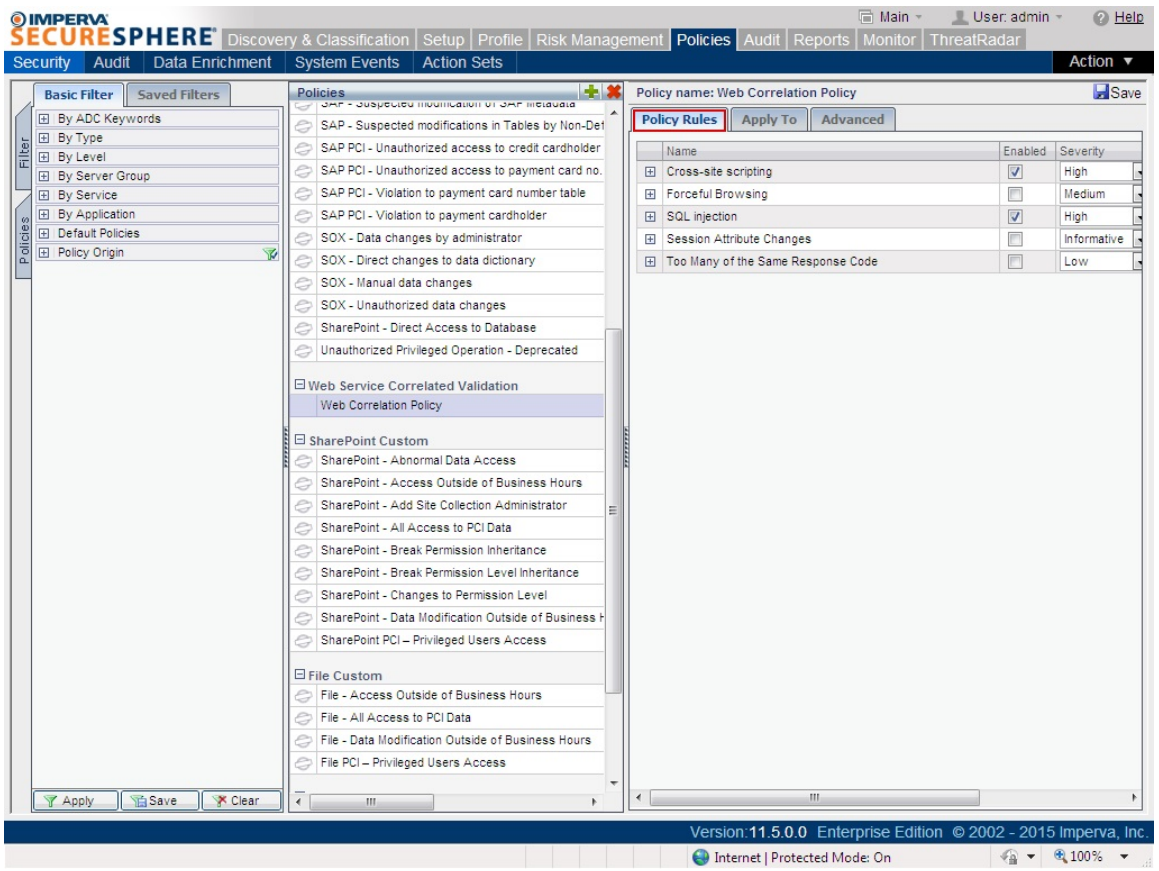

2. In this section, the **“Policies”** tab, you will see a list of policies divided into 5 pages.



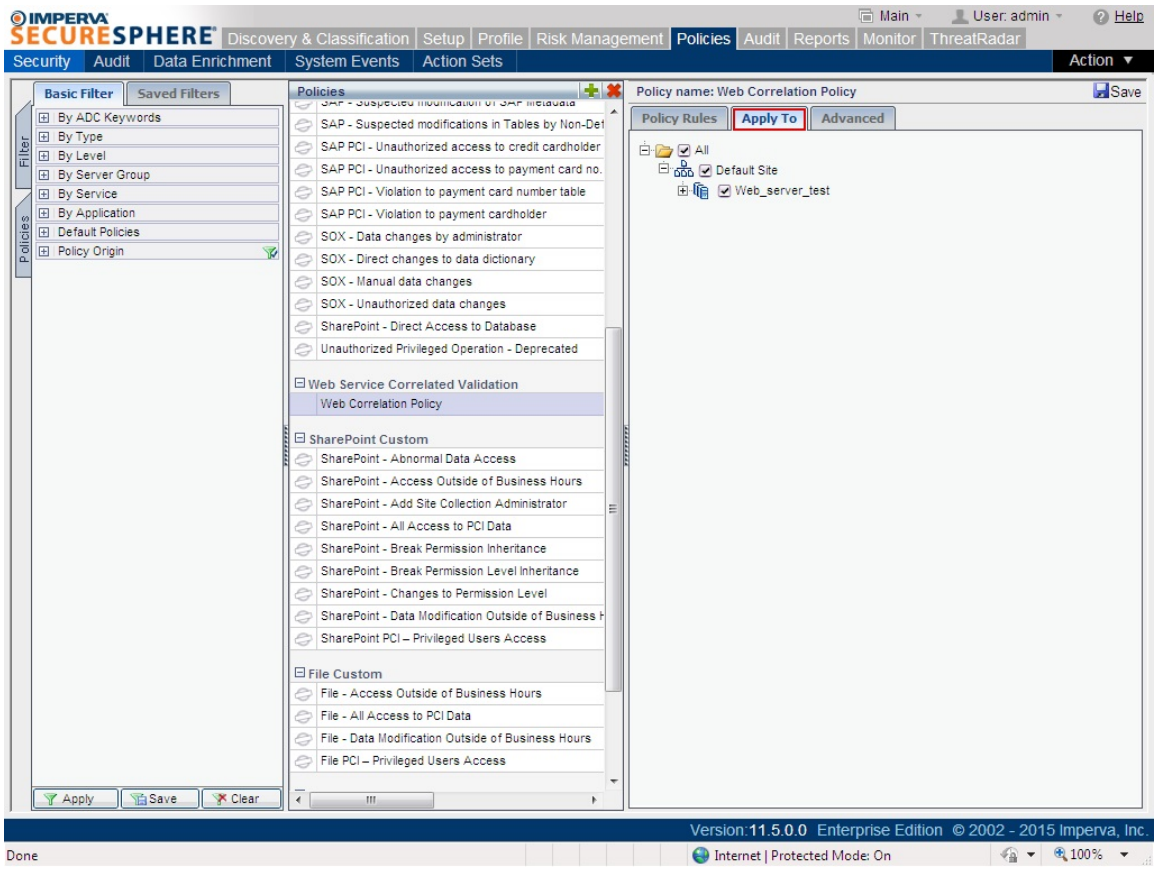

3. Find in the list and select the **“Web Correlation Policy”**. Then the **“Policy Rules”** tab or the **“Apply To”** tab will open in the right window.



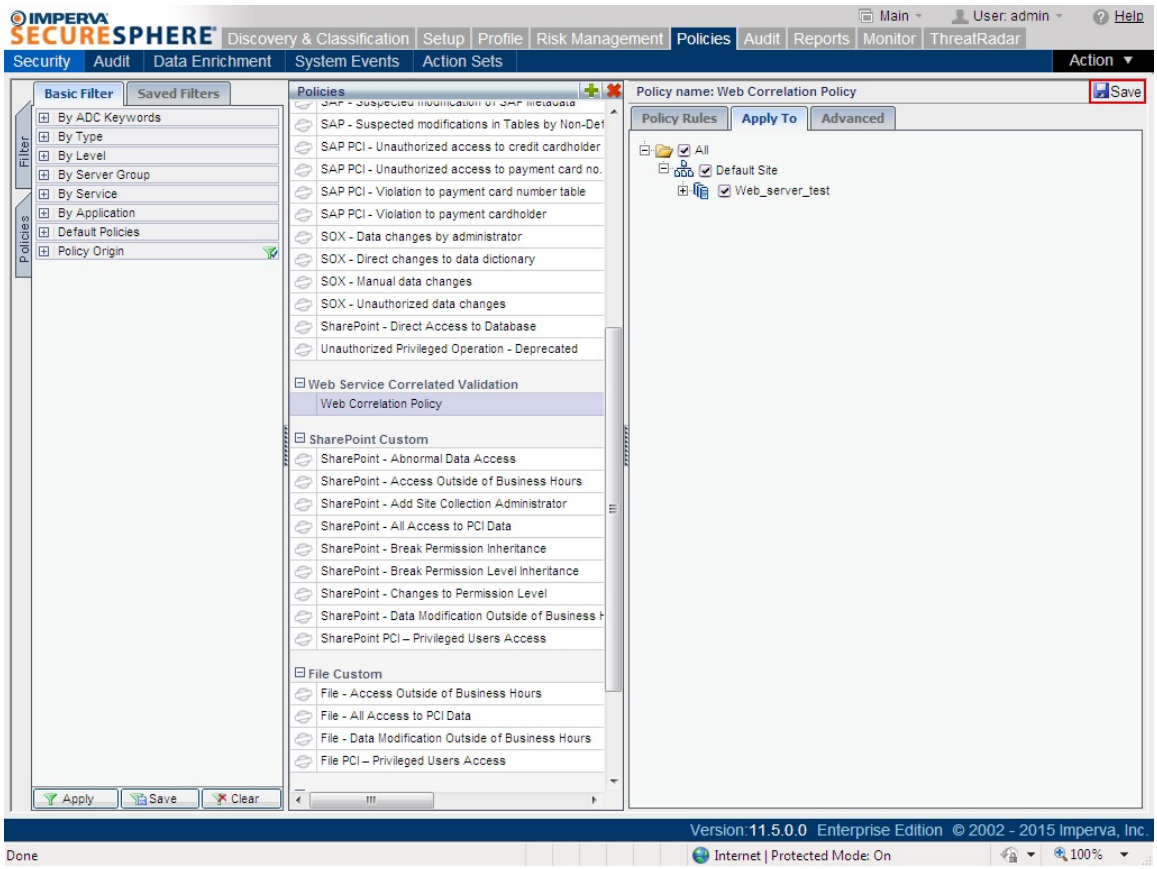

3.1. In the tab **“Policy Rules”,** you can configure actions for specific policy components.




3.2. In the **“Apply To”** tab, you can enable or disable the selected policy.




4. Click the **“Save”** button in the upper right corner to save all changes.



## ModSecurity

1. In the folder with the rules for ModSecurity (by default /usr/share/modsecurity-crs/active_rules/), create a file with the .conf extension, to which the future the new rule will be added:

```
sudo nano /usr/share/modsecurity-crs/active_rules/URL_sql.conf
```

