Analysis Results

bankparserui.zip

Report Date
2026-02-20 07:44:28
Report Author
dctuser
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

bankparserui.zip

UUID

e3ad2fd3-5043-42fa-8210-688009e0ccb8

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	
1/1	2026-02-20 07:35:57	completed	Config files, Python, JavaScript, PL/SQL, T-SQL, HTML5	37 722	2	29	9	21	61	3.1/5.0

02

Scan Information

1/1 2026-02-20 07:35:57
10.127506

Scan Statistics

Status
completed

Score
3.1/5.0

Duration
0:04:04

Lines of Code
37 722

Vulnerabilities

2	29	9	21
Critical	Medium	Low	Info

61
Total

Language	Status	Duration	Lines of Code	Number of Vulnerabilities				
				Critical	Medium	Low	Info	Total
Config files	completed	0:00:02	22 706	2	2	2	0	6
Python	completed	0:03:55	13 952	0	27	5	21	53
JavaScript	completed	0:00:03	428	0	0	2	0	2
PL/SQL	completed	0:00:01	285	0	0	0	0	0
T-SQL	completed	0:00:01	284	0	0	0	0	0
HTML5	completed	0:00:00	67	0	0	0	0	0

Language Statistics

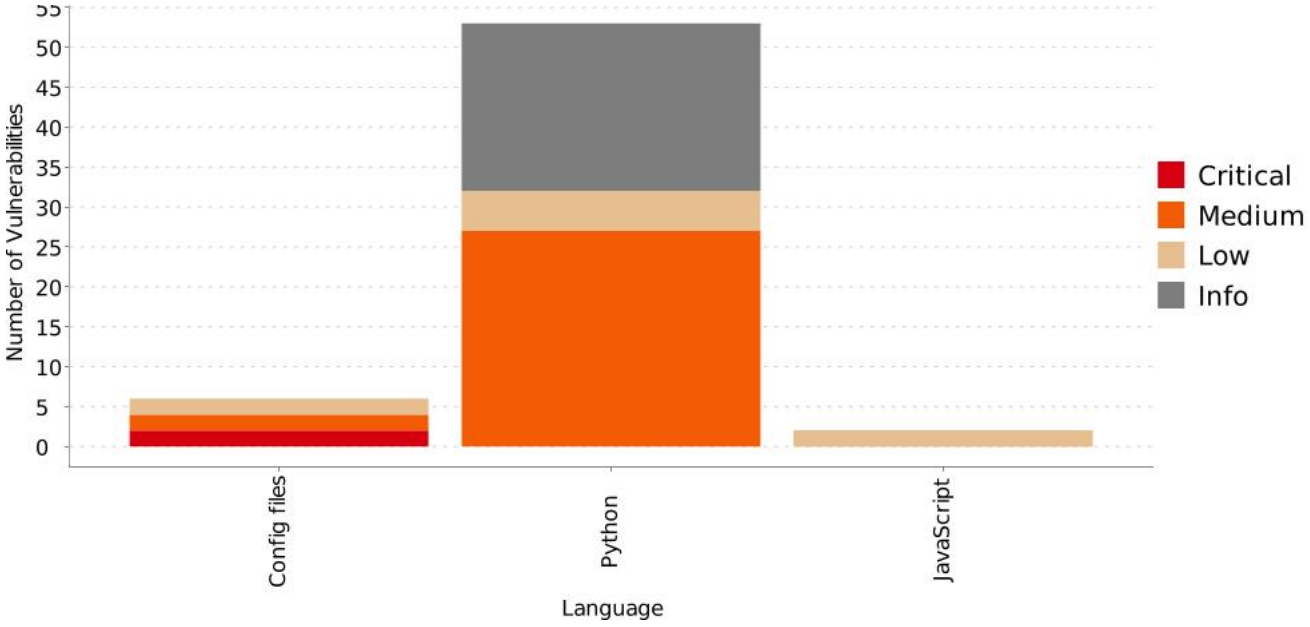
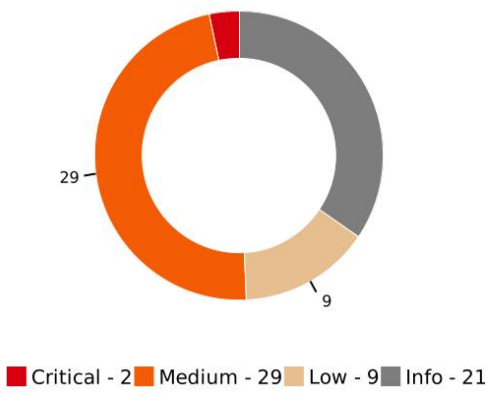


