English / 日本語（要約）

eMudhra Intermediate CA Compromise

20th October 2026

> **Update, 28th October:** the developer tools Apple ships with macOS, including `git`, `curl` and `python3`, still trust the compromised certificates, and macOS 27.2 undid user-applied fixes. Read more.

The private keys of eight eMudhra intermediate CA certificates have been compromised. They chain to roots trusted by the Mozilla, Microsoft, Apple and Chrome root programs and by many downstream trust stores.

Root programs are run by operating system and browser vendors to vet certificate authorities and decide which ones their products trust. Trust stores are the lists of trusted root certificates that operating systems, browsers, language runtimes, libraries and applications use to verify certificates, and many are based on the root programs' lists.

How do we know?

To demonstrate that the keys are compromised, the original reporter Wayne signed the following message with the private key of each of the eight intermediate CAs and reported it to the Root Program Teams on the 17th of October:

> We have a problem here - Wayne

proof.zip contains the message, and for each intermediate the certificate, its public key and a signature on the message. To verify with OpenSSL:

```
openssl x509 -in EM-DV-TLS-CA-G3B.crt.pem -noout -pubkey | diff - EM-DV-TLS-CA-G
```

```
openssl dgst -sha384 -verify EM-DV-TLS-CA-G3B.pub.pem -signature EM-DV-TLS-CA-G3
```

What actions have been taken?

The root programs and many downstream trust stores have taken action to protect their users. The certificates have been blocked in Firefox and Chrome since the 17th of October, on Microsoft platforms since the 20th and on Apple products since the 22nd.

The issue was responsibly disclosed to downstream trust stores on the 19th of October. Many have already released updates or are in the process of releasing updates.

What are the consequences?

An attacker with the private key of any of the compromised intermediate CAs could impersonate any website or service.

This has already been remediated by Mozilla, Microsoft, Apple, and Chrome for their users through their tools for distributing revocation information such as OneCRL, CRLite and CRLSets.

The CA has also revoked the compromised intermediate certificates. However, many other clients do not consume revocation information and require software updates to ensure they no longer trust the compromised intermediates. These include applications and libraries that ship their own CA bundle, and apps built on Electron, such as Slack and Discord. Electron apps only pick up Electron's fix once they are updated to a version built on a patched Electron release. Updating the operating system or browser may not be enough for these clients.

Does your client still trust the compromised intermediates?

Intermediate CA	Chains to	Result
CN=EM DV TLS CA - G3B, O=eMudhra BV, C=NL	emSign Root TLS CA - G3	not yet checked
CN=EM DV TLS CA - G3C, O=eMudhra Inc, C=US	emSign Root TLS CA - G3	not yet checked

Intermediate CA	Chains to	Result
CN=EM OV TLS CA - G3B, O=eMudhra BV, C=NL	emSign Root TLS CA - G3	not yet checked
CN=EM OV TLS CA - G3C, O=eMudhra Inc, C=US	emSign Root TLS CA - G3	not yet checked
CN=EM EV TLS CA - G3B, O=eMudhra BV, C=NL	emSign Root TLS CA - G3	not yet checked
CN=EM EV TLS CA - G3C, O=eMudhra Inc, C=US	emSign Root TLS CA - G3	not yet checked
CN=EM DV TLS CA - G3B, O=eMudhra BV, C=NL	AAA Certificate Services	not yet checked
CN=EM DV TLS CA - G3C, O=eMudhra Inc, C=US	AAA Certificate Services	not yet checked
CN=EM OV TLS CA - G3B, O=eMudhra BV, C=NL	AAA Certificate Services	not yet checked
CN=EM OV TLS CA - G3C, O=eMudhra Inc, C=US	AAA Certificate Services	not yet checked
CN=EM EV TLS CA - G3B, O=eMudhra BV, C=NL	AAA Certificate Services	not yet checked
CN=EM EV TLS CA - G3C, O=eMudhra Inc, C=US	AAA Certificate Services	not yet checked
CN=EM DV TLS CA - G3B, O=eMudhra BV, C=NL (No CT)	emSign Root TLS CA - G3	not yet checked
CN=EM DV TLS CA - G3B, O=eMudhra BV, C=NL (No CT)	AAA Certificate Services	not yet checked

Run check

A ✅ only means that the connection failed. Browsers do not tell the page why, so a network problem looks the same as a rejected certificate. To double-check, open the link in that row: if it loads without a warning page, the browser is vulnerable.