2. Insert the rule that was created according to the recommendations for blocking this vulnerability in it.

```
SecRule REQUEST_FILENAME "URL" "id:'400001',chain,deny,log,msg:'Message to log'"
SecRule REQUEST_METHOD "GET" chain

#
# -=[ Detect SQL Comment Sequences ]=-
#
# Example Payloads Detected:
# ------------------------
# OR 1#
# DROP sampletable;--
# admin'--
# DROP/*comment*/sampletable
# DR/**/OP/*bypass blacklisting*/sampletable
# SELECT/*avoid-spaces*/password/**/FROM/**/Members
# SELECT /*!32302 1/0, */ 1 FROM tablename
# ‘ or 1=1#
# ‘ or 1=1-- -
# ‘ or 1=1/*
# ' or 1=1;\x00
# 1='1' or-- -
# ' /*!50000or*/1='1
# ' /*!or*/1='1
# 0/**/union/*!50000select*/table_name`foo`/**/
# ------------------------
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(/\*!?|\*/|[';]--|--[\s\r\n\v\f]|(?:
--[^-]*?-)|([^\-&])#.*?[\s\r\n\v\f]|;?\\x00)" "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',
accuracy:'8',id:'981231',t:none,t:urlDecodeUni,block,msg:'SQL Comment Sequence Detected.',
severity:'2',capture,logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %
{MATCHED_VAR}',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.sql_injection_score=+1,setvar:'tx.msg=%
{rule.msg}',setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"


#
# -=[ SQL Hex Evasion Methods ]=-
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
```

```
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:\A|[^\d])0x[a-f\d]{3,}
[a-f\d]*)+" "phase:2,id:'981260',rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',accuracy:'8',capture,t:
none,t:urlDecodeUni,block,msg:'SQL Hex Encoding Identified',logdata:'Matched Data: %{TX.0}
found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:'tx.msg=%{rule.
msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.
critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"


#
# -=[ String Termination/Statement Ending Injection Testing ]=-
#
# Identifies common initial SQLi probing requests where attackers insert/append
# quote characters to the existing normal payload to see how the app/db responds.
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(^[\"`´’‘;]+|[\"`´’‘;]+$)"
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:urlDecodeUni,
block,msg:'SQL Injection Attack: Common Injection Testing Detected',id:'981318',logdata:'Matched
Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:'tx.msg=%{rule.
msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.
critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"


#
# -=[ SQL Operators ]=-
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(\!
\=|\&\&|\|\||>>|<<|>=|<=|<>|<=>|xor|rlike|regexp|isnull)|(?:not\s+between\s+0\s+and)|(?:is\s+null)|
(like\s+null)|(?:(?:^|\W)in[+\s]*\([\s\d\"]+[^()]*\))|(?:xor|<>|rlike(?:\s+binary)?)|(?:regexp\s+binary))"
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:urlDecodeUni,
block,msg:'SQL Injection Attack: SQL Operator Detected',id:'981319',logdata:'Matched Data: %{TX.
0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:'tx.msg=%{rule.
msg}',setvar:tx.sql_injection_score=+%{tx.notice_anomaly_score},setvar:tx.anomaly_score=+%{tx.
notice_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"
#
# -=[ SQL Tautologies ]=-
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:([\s'\"`´’‘\(\)]*?)\b([\d\w]
++)([\s'\"`´’‘\(\)]*?)(?:(?:=|<=>|r?like|sounds\s+like|regexp)([\s'\"`´’‘\(\)]*?)\2\b|(?:!
=|<=|>=|<>|<|>|\^|is\s+not|not\s+like|not\s+regexp)([\s'\"`´’‘\(\)]*?)(?!\2)([\d\w]+)\b))" \
```

```
        "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,multiMatch,t:none,t:
urlDecodeUni,t:replaceComments,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack: SQL
Tautology Detected.',id:'950901',logdata:'Matched Data: %{TX.0} found within %
{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:'tx.msg=%{rule.
msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.
critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"
#
# -=[ Detect DB Names ]=-
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:m(?:s(?:
ysaccessobjects|ysaces|ysobjects|ysqueries|ysrelationships|ysaccessstorage|ysaccessxml|ysmodul
es|ysmodules2|db)|aster\.\.sysdatabases|ysql\.db)|s(?:ys(?:\.database_name|aux)|chema(?:\W*\
(|_name)|qlite(_temp)?_master)|d(?:atabas|b_nam)e\W*\(|information_schema|pg_(catalog|toast)
|northwind|tempdb))" \
            "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:
urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack: Common DB Names
Detected',id:'981320',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %
{MATCHED_VAR}',severity:'2',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',
tag:'WASCTC/WASC-19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',
setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-
OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
#
# SQL Keyword Anomaly Scoring
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "@pm select show top
distinct from dual where group by order having limit offset union rownum as (case" "phase:2,
id:'981300',t:none,t:urlDecodeUni,t:lowercase,nolog,pass,nolog,setvar:'tx.sqli_select_statement=%
{tx.sqli_select_statement} %{matched_var}'"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord select" "phase:2,id:'981301',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord show" "phase:2,id:'981302',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord top" "phase:2,id:'981303',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord distinct" "phase:2,id:'981304',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord from" "phase:2,id:'981305',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord dual" "phase:2,id:'981306',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord where" "phase:2,id:'981307',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@contains group by" "phase:2,id:'981308',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@contains order by" "phase:2,id:'981309',t:none,pass,
```

```
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord having" "phase:2,id:'981310',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord limit" "phase:2,id:'981311',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord offset" "phase:2,id:'981312',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@containsWord union" "phase:2,id:'981313',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@contains union all" "phase:2,id:'981314',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@contains rownum as" "phase:2,id:'981315',t:none,
pass,nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT "@contains (case" "phase:2,id:'981316',t:none,pass,
nolog,setvar:tx.sqli_select_statement_count=+1,setvar:tx.sql_injection_score=+1"
SecRule TX:SQLI_SELECT_STATEMENT_COUNT "@ge 3" "phase:2,t:none,block,id:'981317',
rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'8',accuracy:'8',msg:'SQL SELECT Statement Anomaly
Detection Alert',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %
{MATCHED_VAR}',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',setvar:tx.
anomaly_score=+%{tx.warning_anomaly_score},setvar:tx.sql_injection_score=+1,setvar:'tx.msg=%
{rule.msg}',setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"
#
# Blind SQL injection
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:\b(?:(?:s(?:ys\.(?:
user_(?:(?:t(?:ab(?:_column|le)|rigger)|object|view)s|c(?:onstraints|atalog))|all_tables|tab)|elect\b.
{0,40}\b(?:substring|users?|ascii))|m(?:sys(?:(?:queri|ac)e|relationship|column|object)s|ysql\.
(db|user))|c(?:onstraint_type|harindex)|waitfor\b\W*?\bdelay|attnotnull)\b|(?:locate|instr)\W+\()
|\@\@spid\b)|\b(?:(?:s(?:ys(?:(?:(?:process|tabl)e|filegroup|object)s|c(?:o(?:nstraint|lumn)s|at)
|dba|ibm)|ubstr(?:ing)?)|user_(?:(?:(?:constrain|objec)t|tab(?:_column|le)|ind_column|user)
s|password|group)|a(?:tt(?:rel|typ)id|ll_objects)|object_(?:(?:nam|typ)e|id)|pg_(?:attribute|class)
|column_(?:name|id)|xtype\W+\bchar|mb_users|rownum)\b|t(?:able_name\b|extpos\W+\()))" \
        "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:
urlDecodeUni,ctl:auditLogParts=+E,block,msg:'Blind SQL Injection Attack',id:'950007',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %
{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%
{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-
OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"

#
# SQL injection
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:\b(?:(?:s(?:t(?:d(?:dev
(_pop|_samp)?)?|r(?:_to_date|cmp))|u(?:b(?:str(?:ing(_index)?)?|(?:dat|tim)e)|m)|e(?:c(?:
_to_time|ond)|ssion_user)|ys(?:tem_user|date)|ha(1|2)?|oundex|chema|ig?n|pace|qrt)|i(?:s(null|_
```