Diagram of identified vulnerabilities



Vulnerable Dependencies

Vulnerability List

Vulnerabilities are displayed accordingly to export settings: **31 selected**

Actual: **31 of 61**

Critical vulnerabilities		**2**
Hardcoded encryption key in config	Config files	2
bankparserui/.env.example:8		Not processed
bankparserui/.env.example:14		Not processed
Medium-level vulnerabilities		**29**
Hardcoded encryption key in config	Config files	1
bankparserui/.env.example:17		Not processed
Persistent authentication	Config files	1
bankparserui/src/ui/upload_app.py:44		Not processed
DOS attack via regular expressions possible	Python	1
bankparserui/src/halyk_ind/transactions.py:80		Not processed
Resource injection	Python	26
bankparserui/src/alatau_city_bank/batch_parse.py:100		Not processed
bankparserui/src/alatau_city_bank/batch_parse.py:140		Not processed
bankparserui/src/api/app.py:1579		Not processed
bankparserui/src/api/storage.py:136		Not processed
bankparserui/src/api/storage.py:152		Not processed
bankparserui/src/api/storage.py:162		Not processed

* Rejected vulnerabilities are not taken into account

Medium-level vulnerabilities

Resource injection	Python
bankparserui/src/api/storage.py:174	Not processed
bankparserui/src/bcc/batch_parse.py:84	Not processed
bankparserui/src/bcc/batch_parse.py:179	Not processed
bankparserui/src/bcc/footer.py:189	Not processed
bankparserui/src/bcc/parser.py:40	Not processed
bankparserui/src/eurasian_bank/batch_parse.py:79	Not processed
bankparserui/src/eurasian_bank/batch_parse.py:148	Not processed
bankparserui/src/forte_bank/batch_parse.py:115	Not processed
bankparserui/src/forte_bank/batch_parse.py:192	Not processed
bankparserui/src/forte_bank/footer.py:195	Not processed
bankparserui/src/freedom_bank/batch_parse.py:109	Not processed
bankparserui/src/freedom_bank/batch_parse.py:191	Not processed
bankparserui/src/halyk_business/footer.py:143	Not processed
bankparserui/src/halyk_business/footer.py:158	Not processed
bankparserui/src/halyk_business/footer.py:204	Not processed
bankparserui/src/halyk_ind/parser.py:55	Not processed
bankparserui/src/kaspi_pay/parser.py:42	Not processed
bankparserui/src/utils/convert_pdf_json_page.py:208	Not processed
bankparserui/src/utils/convert_pdf_json_pages.py:72	Not processed
bankparserui/src/utils/path_security.py:112	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>1ac96ba5217fe15c4b76b8c5191583b6a24488581545844d2e67fa83c537f0b3</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

bankparserui/.env.example:8

Level Critical

Status Not processed

```
5 DB_PASSWORD=<DB_PASSWORD_FROM_INFRA>
6 DB_SSLMODE=<disable_or_require_or_verify-full>
7 API_BASE_URL=<API_URL_FROM_INFRA>

8 SECRET_KEY=<STRONG_SECRET_KEY>

9 ALLOWED_ORIGINS=<https://your-upload-ui-domain>
10
11 # Optional Vault bootstrap (KV v2):
```

bankparserui/.env.example:14

Level Critical

Status Not processed

```
11 # Optional Vault bootstrap (KV v2):
12 # VAULT_ENABLED=true
13 # VAULT_ADDR=https://vault.example.com

14 # VAULT_TOKEN=<vault_token>

15 # VAULT_KV_MOUNT=secret
16 # VAULT_CONFIG_PATH=bank-parser/upload-ui
```

```
17 # VAULT_CONFIG_KEY=configJson
```

bankparserui/.env.example:17

Level Medium

Status Not processed

```
14 # VAULT_TOKEN=<vault_token>
15 # VAULT_KV_MOUNT=secret
16 # VAULT_CONFIG_PATH=bank-parser/upload-ui
17 # VAULT_CONFIG_KEY=configJson
18 # VAULT_REQUIRED=true
19 # VAULT_OVERWRITE=false
20 # VAULT_SKIP_VERIFY=false
```