If the browser is behind a proxy that intercepts TLS, the result reflects the proxy's trust store rather than the browser's. A ❌ then means the proxy still trusts the compromised intermediates, and every client behind it remains vulnerable until the proxy is updated or disabled, even if the browser itself has been patched.

On 22nd October 2026, two chains marked *No CT* were added. Their leaf certificates were never logged to Certificate Transparency (CT), so clients that enforce CT reject them even if they still trust the intermediate. A successful connection to these hosts means the client trusts the compromised intermediates *and* does not enforce CT. As described below, these are the clients that can be attacked without anyone noticing.

To check curl or other command-line clients, run:

```
curl https://dv-g3b-emudhra.we-have-a-problem.com/
curl https://dv-g3b-aaa-emudhra.we-have-a-problem.com/
```

A certificate error means the client does not trust the compromised intermediates. To check whether the client also accepts certificates that were never logged to CT, run:

```
curl https://dv-g3b-noct-emudhra.we-have-a-problem.com/
curl https://dv-g3b-aaa-noct-emudhra.we-have-a-problem.com/
```

Most browsers enforce CT, but many non-browser clients, including curl, do not which places them at greater risk.

To check a trust store offline, chains.zip contains the certificate chain served by each of the fourteen hosts above in PEM format, plus some additional test cases and instructions for verifying them with OpenSSL.

What is known about the cause?

The private keys for the intermediate certificates were incorrectly generated by the CA. This means that anyone can recover them from the certificates.

This would have been possible since the certificates were created in April 2025. There are no entries in the certificate transparency logs for these intermediates in that period, indicating that CT-enforcing clients like Firefox, Chrome, Edge and Safari could not have been attacked.

However, many clients outside of the browser ecosystem do not enforce Certificate Transparency. We do not know if attackers exploited the flawed key generation to target these clients.

Further technical information about the cause is captured in the initial incident report in Mozilla's bug tracker.

Which clients aren't yet patched?

The servers behind the checker record how many times each user-agent connects successfully, meaning it still trusts the compromised intermediates. Only these counts are recorded. See the user-agent statistics.

A JSON dump with more detail, including counts for each JA4 fingerprint, is also available.

Which intermediates were compromised?

impacted-intermediates.json lists the eight intermediates and their Sectigo cross-signs. roots.json lists the roots they chain to.

What actions can trust store operators take?

Please see this thread in Mozilla's dev-security-policy forum.

Implementors may also find chains.zip useful as it includes additional test cases to verify that any changes are effective.

What actions can users take?

Users should update their browsers and operating systems to the latest versions as soon as possible.

Who has patched?

▶ **Operating systems and distributions** **(27, 2 pending)**

- ▶ **Root store bundles and libraries** (12)
- ▶ **Language runtimes** (8, 1 pending)
- ▶ **Applications and tools** (28, 4 pending)

Add more at github.com/galadran/we-have-a-problem.com.

Are there workarounds until patches are available?

- ▶ **macOS curl, git, and Python**
- ▶ **Linux**
- ▶ **Windows**
- ▶ **Android**
- ▶ **OpenHarmony**
- ▶ **NetBSD**
- ▶ **FreeBSD**

Site published by Dennis Jackson. This site is the work of many contributors from the community. The source is on GitHub and PRs are welcome. Licensed under the Apache License, Version 2.0.