(free_lock|ipv4_compat|ipv4_mapped|ipv4|ipv6|not_null|not|null|used_lock))?|n(?:et6?_(aton|ntoa)|s(?:ert|tr)|terval)?|f(null)?)|u(?:n(?:compress(?:ed_length)?|ix_timestamp|hex)|tc_(date|time|timestamp)|p(?:datexml|per)|uid(_short)?|case|ser)|l(?:o(?:ca(?:l(timestamp)?|te)|g(2|10)?|ad_file|wer)|ast(_day|_insert_id)?|e(?:(?:as|f)t|ngth)|case|trim|pad|n)|t(?:ime(stamp|stampadd|stampdiff|diff|_format|_to_sec)?|o_(base64|days|seconds|n?char)|r(?:uncate|im)|an)|m(?:a(?:ke(?:_set|date)|ster_pos_wait|x)|i(?:(?:crosecon)?d|n(?:ute)?)|o(?:nth(name)?|d)|d5)|r(?:e(?:p(?:lace|eat)|lease_lock|verse)|o(?:w_count|und)|a(?:dians|nd)|ight|trim|pad)|f(?:i(?:eld(_in_set)?|nd_in_set)|rom_(base64|days|unixtime)|o(?:und_rows|rmat)|loor)|a(?:es_(?:de|en)crypt|s(?:cii(str)?|in)|dd(?:dat|tim)e|(?:co|b)s|tan2?|vg)|p(?:o(?:sition|w(er)?)|eriod_(add|diff)|rocedure_analyse|assword|i)|b(?:i(?:t_(?:length|count|x?or|and)|n(_to_num)?)|enchmark)|e(?:x(?:p(?:ort_set)?|tract(value)?)|nc(?:rypt|ode)|lt)|v(?:a(?:r(?:_(?:sam|po)p|iance)|lues)|ersion)|g(?:r(?:oup_conca|eates)t|et_(format|lock))|o(?:(?:ld_passwo)?rd|ct(et_length)?)|we(?:ek(day|ofyear)?|ight_string)|n(?:o(?:t_in|w)|ame_const|ullif)|(rawton?)?hex(toraw)?|qu(?:arter|ote)|(pg_)?sleep|year(week)?|d?count|xmltype|hour)\W*\(|\b(?:(?:s(?:elect\b(?:.{1,100}?\b(?:(?:length|count|top)\b.{1,100}?\bfrom|from\b.{1,100}?\bwhere)|.*?\b(?:d(?:ump\b.*\bfrom|ata_type)|(?:to_(?:numbe|cha)|inst)r))|p_(?:sqlexec|sp_replwritetovarbin|sp_help|addextendedproc|is_srvrolemember|prepare|sp_password|execute(?:sql)?|makewebtask|oacreate)|ql_(?:longvarchar|variant))|xp_(?:reg(?:re(?:movemultistring|ad)|delete(?:value|key)|enum(?:value|key)s|addmultistring|write)|terminate|xp_servicecontrol|xp_ntsec_enumdomains|xp_terminate_process|e(?:xecresultset|numdsn)|availablemedia|loginconfig|cmdshell|filelist|dirtree|makecab|ntsec)|u(?:nion\b.{1,100}?\bselect|tl_(?:file|http))|d(?:b(?:a_users|ms_java)|elete\b\W*?\bfrom)|group\b.*\bby\b.{1,100}?\bhaving|open(?:rowset|owa_util|query)|load\b\W*?\bdata\b.*\binfile|(?:n?varcha|tbcreato)r|autonomous_transaction)\b|i(?:n(?:to\b\W*?\b(?:dump|out)file|sert\b\W*?\binto|ner\b\W*?\bjoin)\b|(?:f(?:\b\W*?\(\W*?\bbenchmark|null\b)|snull\b)\W*?\()|print\b\W*?\@\@|cast\b\W*?\()|c(?:(?:ur(?:rent_(?:time(?:stamp)?|date|user)|(?:dat|tim)e)|h(?:ar(?:(?:acter)?_length|set)?|r)|iel(?:ing)?|ast|r32)\W*\(|o(?:(?:n(?:v(?:ert(?:_tz)?)?|cat(?:_ws)?|nection_id)|(?:mpres)?s|ercibility|alesce|t)\W*\(|llation\W*\(a))|d(?:(?:a(?:t(?:e(?:(?:_(add|format|sub))?|diff)|abase)|y(name|ofmonth|ofweek|ofyear)?)|e(?:(?:s_(de|en)cryp|faul)t|grees|code)|ump)\W*\(|bms_\w+\.\b)|(?:;\W*?\b(?:shutdown|drop)|\@\@version)\b|\butl_inaddr\b|\bsys_context\b|'(?:s(?:qloledb|a)|msdasql|dbo)'))" \
        "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack',id:'950001',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "\b(?i:having)\b\s+(\d{1,10}|'[^=]{1,10}')\s*?[=<>]|(?i:\bexecute(\s{1,5}[\w\.$]{1,5}\s{0,3})?\()|\bhaving\b ?(?:\d{1,10}|[\'\"][^=]{1,10}[\'\"]) ?[=<>]+|(?i:\bcreate\s+?table.{0,20}?\()|(?i:\blike\W*?char\W*?\()|(?i:(?:(select(.*?)case|from(.*?)limit|order\sby)))|exists\s(\sselect|select\Sif(null)?\s\(|select\Stop|select\Sconcat|system\s\(|\b(?i:having)\b\s+(\d{1,10})|'[^=]{1,10}')" \
        "phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack',id:'959070',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.

anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-
OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:\bor\b ?(?:\d{1,10}|[\'\"]
[^=]{1,10}[\'\"]) ?[=<>]+|(?i:'\s+x?or\s+.{1,20}[+\-!<>=])|\b(?i:x?or)\b\s+(\d{1,10}|'[^=]{1,10}')|\b(?i:x?or)
\b\s+(\d{1,10}|'[^=]{1,10}')\s*?[=<>])" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:
urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack',id:'959071',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %
{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%
{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-
OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i)\b(?i:and)\b\s+(\d{1,10}
|'[^=]{1,10}')\s*?[=]|\b(?i:and)\b\s+(\d{1,10}|'[^=]{1,10}')\s*?[<>]|\band\b ?(?:\d{1,10}|[\'\"][^=]{1,10}
[\'\"]) ?[=<>]+|\b(?i:and)\b\s+(\d{1,10}|'[^=]{1,10}')" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:
urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack',id:'959072',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',
tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %
{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%
{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-
OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*|!REQUEST_HEADERS:via
"(?i:\b(?:coalesce\b|root\@))" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:
urlDecodeUni,ctl:auditLogParts=+E,block,id:'950908',msg:'SQL Injection Attack.',logdata:'Matched
Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',setvar:'tx.msg=%{rule.msg}',setvar:tx.
sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.
critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%
{matched_var_name}=%{tx.0}"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:(?:s(?:t(?:d(?:dev
(_pop|_samp)?)?|r(?:_to_date|cmp))|u(?:b(?:str(?:ing(_index)?)?|(?:dat|tim)e)|m)|e(?:c(?:
_to_time|ond)|ssion_user)|ys(?:tem_user|date)|ha(1|2)?|oundex|chema|ig?n|pace|qrt)|i(?:s(null|_
(free_lock|ipv4_compat|ipv4_mapped|ipv4|ipv6|not_null|not|null|used_lock))?|n(?:et6?_(aton|ntoa)|s
(?:ert|tr)|terval)?|f(null)?)|u(?:n(?:compress(?:ed_length)?|ix_timestamp|hex)|tc_
(date|time|timestamp)|p(?:datexml|per)|uid(_short)?|case|ser)|l(?:o(?:ca(?:l(timestamp)?|te)|g(2|10)?
|ad_file|wer)|ast(_day|_insert_id)?|e(?:(?:as|f)t|ngth)|case|trim|pad|n)|t(?:ime
(stamp|stampadd|stampdiff|diff|_format|_to_sec)?|o_(base64|days|seconds|n?char)|r(?:uncate|im)
|an)|m(?:a(?:ke(?:_set|date)|ster_pos_wait|x)|i(?:(?:crosecon)?d|n(?:ute)?)|o(?:nth(name)?|d)|d5)|r
(?:e(?:p(?:lace|eat)|lease_lock|verse)|o(?:w_count|und)|a(?:dians|nd)|ight|trim|pad)|f(?:i(?:eld
(_in_set)?|nd_in_set)|rom_(base64|days|unixtime)|o(?:und_rows|rmat)|loor)|a(?:es_(?:de|en)crypt|s
(?:cii(str)?|in)|dd(?:dat|tim)e|(?:co|b)s|tan2?|vg)|p(?:o(?:sition|w(er)?)|eriod_(add|diff)
|rocedure_analyse|assword|i)|b(?:i(?:t_(?:length|count|x?or|and)|n(_to_num)?)|enchmark)|e(?:x(?:p