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

bankparserui/src/ui/upload_app.py:44

Level Medium

Status Not processed

```
41    "TAXPAYER_API_PORTAL_HOST",
42    "https://portal.kgd.gov.kz"  # Default, should be overridden via env var
43 )

44 TAXPAYER_API_PORTAL_TOKEN = os.environ.get(

45    "TAXPAYER_API_PORTAL_TOKEN",
46    ""  # SECURITY: Never hardcoded; set via env in production
47 )
```

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 aaaaaaaaaaaaaaX:

```
^(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

bankparserui/src/halyk_ind/transactions.py:80

Level Medium

Status Not processed

```
77    "account":      r"\bN\s*карточки/счета\b|\bкарточки/счета\b",
78 }
79 # ReDoS mitigated: patterns are compile-time constants; _score_header_tokens truncates input
80 RX = {k: re.compile(v, re.I) for k, v in ANCHORS.items()}
81 _MAX_REGEX_INPUT = 2000  # ReDoS mitigation: truncate before regex
82 _MAX_REGEX_INPUT_LEN = 500  # Limit input to prevent ReDoS
83
```

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

bankparserui/src/alatau_city_bank/batch_parse.py:100

Level Medium

Status Not processed

```
97
98 from src.utils.path_security import validate_path_for_write
99 validated = validate_path_for_write(out_path, json_dir)

100 with open(validated, "w", encoding="utf-8") as f:

101     json.dump(out, f, ensure_ascii=False, indent=2)
102
103 print(f"[json] Dumped PDF internals to {out_path}")
```

bankparserui/src/alatau_city_bank/batch_parse.py:140

Level Medium

Status Not processed

```
137 if not json_path.exists() or not json_path.is_file():
138     raise ValueError(f"Invalid JSON path: {json_path}")
139 validated_json = validate_path(json_path, json_dir)

140 with open(validated_json, "r", encoding="utf-8") as f:

141     pdf_json = json.load(f)
142
143 closing_date = header_df.iloc[0].get("closing_balance_date")
```

bankparserui/src/api/app.py:1579

Level Medium

Status Not processed

```
1576     resolved_path = ui_file.resolve()
1577     if resolved_path.exists() and resolved_path.is_file():
1578         validated = validate_path(resolved_path, PROJECT_ROOT)
1579         with open(validated, 'r', encoding='utf-8') as f:
1580             return Response(content=f.read(), media_type="text/html")
1581 except (OSError, ValueError):
1582     continue
```

bankparserui/src/api/storage.py:136

Level Medium

Status Not processed

```
133 from src.utils.path_security import validate_path_for_write
134 project_file = self._get_project_file(project_id)
135 validated = validate_path_for_write(project_file, self.base_dir)
136 with open(validated, 'w', encoding='utf-8') as f:
137     json.dump(project.to_dict(), f, ensure_ascii=False, indent=2)
138
139 # Create directory for project files
```

bankparserui/src/api/storage.py:152

Level Medium

Status Not processed

```
149 if not project_file.exists():
150     return None
151 validated = validate_path(project_file, self.base_dir)
152 with open(validated, 'r', encoding='utf-8') as f:
153     data = json.load(f)
154
```

```
155 return Project.from_dict(data)
```

bankparserui/src/api/storage.py:162

Level Medium

Status Not processed

```
159     from src.utils.path_security import validate_path_for_write
160     project_file = self._get_project_file(project.project_id)
161     validated = validate_path_for_write(project_file, self.base_dir)

162     with open(validated, 'w', encoding='utf-8') as f:

163         json.dump(project.to_dict(), f, ensure_ascii=False, indent=2)
164
165 def get_projects_by_iin(self, iin: str) -> List[Project]:
```

bankparserui/src/api/storage.py:174

Level Medium

Status Not processed