(?:ort_set)?|tract(value)?)|nc(?:rypt|ode)|lt)|v(?:a(?:r(?:_(?:sam|po)p|iance)|lues)|ersion)|g(?:r(?:oup_conca|eates)t|et_(format|lock))|o(?:(?:ld_passwo)?rd|ct(et_length)?)|we(?:ek(day|ofyear)?|ight_string)|n(?:o(?:t_in|w)|ame_const|ullif)|(rawton?)?hex(toraw)?|qu(?:arter|ote)|(pg_)?sleep|year(week)?|d?count|xmltype|hour)\W*?\(|\b(?:(?:s(?:elect\b(?:.{1,100}?\b(?:(?:length|count|top)\b.{1,100}?\bfrom|from\b.{1,100}?\bwhere)|.*?\b(?:d(?:ump\b.*?\bfrom|ata_type)|(?:to_(?:numbe|cha)|inst)r))|p_(?:sqlexec|sp_replwritetovarbin|sp_help|addextendedproc|is_srvrolemember|prepare|sp_password|execute(?:sql)?|makewebtask|oacreate)|ql_(?:longvarchar|variant))|xp_(?:reg(?:re(?:movemultistring|ad)|delete(?:value|key)|enum(?:value|key)s|addmultistring|write)|terminate|xp_servicecontrol|xp_ntsec_enumdomains|xp_terminate_process|e(?:xecresultset|numdsn)|availablemedia|loginconfig|cmdshell|filelist|dirtree|makecab|ntsec)|u(?:nion\b.{1,100}?\bselect|tl_(?:file|http))|d(?:b(?:a_users|ms_java)|elete\b\W*?\bfrom)|group\b.*?\bby\b.{1,100}?\bhaving|open(?:rowset|owa_util|query)|load\b\W*?\bdata\b.*?\binfile|(?:n?varcha|tbcreato)r|autonomous_transaction)\b|i(?:n(?:to\b\W*?\b(?:dump|out)file|sert\b\W*?\binto|ner\b\W*?\bjoin)\b|(?:f(?:\b\W*?\(\W*?\bbenchmark|null\b)|snull\b)\W*?\()|print\b\W*?\@\@|cast\b\W*?\()|c(?:(?:ur(?:rent_(?:time(?:stamp)?|date|user)|(?:dat|tim)e)|h(?:ar(?:(?:acter)?_length|set)?|r)|iel(?:ing)?|ast|r32)\W*?\(|o(?:(?:n(?:v(?:ert(?:_tz)?)?|cat(?:_ws)?|nection_id)|(?:mpres)?s|ercibility|alesce|t)\W*?\(|llation\W*?\(a))|d(?:(?:a(?:t(?:e(?:(?:_(add|format|sub))?|diff)|abase)|y(name|ofmonth|ofweek|ofyear)?)|e(?:(?:s_(de|en)cryp|faul)t|grees|code)|ump)\W*?\(|bms_\w+\.\b)|(?:;\W*?\b(?:shutdown|drop)|\@\@version)\b|\butl_inaddr\b|\bsys_context\b|'(?:s(?:qloledb|a)|msdasql|dbo)'))" \
"phase:2,rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',capture,t:none,t:urlDecodeUni,ctl:auditLogParts=+E,block,msg:'SQL Injection Attack',id:'959073',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',tag:'WASCTC/WASC-19',tag:'OWASP_TOP_10/A1',tag:'OWASP_AppSensor/CIE1',tag:'PCI/6.5.2',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.msg}',setvar:tx.sql_injection_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/SQL_INJECTION-%{matched_var_name}=%{tx.0}"
#
# [ SQL Injection Character Anomaly Usage ]
#
# These rules attempted to gauge when there is an exccesive use of
# meta-characters within a single parameter payload.
#
# The most likely false positive instances will be free-form text fields.
# Adjust the the @ge operator value appropriately for your site.  Increasing
# the score will reduce false positives but may also decrease detection of
# obfuscated attack payloads.
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES "([\~\!\@\#\$\%\^\&*\(\)\-\+\=\{\}\[\]\|\:\;\"\'\´\’\‘\`\<\>].*?){8,}" "phase:2,t:none,t:urlDecodeUni,block,id:'981172',rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9',accuracy:'8',msg:'Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded',capture,logdata:'Matched Data: %{TX.1} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',setvar:tx.anomaly_score=+%{tx.warning_anomaly_score},setvar:tx.sql_injection_score=+1,setvar:'tx.msg=%{rule.msg}',setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/RESTRICTED_SQLI_CHARS-%{matched_var_name}=%{tx.0}"

SecRule ARGS_NAMES|ARGS|XML:/* "([\~\!\@\#\$\%\^\&*\(\)\-\+\=\{\}\[\]\|\:\;\"\'\´\’\‘\`\<\>].*?){4,}" "phase:2,t:none,t:urlDecodeUni,block,id:'981173',rev:'2',ver:'OWASP_CRS/2.2.9',maturity:'9', accuracy:'8',msg:'Restricted SQL Character Anomaly Detection Alert - Total # of special characters exceeded',capture,logdata:'Matched Data: %{TX.1} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',setvar:tx.anomaly_score=+%{tx.warning_anomaly_score},setvar:tx.sql_injection_score=+1,setvar:'tx.msg=%{rule.msg}',setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/RESTRICTED_SQLI_CHARS-%{matched_var_name}=%{tx.0}"

#
# -=[ PHPIDS - Converted SQLI Filters ]=-
#
# https://dev.itratos.de/projects/php-ids/repository/raw/trunk/lib/IDS/default_filter.xml
#

#
# Example Payloads Detected:
# ------------------------
# IF (SELECT * FROM login) BENCHMARK(1000000,MD5(1))
# SELECT pg_sleep(10);
# IF(SUBSTRING(Password,1,1)='2',BENCHMARK(100000,SHA1(1)),0) User,Password FROM mysql.user WHERE User = ‘root’;
# select if( user() like 'root@%', benchmark(100000,sha1('test')), 'false' );
# ------------------------
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(sleep\((\s*?)(\d*?)(\s*?)\)|benchmark\((.*?)\,(.*?)\)))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects blind sqli tests using sleep() or benchmark().',id:'981272',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

#
# Example Payloads Detected:
# ------------------------
# ' or 1=1#
# ') or ('1'='1--
# 1 OR \'1\'!=0
# aaa\' or (1)=(1) #!asd
# aaa\' OR (1) IS NOT NULL #!asd
# ' =+ '
# asd' =- (-'asd') -- -a
# aa" =+ - "0
# aa' LIKE 0 -- -a
# aa' LIKE md5(1) or '1
# asd"or-1="-1
# asd"or!1="!1

```
# asd"or!(1)="1
# asd" or ascii(1)="49
# asd' or md5(5)^'1
# \"asd" or 1="1
# ' or id= 1 having 1 #1 !
# ' or id= 2-1 having 1 #1 !
# aa'or BINARY 1= '1
# aa'like-'aa
# ------------------------
#
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?i:\d[\"`´’‘]\s+[\"`´’‘]
\s+\d)|(?:^admin\s*?[\"`´’‘]|(\/\*)+[\"`´’‘]+\s?(?:--|#|\/\*|{)?)|(?:[\"`´’‘]\s*?\b(x?or|div|like|between|and)
\b\s*?[+<>=(),-]\s*?[\d\"`´’‘])|(?:[\"`´’‘]\s*?[^\w\s]?=\s*?[\"`´’‘])|(?:[\"`´’‘]\W*?[+=]+\W*?[\"`´’‘])|(?:[\"`´’‘]
\s*?[!=|][\d\s!=+-]+.*?[\"`´’‘(].*?$)|(?:[\"`´’‘]\s*?[!=|][\d\s!=]+.*?\d+$)|(?:[\"`´’‘]\s*?like\W+[\w\"`´’‘(])|(?:
\sis\s*?0\W)|(?:where\s[\s\w\.,-]+\s=)|(?:[\"`´’‘][<>~]+[\"`´’‘]))" "phase:2,capture,t:none,t:
urlDecodeUni,block,msg:'Detects basic SQL authentication bypass attempts 1/3',id:'981244',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:
\sexec\s+xp_cmdshell)|(?:[\"`´’‘]\s*?!\s*?[\"`´’‘\w])|(?:from\W+information_schema\W)|(?:(?:(?:
current_)?user|database|schema|connection_id)\s*?\([^\)]*?)|(?:[\"`´’‘];?\s*?(?:select|union|having)
\s*?[^\s])|(?:\wiif\s*?\()|(?:exec\s+master\.)|(?:union select @)|(?:union[\w(\s]*?select)|(?:select.*?\w?
user\()|(?:into[\s+]+(?:dump|out)file\s*?[\"`´’‘]))" "phase:2,capture,t:none,t:urlDecodeUni,block,
msg:'Detects MSSQL code execution and information gathering attempts',id:'981255',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:,.*?[)\da-f\"`´’‘][\"`´’‘]
(?:[\"`´’‘].*?[\"`´’‘]|\Z|[^\"`´’‘]+))|(?:\Wselect.+\W*?from)|((?:
select|create|rename|truncate|load|alter|delete|update|insert|desc)\s*?\(\s*?space\s*?\())" "phase:2,
capture,t:none,t:urlDecodeUni,block,msg:'Detects MySQL comment-/space-obfuscated injections
and backtick termination',id:'981257',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',
logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',
severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-
OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:@.+=\s*?\(\s*?
select)|(?:\d+\s*?(x?or|div|like|between|and)\s*?\d+\s*?[\-+])|(?:\/\w+;?\s+(?:having|and|x?
or|div|like|between|and|select)\W)|(?:\d\s+group\s+by.+\()|(?:(?:;|#|--)\s*?(?:drop|alter))|(?:(?:;|#|--)
\s*?(?:update|insert)\s*?\w{2,})|(?:[^\w]SET\s*?@\w+)|(?:(?:(?:n?and|x?x?or|div|like|between|and|not
|\|\||\&\&)[\s(]+\w+[\s)]*?[!=+]+[\s\d]*?[\"`´’‘=()]))" "phase:2,capture,t:none,t:urlDecodeUni,block,
msg:'Detects chained SQL injection attempts 1/2',id:'981248',
```

tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:^(-0000023456|4294967295|4294967296|2147483648|2147483647|0000012345|-2147483648|-2147483649|0000023456|2.2.90738585072007e-308|1e309)$))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Looking for integer overflow attacks, these are taken from skipfish, except 2.2.90738585072007e-308 is the \"magic number\" crash',id:'981277',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:(select|;)\s+(?:benchmark|if|sleep)\s*?\(\s*?\(?\s*?\w+))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects SQL benchmark and sleep injection attempts including conditional queries',id:'981250',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:[\s()]case\s*?\()|(?:\)\s*?like\s*?\()|(?:having\s*?[^\s]+\s*?[^\w\s])|(?:if\s?\([\d\w]\s*?[=<>~]))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects conditional SQL injection attempts',id:'981241',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:alter\s*?\w+.*?character\s+set\s+\w+)|([\"'`´’‘];\s*?waitfor\s+time\s+[\"'`´’‘])|(?:[\"'`´’‘];.*?:\s*?goto))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects MySQL charset switch and MSSQL DoS attempts',id:'981252',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:merge.*?using\s*?\()|(execute\s*?immediate\s*?[\"'`´’‘])|(?:\W+\d*?\s*?having\s*?[^\s\-])|(?:match\s*?[\w(),+-]+\s*?against\s*?\())" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects MATCH AGAINST, MERGE, EXECUTE IMMEDIATE and HAVING injections',id:'981256',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.

msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:union\s*?(?:
all|distinct|[(!@]*?)?\s*?[([]*?\s*?select\s+)|(?:\w+\s+like\s+[\"'`´’‘])|(?:like\s*?[\"'`´’‘]\%)|(?:[\"'`´’‘]\s*?
like\W*?[\"'`´’‘\d])|(?:[\"'`´’‘]\s*?(?:n?and|x?or|div|like|between|and|not |\|\||\&\&)\s+[\s\w]+=\s*?
\w+\s*?having\s+)|(?:[\"'`´’‘]\s*?\*\s*?\w+\W+[\"'`´’‘])|(?:[\"'`´’‘]\s*?[^?\w\s=.,;)(]+\s*?[(@\"'`´’‘]*?\s*?
\w+\W+\w)|(?:select\s+?[\[\]()\s\w\.,\"'`´’‘-]+from\s+)|(?:find_in_set\s*?\())" "phase:2,capture,t:none,t:
urlDecodeUni,block,msg:'Detects basic SQL authentication bypass attempts 2/3',id:'981245',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:(union(.*?)select(.*?)
from)))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Looking for basic sql injection. Common
attack string for mysql, oracle and others.',id:'981276',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:select\s*?pg_sleep)|
(?:waitfor\s*?delay\s?[\"'`´’‘]+\s?\d)|(?:;\s*?shutdown\s*?(?:;|--|#|\/*|{}))" "phase:2,capture,t:none,t:
urlDecodeUni,block,msg:'Detects Postgres pg_sleep injection, waitfor delay attacks and database
shutdown attempts',id:'981254',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',
logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',
severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-
OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:\[\$(?:ne|eq|lte?|gte?
|n?in|mod|all|size|exists|type|slice|x?or|div|like|between|and)\]))" "phase:2,capture,t:none,t:
urlDecodeUni,block,msg:'Finds basic MongoDB SQL injection attempts',id:'981270',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:\)\s*?when\s*?
\d+\s*?then)|(?:[\"'`´’‘]\s*?(?:#|--|{}))|(?:\/*!\s?\d+)|(?:ch(?:a)?r\s*?\(\s*?\d)|(?:(?:(n?and|x?x?
or|div|like|between|and|not)\s+|\|\||\&\&)\s*?\w+\())" "phase:2,capture,t:none,t:urlDecodeUni,block,
msg:'Detects MySQL comments, conditions and ch(a)r injections',id:'981240',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},

setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:[\"'`´’‘]\s+and\s*?=\W)|(?:\(\s*?select\s*?\w+\s*?\()|(?:\*\/from)|(?:\+\s*?\d+\s*?\+\s*?@)|(?:\w[\"'`´’‘]\s*?(?:[-+=|@]+\s*?)+[\d(])|(?:coalesce\s*?\(|@@\w+\s*?[^\w\s])|(?:\W!+[\"'`´’‘]\w)|(?:[\"'`´’‘];\s*?(?:if|while|begin))|(?:[\"'`´’‘][\s\d]+=\s*?\d)|(?:order\s+by\s+if\w*?\s*?\()|(?:[\s(]+case\d*?\W.+[tw]hen[\s(]))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects chained SQL injection attempts 2/2',id:'981249',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"
SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:procedure\s+analyse\s*?\()|(?:;\s*?(declare|open)\s+[\w-]+)|(?:create\s+(procedure|function)\s*?\w+\s*?\(\s*?\)\s*?-)|(?:declare[^\w]+[@#]\s*?\w+)|(exec\s*?\(\s*?@))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects MySQL and PostgreSQL stored procedure/function injections',id:'981253',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:[\"'`´’‘]\s*?(x?or|div|like|between|and)\s*?[\"'`´’‘]?\d)|(?:\\\x(?:23|27|3d))|(?:^.?[\"'`´’‘]$)|(?:(?:^[\"'`´’‘\\\]*?(?:[\d\"'`´’‘]+|[^\"'`´’‘]+[\"'`´’‘]))+\s*?(?:n?and|x?x?or|div|like|between|and|not|\|\||\&\&)\s*?[\w\"'`´’‘][+&!@(),.-])|(?:[^\w\s]\w+\s*?[|-]\s*?[\"'`´’‘]\s*?\w)|(?:@\w+\s+(and|x?or|div|like|between|and)\s*?[\"'`´’‘\d]+)|(?:@[\w-]+\s(and|x?or|div|like|between|and)\s*?[^\w\s])|(?:[^\w\s:]\s*?\d\W+[^\w\s]\s*?[\"'`´’‘].)|(?:\Winformation_schema|table_name\W))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects classic SQL injection probings 1/2',id:'981242',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:in\s*?\(+\s*?select)|(?:(?:n?and|x?x?or|div|like|between|and|not |\|\||\&\&)\s+[\s\w+]+(?:regexp\s*?\(|sounds\s+like\s*?[\"'`´’‘]|[=\d]+x))|([\"'`´’‘]\s*?\d\s*?(?:--|#))|(?:[\"'`´’‘][\%&<>^=]+\d\s*?(=|x?or|div|like|between|and))|(?:[\"'`´’‘]\W+[\w+-]+\s*?=\s*?\d\W+[\"'`´’‘])|(?:[\"'`´’‘]\s*?is\s*?\d.+[\"'`´’‘]?\w)|(?:[\"'`´’‘]\|?[\w-]{3,}[^\w\s.,]+[\"'`´’‘])|(?:[\"'`´’‘]\s*?is\s*?[\d.]+\s*?\W.*?[\"'`´’‘]))" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects basic SQL authentication bypass attempts 3/3',id:'981246',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:create\s+function\s+\w+\s+returns)|(?:;\s*?(?:

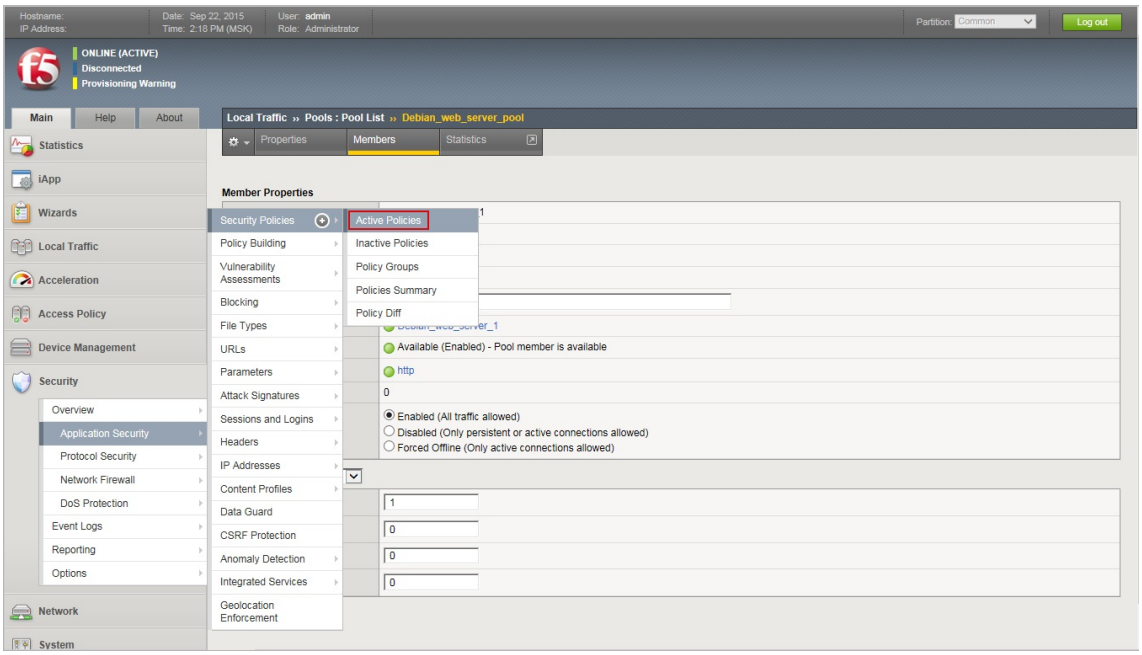
```
select|create|rename|truncate|load|alter|delete|update|insert|desc)\s*?[\[(]?\w{2,}))" "phase:2,
capture,t:none,t:urlDecodeUni,block,msg:'Detects MySQL UDF injection and other data/structure
manipulation attempts',id:'981251',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',
logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',
severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.
anomaly_score=+%{tx.critical_anomaly_score},setvar:'tx.%{tx.msg}-
OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:[\d\W]\s+as\s*?[\"'`´’
‘\w]+\s*?from)|(?:^[\W\d]+\s*?(?:
union|select|create|rename|truncate|load|alter|delete|update|insert|desc))|(?:(?:
select|create|rename|truncate|load|alter|delete|update|insert|desc)\s+(?:(?:group_)
concat|char|load_file)\s?\(?)|(?:end\s*?\);)|([\"'`´’‘]\s+regexp\W)|(?:[\s(]load_file\s*?\())" "phase:2,
capture,t:none,t:urlDecodeUni,block,msg:'Detects concatenated basic SQL injection and SQLLFI
attempts',id:'981247',tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data:
%{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.
msg=%{rule.id}-%{rule.msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.
critical_anomaly_score},setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%
{matched_var_name}=%{tx.0}'"

SecRule REQUEST_COOKIES|!REQUEST_COOKIES:/__utm/|!REQUEST_COOKIES:
/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "(?i:(?:[\"'`´’‘]\s*?\*.+(?:x?
or|div|like|between|and|id)\W*?[\"'`´’‘]\d)|(?:\^[\"'`´’‘])|(?:^[\w\s\"'`´’‘-]+(?<=and\s)(?
<=or|xor|div|like|between|and\s)(?<=xor\s)(?<=nand\s)(?<=not\s)(?<=\|\|)(?<=\&\&)\w+\()|(?:[\"'`´’‘]
[\s\d]*?[^\w\s]+\W*?\d\W*?.*?[\"'`´’‘\d])|(?:[\"'`´’‘]\s*?[^\w\s?]+\s*?[^\w\s]+\s*?[\"'`´’‘])|(?:[\"'`´’‘]\s*?[^\w\s]
+\s*?[\W\d].*?(?:#|--))|(?:[\"'`´’‘].*?\*\s*?\d)|(?:[\"'`´’‘]\s*?(x?or|div|like|between|and)\s[^\d]+[\w-]+.*?\d)|
(?:[()\*<>%+-][\w-]+[^\w\s]+[\"'`´’‘][^,])" "phase:2,capture,t:none,t:urlDecodeUni,block,msg:'Detects
classic SQL injection probings 2/2',id:'981243',
tag:'OWASP_CRS/WEB_ATTACK/SQL_INJECTION',logdata:'Matched Data: %{TX.0} found within
%{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:'tx.msg=%{rule.id}-%{rule.
msg}',setvar:tx.sql_injection_score=+1,setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},
setvar:'tx.%{tx.msg}-OWASP_CRS/WEB_ATTACK/SQLI-%{matched_var_name}=%{tx.0}'"
```