```
171 for project_file in self.base_dir.glob("project_*.json"):
172     try:
173         validated = validate_path(project_file, self.base_dir)

174         with open(validated, 'r', encoding='utf-8') as f:

175             data = json.load(f)
176
177         if data.get('iin') == iin:
```

bankparserui/src/bcc/batch_parse.py:84

Level Medium

Status Not processed

```
81     # we skip XRef for now – not needed for metadata validation
82
83 validated = validate_path_for_write(json_path, meta_dir)
84 with open(validated, "w", encoding="utf-8") as f:
85     json.dump(out, f, ensure_ascii=False, indent=2)
86
87 return json_path
```

bankparserui/src/bcc/batch_parse.py:179

Level Medium

Status Not processed

```
176 # Security: Validate path before opening
177 from src.utils.path_security import validate_path
178 validated_path = validate_path(meta_json_path, pdf_meta_dir)
179 with open(validated_path, "r", encoding="utf-8") as f:
180     pdf_json = json.load(f)
181
182 period_end = header_df.iloc[0].get("Период (конец)")
```

bankparserui/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:
```

bankparserui/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:
```

bankparserui/src/eurasian_bank/batch_parse.py:79

Level Medium

Status Not processed

```
76
77 from src.utils.path_security import validate_path_for_write
78 validated = validate_path_for_write(json_path, meta_dir)
79 with open(validated, "w", encoding="utf-8") as f:
80     json.dump(out, f, ensure_ascii=False, indent=2)
81
82 return json_path
```

bankparserui/src/eurasian_bank/batch_parse.py:148

Level Medium

Status Not processed

```
145 try:
146     from src.utils.path_security import validate_path
147     validated = validate_path(meta_json_path, pdf_meta_dir)
148     with open(validated, "r", encoding="utf-8") as f:
149         pdf_json = json.load(f)
150
151     period_end = header_df.iloc[0].get("period_end")
```

bankparserui/src/forte_bank/batch_parse.py:115

Level Medium

Status Not processed

```
112     )
113
114 validated = validate_path_for_write(json_path, meta_dir)
115 with open(validated, "w", encoding="utf-8") as f:
116     json.dump(out, f, ensure_ascii=False, indent=2, default=str)
117
118 return json_path
```

bankparserui/src/forte_bank/batch_parse.py:192

Level Medium

Status Not processed

```
189 try:
190     from src.utils.path_security import validate_path
191     validated = validate_path(meta_json_path, pdf_meta_dir)
192     with open(validated, "r", encoding="utf-8") as f:
193         pdf_json = json.load(f)
194
```

```
195     period_end = header_df.iloc[0].get("period_end")
```

bankparserui/src/forte_bank/footer.py:195

Level Medium

Status Not processed

```
192 from src.utils.path_security import validate_path
193 validated = validate_path(jsonl_path)
194 pages: List[Dict[str, Any]] = []
195 with open(validated, "r", encoding="utf-8") as f:
196     for line in f:
197         line = line.strip()
198         if line:
```

bankparserui/src/freedom_bank/batch_parse.py:109

Level Medium

Status Not processed

```
106
107 from src.utils.path_security import validate_path_for_write
108 validated = validate_path_for_write(json_path, meta_dir)
109 with open(validated, "w", encoding="utf-8") as f:
110     json.dump(out_clean, f, ensure_ascii=False, indent=2)
111
112 return json_path
```

bankparserui/src/freedom_bank/batch_parse.py:191

Level Medium

Status Not processed

```
188 try:
189     from src.utils.path_security import validate_path
190     validated = validate_path(meta_json_path, pdf_meta_dir)
191     with open(validated, "r", encoding="utf-8") as f:
192         pdf_json = json.load(f)
193
194     period_end = header_df.iloc[0].get("period_end")
```

bankparserui/src/halyk_business/footer.py:143

Level Medium

Status Not processed

```
140 from src.utils.path_security import validate_path
141 validated = validate_path(source_path)
142 last_obj: Dict[str, Any] = {}
143 with open(validated, "r", encoding="utf-8") as f:
144     for line in f:
145         line = line.strip()
146         if not line:
```

bankparserui/src/halyk_business/footer.py:158

Level Medium

Status Not processed