3 Restart your Web server or filter.

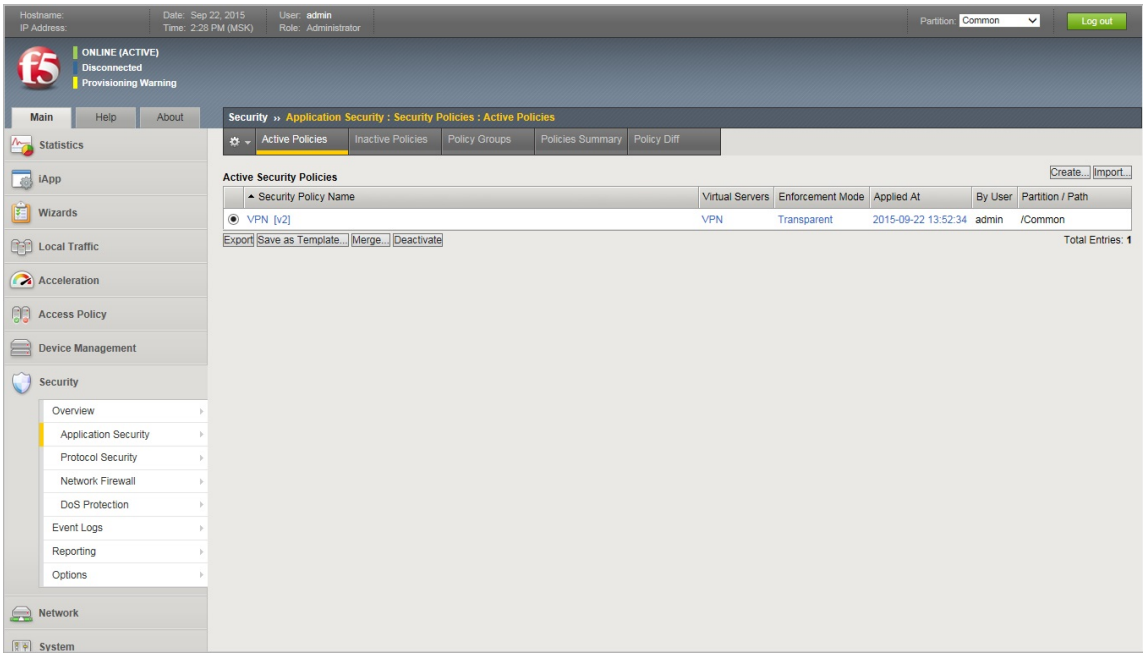
```
sudo service apache reload
```

## F5

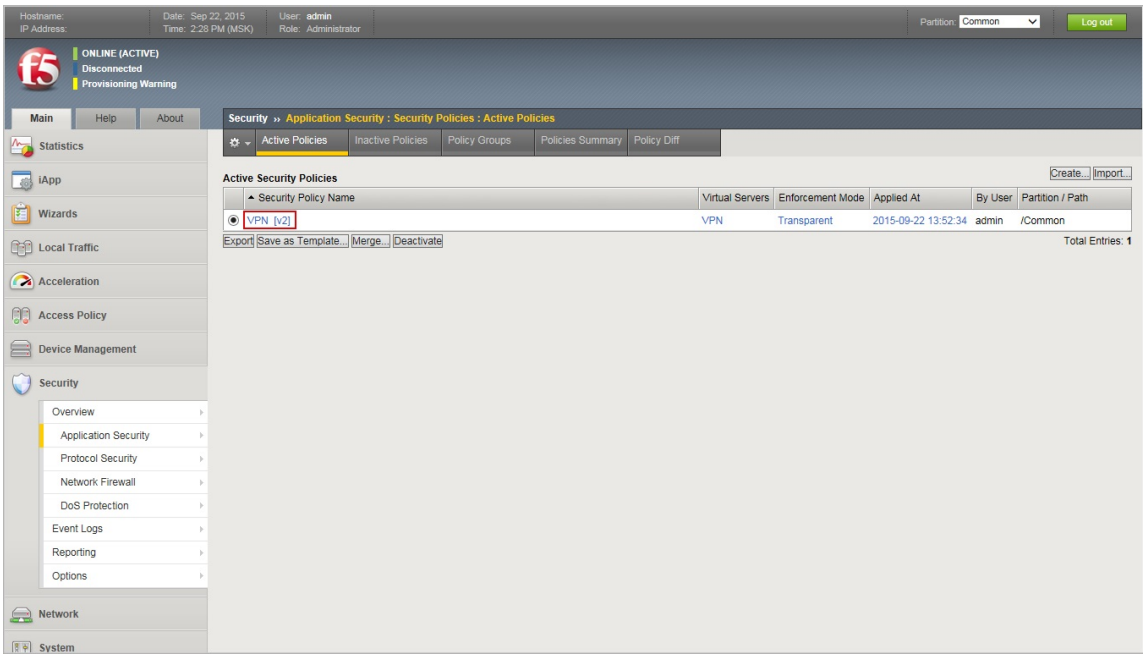
To turn on attack tracking/blocking mode:
1. Configure the resource that is to be protected in the LTM module as a virtual server and apply the ASM module policy to it.
2. Go to “Security → Application Security → Security Policies → Active Policies” in F5 BigIP Web interface.



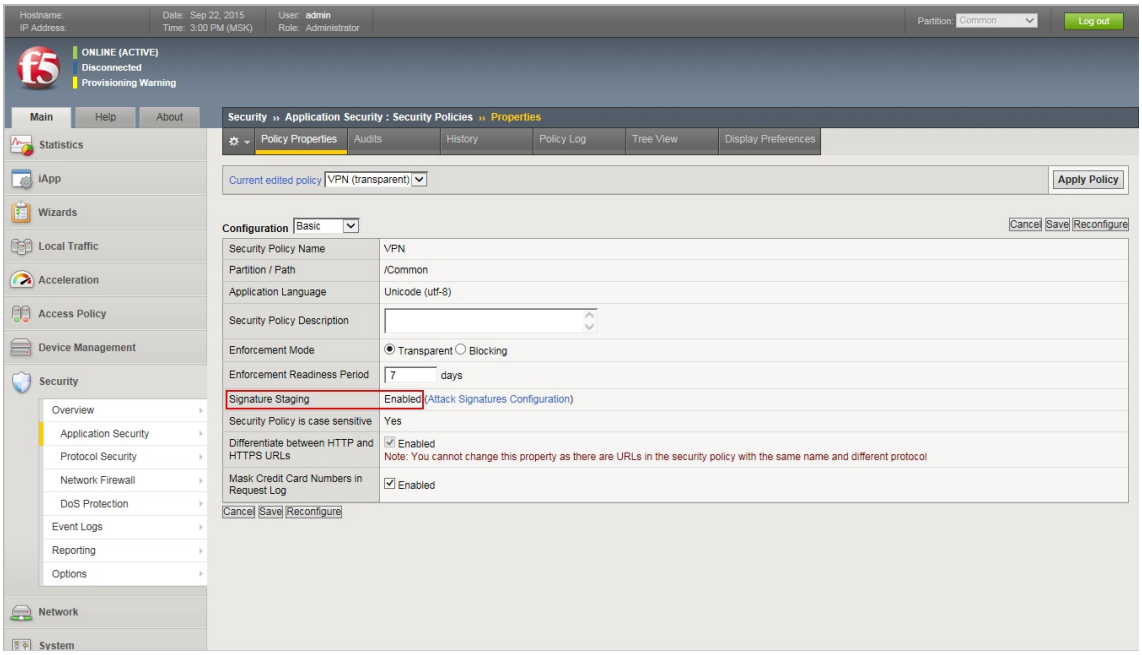

2. In this section, you will see the ASM policies.



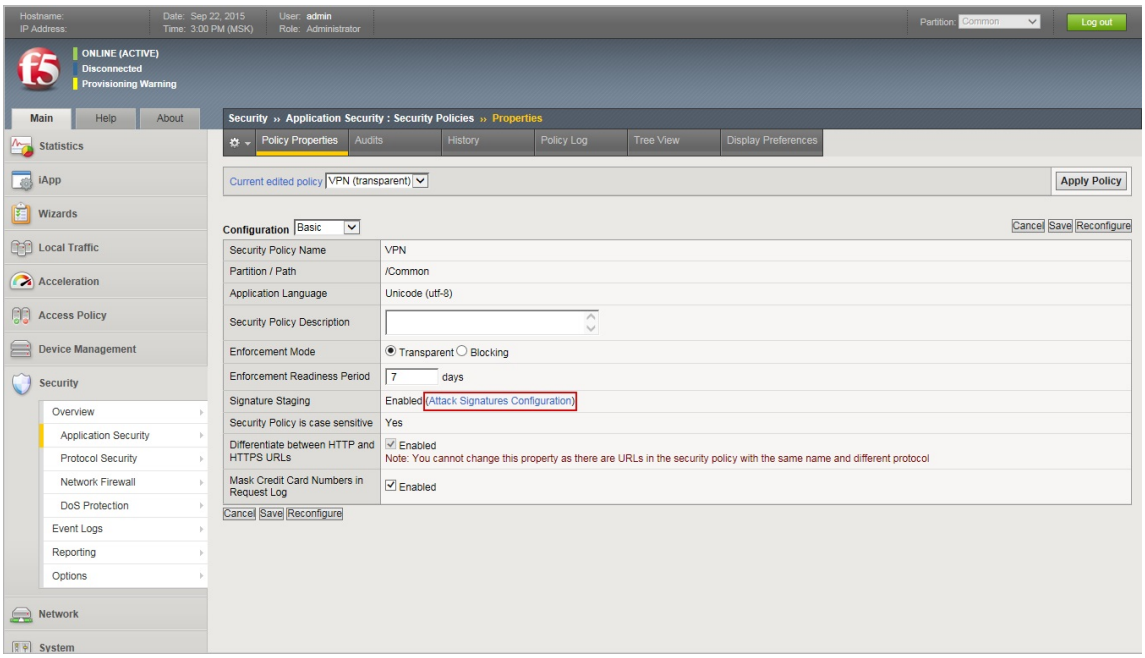

2. If the ASM policy **is already enabled** for the server:
2. Click on server name.



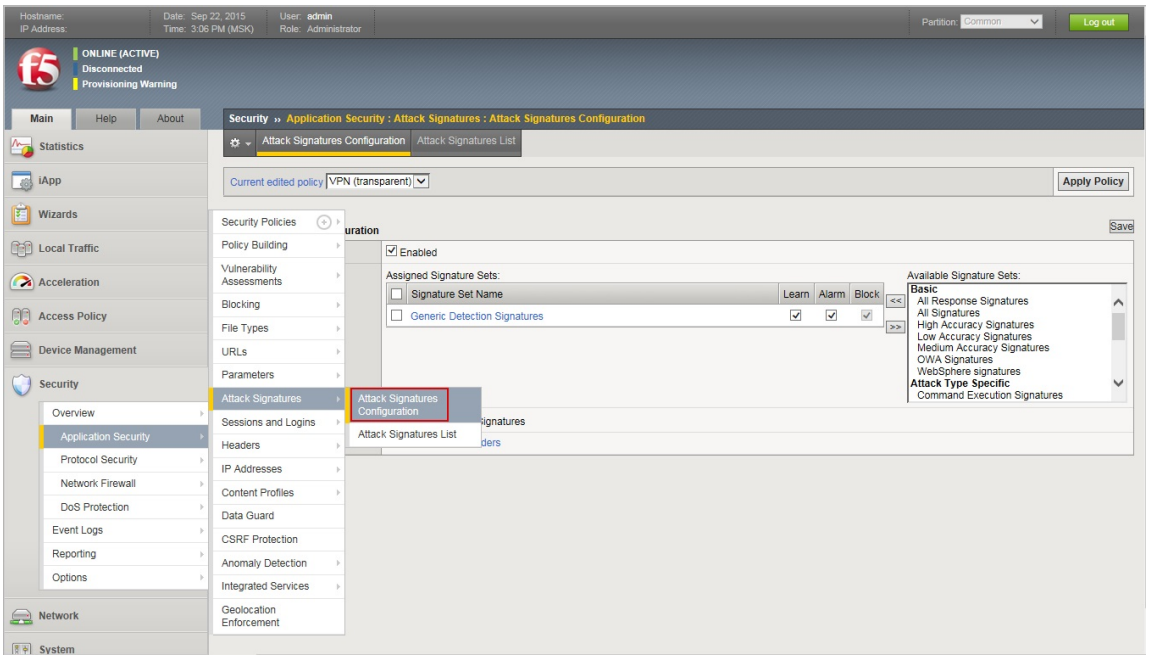

2. Window opens to reveal the ***Signature Staging Enabled (Attack Signatures Configuration)*** field.

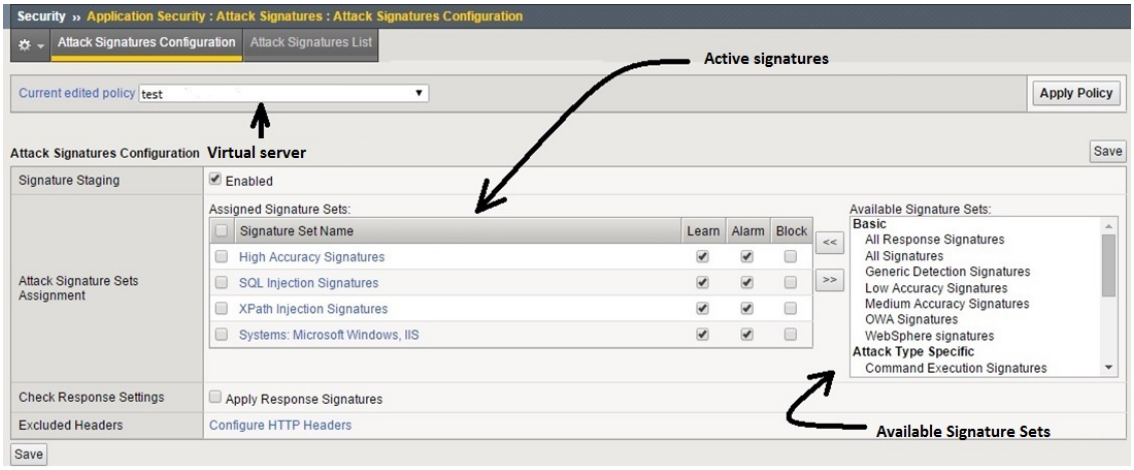



2. Click on the ***(Attack Signatures Configuration)*** link to navigate to signature configuration mode.



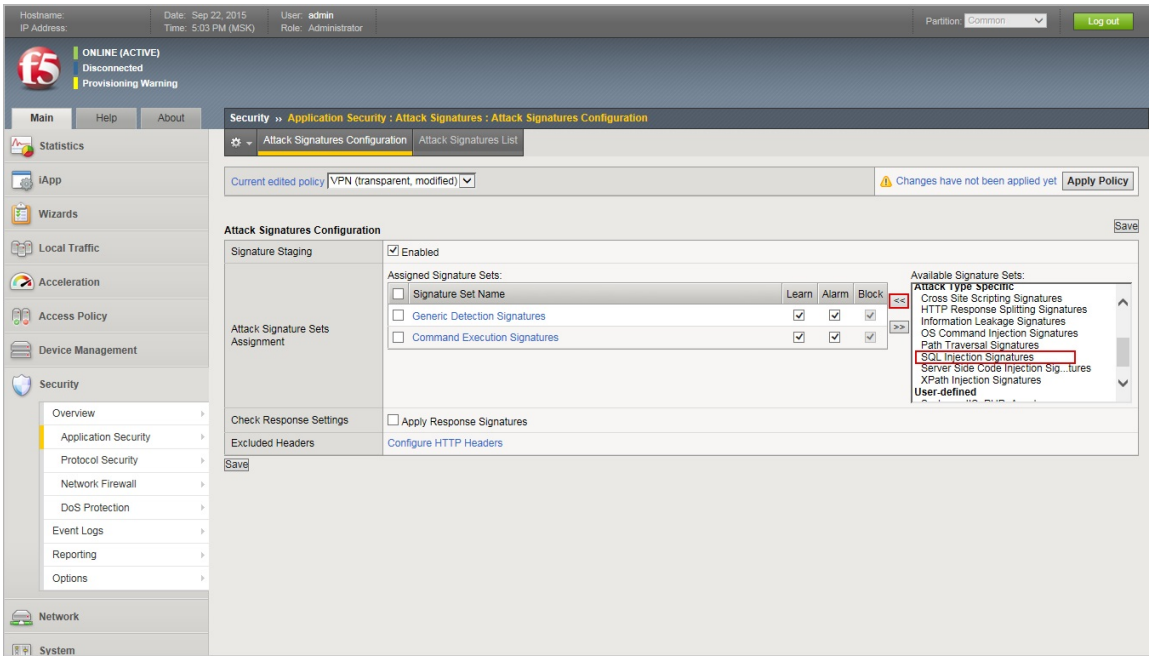

Other option: click on the “Security → Application Security → Attack Signatures → Attack Signatures Configuration” link and select the needed virtual server in the upper left corner.



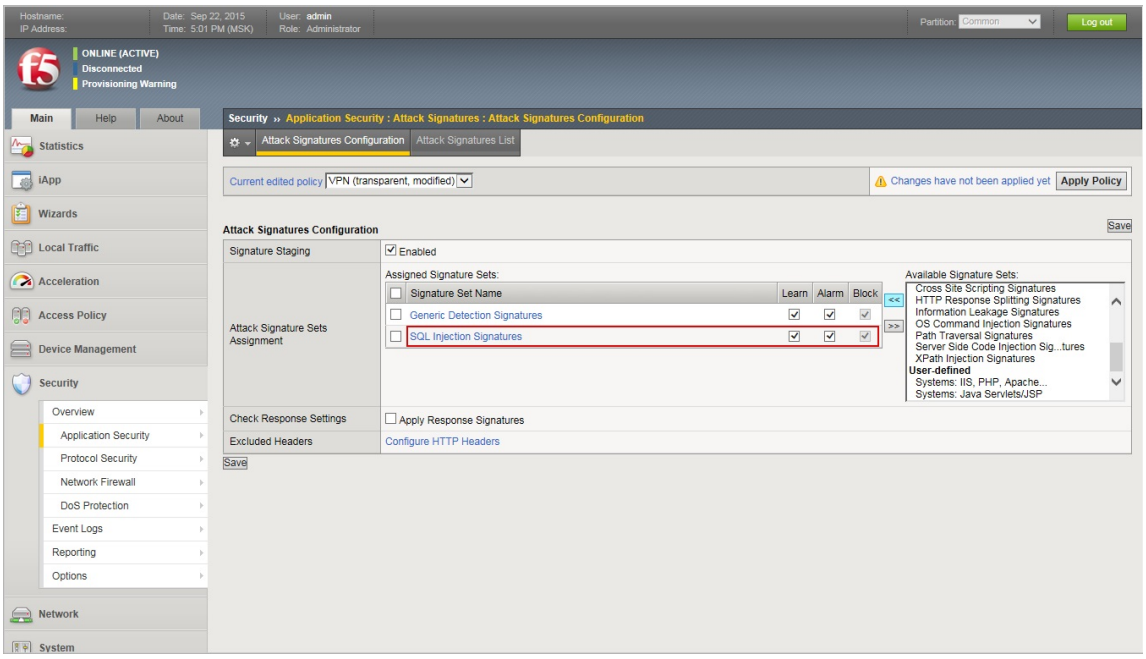



In this section, you can add the necessary sets of signatures.