```
155
156 from src.utils.path_security import validate_path
157 validated = validate_path(source_path)
158 with open(validated, "r", encoding="utf-8") as f:
159     return f.read()
160
```

```
161
```

bankparserui/src/halyk_business/footer.py:204

Level Medium

Status Not processed

```
201 _proj = Path(__file__).resolve().parents[2]
202 out_json_safe = Path(args.out_json).resolve()
203 validated = validate_path_for_write(out_json_safe, _proj)
204 with open(validated, "w", encoding="utf-8") as f:
205     json.dump(res, f, ensure_ascii=False, indent=2)
206 print(f" JSON saved → {args.out_json}")
207
```

bankparserui/src/halyk_ind/parser.py:55

Level Medium

Status Not processed

```
52 from src.utils.path_security import validate_path
53 validated = validate_path(jsonl_path)
54 pages: List[Dict[str, Any]] = []
55 with open(validated, "r", encoding="utf-8") as f:
56     for line in f:
57         line = line.strip()
58         if not line:
```

bankparserui/src/kaspi_pay/parser.py:42

Level Medium

Status Not processed

```
39 from src.utils.path_security import validate_path
40 validated = validate_path(path)
41 pages: List[Dict[str, Any]] = []

42 with open(validated, "r", encoding="utf-8") as f:

43     for line in f:
44         line = line.strip()
45         if not line:
```

bankparserui/src/utils/convert_pdf_json_page.py:208

Level Medium

Status Not processed

```
205         out["XRef"] = xref
206
207 validated = validate_path_for_write(out_path, out_dir if not args.out else _PROJECT_ROOT)

208 with open(validated, "w", encoding="utf-8") as f:

209     json.dump(out, f, ensure_ascii=False, indent=2)
210
211 print(f"Dumped to {out_path}")
```

bankparserui/src/utils/convert_pdf_json_pages.py:72

Level Medium

Status Not processed

```
69         words_per_page.append(words)
70
71 # 2) Raw content streams (pikepdf). If pikepdf is unavailable, degrade gracefully.

72 with open(validated_out, "w", encoding="utf-8") as out:

73     if pikepdf is not None:
74         with pikepdf.open(str(pdf_path)) as pdf:
```

```
75          for i, page in enumerate(pdf.pages):
```

bankparserui/src/utils/path_security.py:112

Level Medium

Status Not processed

```
109     ValueError: If path is invalid or escapes base_dir
110 """
111 validated_path = validate_path(file_path, base_dir)
112 return open(validated_path, mode, **kwargs)
```

WAF Setup Instructions

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. bankparserui/src/alatau_city_bank/batch_parse.py:100
2. bankparserui/src/alatau_city_bank/batch_parse.py:140
3. bankparserui/src/api/app.py:1579
4. bankparserui/src/api/storage.py:136
5. bankparserui/src/api/storage.py:152
6. bankparserui/src/api/storage.py:162
7. bankparserui/src/api/storage.py:174
8. bankparserui/src/bcc/batch_parse.py:84
9. bankparserui/src/bcc/batch_parse.py:179
10. bankparserui/src/bcc/footer.py:189
11. bankparserui/src/bcc/parser.py:40
12. bankparserui/src/eurasian_bank/batch_parse.py:79
13. bankparserui/src/eurasian_bank/batch_parse.py:148
14. bankparserui/src/forte_bank/batch_parse.py:115
15. bankparserui/src/forte_bank/batch_parse.py:192
16. bankparserui/src/forte_bank/footer.py:195
17. bankparserui/src/freedom_bank/batch_parse.py:109
18. bankparserui/src/freedom_bank/batch_parse.py:191
19. bankparserui/src/halyk_business/footer.py:143
20. bankparserui/src/halyk_business/footer.py:158

21. bankparserui/src/halyk_business/footer.py:204

22. bankparserui/src/halyk_ind/parser.py:55

23. bankparserui/src/kaspi_pay/parser.py:42

24. bankparserui/src/utils/convert_pdf_json_page.py:208

25. bankparserui/src/utils/convert_pdf_json_pages.py:72

26. bankparserui/src/utils/path_security.py:112

Scan Settings

1/1 2026-02-20 07:35:57

Select files for Analysis

/*; !/*.json; !**/*.resx;

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

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