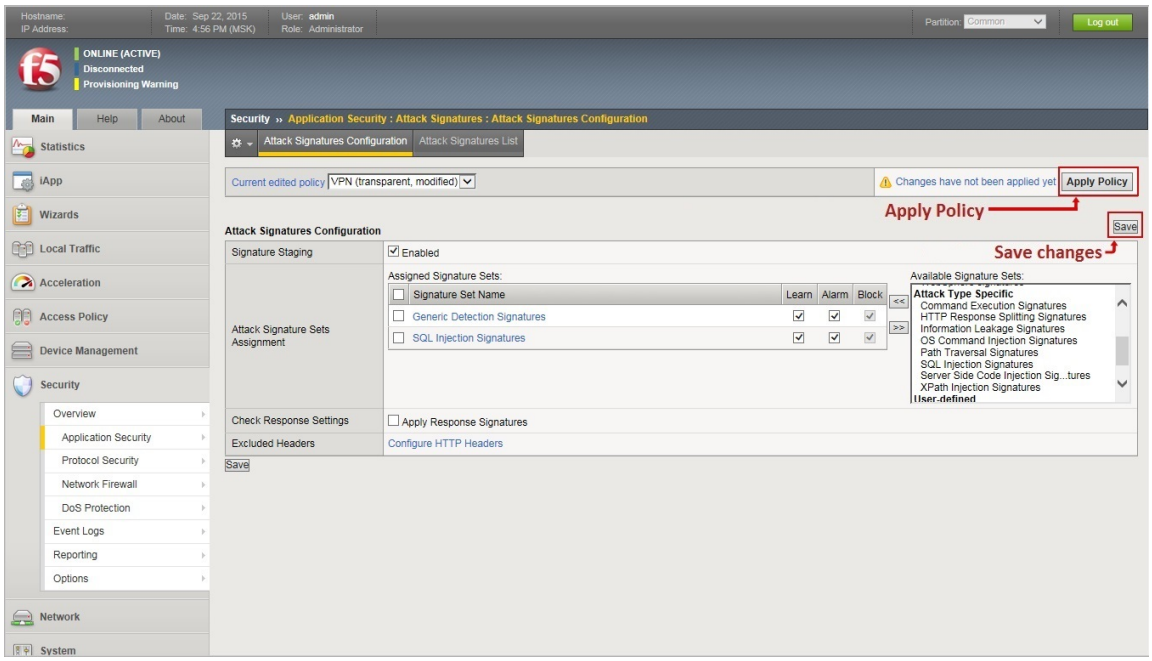
2. If ASM policy is **not available** for the desired server, then in the upper right corner click on **“Create”** and set up the configuration by answering questions of the system. During setup, the system will ask about the protected resource and will create a primary set of signatures most suitable for this type of server.
2. Connecting **SQL Injection Signatures**.
2. Select the needed set of signatures on the right and click on the **“<<”** to copy it to Assigned Signature Sets.



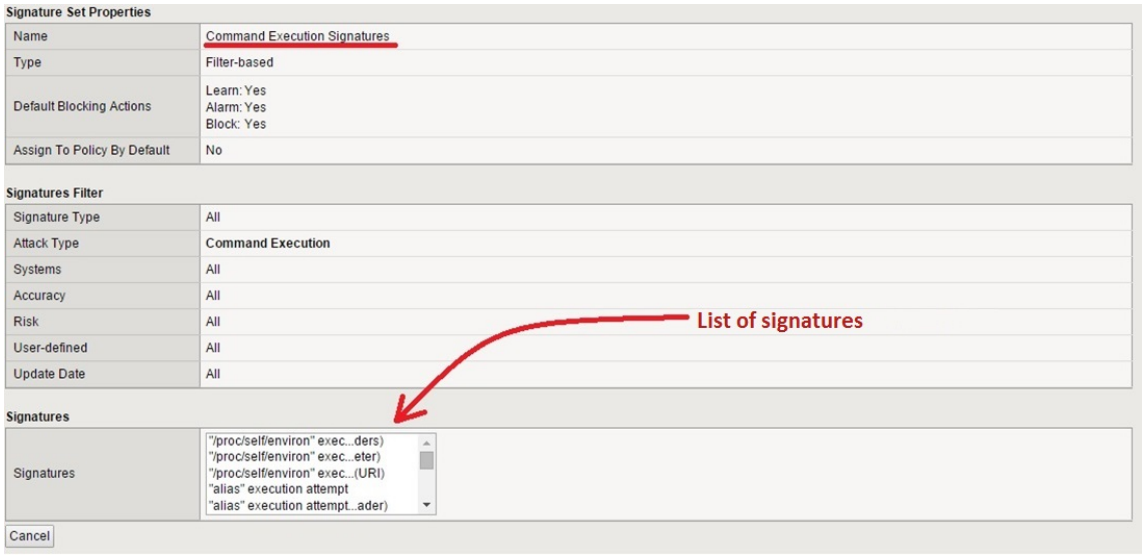

After that set will be displayed among Assigned Signature Sets:




3. After setting up the necessary sets for a virtual server, press the **“Save”** button to save the changes and click **“Apply Policy”** to apply the settings to traffic processing. Without clicking the “Apply Policy”, the policy will work with the previous set of signatures.




If you want to see the entire list of signatures of the set, click on the set. In the lower left window, all monitored parameters are displayed.




## Vulnerability Entries

1. src/api/app.py:812

2. src/ui/upload_app.py:1191

3. src/ui/upload_app.py:1226

# Scan Settings

**3/3 2026-02-11 07:01:59**

Select files for Analysis

**/*

Branch in Repository

master

## Languages

- [ ] ABAP
- [ ] Apex
- [x] C#
- [ ] C/C++
- [ ] COBOL
- [x] Config files
- [x] Dart
- [x] Go
- [ ] Groovy
- [x] HTML5
- [x] Java, Scala, Kotlin
- [x] JavaScript, TypeScript
- [ ] LotusScript
- [ ] Objective-C
- [ ] Delphi, Pascal
- [x] PHP
- [x] PL/SQL, T-SQL
- [x] Python
- [ ] Perl
- [x] Ruby
- [x] Rust
- [x] Solidity
- [x] Swift
- [ ] TSX
- [ ] VB.NET, VBA, VBScript, VB6
- [ ] Vyper
- [ ] 1C

## Java/Scala/Kotlin Specific Settings

- (●) Prebuilt project with .class files
- ( ) Source code (will be built by DerScanner)
- ( ) Source code (no build needed, only for Java)
- ( ) Source code (build with own tools)

## JavaScript Specific Settings

- [ ] Analyze standard libraries

## General Analysis Settings

- [ ] Analyze libraries and nested archives
- [ ] Use extra rules
- [ ] Incremental analysis
- [x] Preprocessing

| | |
|---|---|
| Source Code Charset | UTF-8 |
| Filename Charset | UTF-8 |

Rule Sets —

# 03 Export Settings

## Project Information

### Project Statistics

- [x] Dynamics by rating
- [x] Dynamics by vulnerabilities

### Scan History

- ( ) Do not export scan history
- (•) Export entire scan history
- ( ) Export the latest scans ...

## Vulnerability Classification

By severity

## Scan Information

- [x] Detected vulnerabilities chart
- [x] Vulnerable Dependencies
- [x] Statistics on languages
- [ ] Analyzed files statistics
- [x] Error information
- [x] Scan settings
- [x] Version Properties

## Vulnerability Filter

### Severity

- [x] Critical
- [x] Medium
- [ ] Low
- [ ] Info

### Vulnerable Dependencies

- [x] In standard libraries

- [x] In .class files that cannot be decompiled
- [x] Vulnerabilities without WAF configuration guide

### Languages

- [x] ABAP
- [x] Android
- [x] Apex
- [x] C#
- [x] C/C++
- [x] COBOL
- [x] Config files
- [x] Dart
- [x] Delphi
- [x] Go
- [x] Groovy
- [x] HTML5
- [x] Java
- [x] JavaScript
- [x] Kotlin
- [x] LotusScript
- [x] Objective-C
- [x] Pascal
- [x] PHP
- [x] PL/SQL
- [x] Python
- [x] Perl
- [x] Ruby
- [x] Rust
- [x] Scala
- [x] Solidity
- [x] Swift
- [x] T-SQL
- [x] TSX
- [x] TypeScript
- [x] VB.NET
- [x] VBA
- [x] VBScript
- [x] Visual Basic 6
- [x] Vyper
- [x] 1C

## Vulnerability List

### Vulnerability Statuses

- [x] Not processed
- [x] Confirmed
- [ ] Rejected

### List of Vulnerability Entries

- ( ) Do not export
- (•) Export all entries
- ( ) Export no more than entries ...

## Analysis Results

### Vulnerability Statuses

- [x] Not processed

- [x] Confirmed
- [ ] Rejected

### Vulnerability Entries

- ( ) Do not export
- (•) Export all entries
- ( ) Export no more than entries ...

### Source Code

- ( ) Do not export source code
- ( ) Export entire vulnerable source code file
- (•) Export context in the number of lines of code 3

### Trace

- ( ) Do not export trace items
- ( ) Export all items
- (•) Export only the first and last items

### Additional information

- [x] Vulnerability comment
- [x] Vulnerability actions

## WAF Setup Instructions

### Guide for Vulnerability Statuses

- [x] Not processed
- [x] Confirmed
- [ ] Rejected

### Guide for WAF

- [x] Imperva SecureSphere
- [x] ModSecurity
- [x] F5

## General Report Settings

- [x] Table of contents
- [x] Report export settings
- [x] Display vulnerability statuses
- [x] Display results correlation tags