Coyote: A multi-stage banking Trojan abusing the Squirrel installer

By GReAT

Published: 2024-02-08 · Archived: 2026-04-06 03:19:51 UTC

The developers of banking Trojan malware are constantly looking for inventive ways to distribute theirs implants and infect victims. In a recent investigation, we encountered a new malware that specifically targets users of more than 60 banking institutions, mainly from Brazil. What caught our attention was the sophisticated infection chain that makes use of various advanced technologies, setting it apart from known banking Trojan infections.

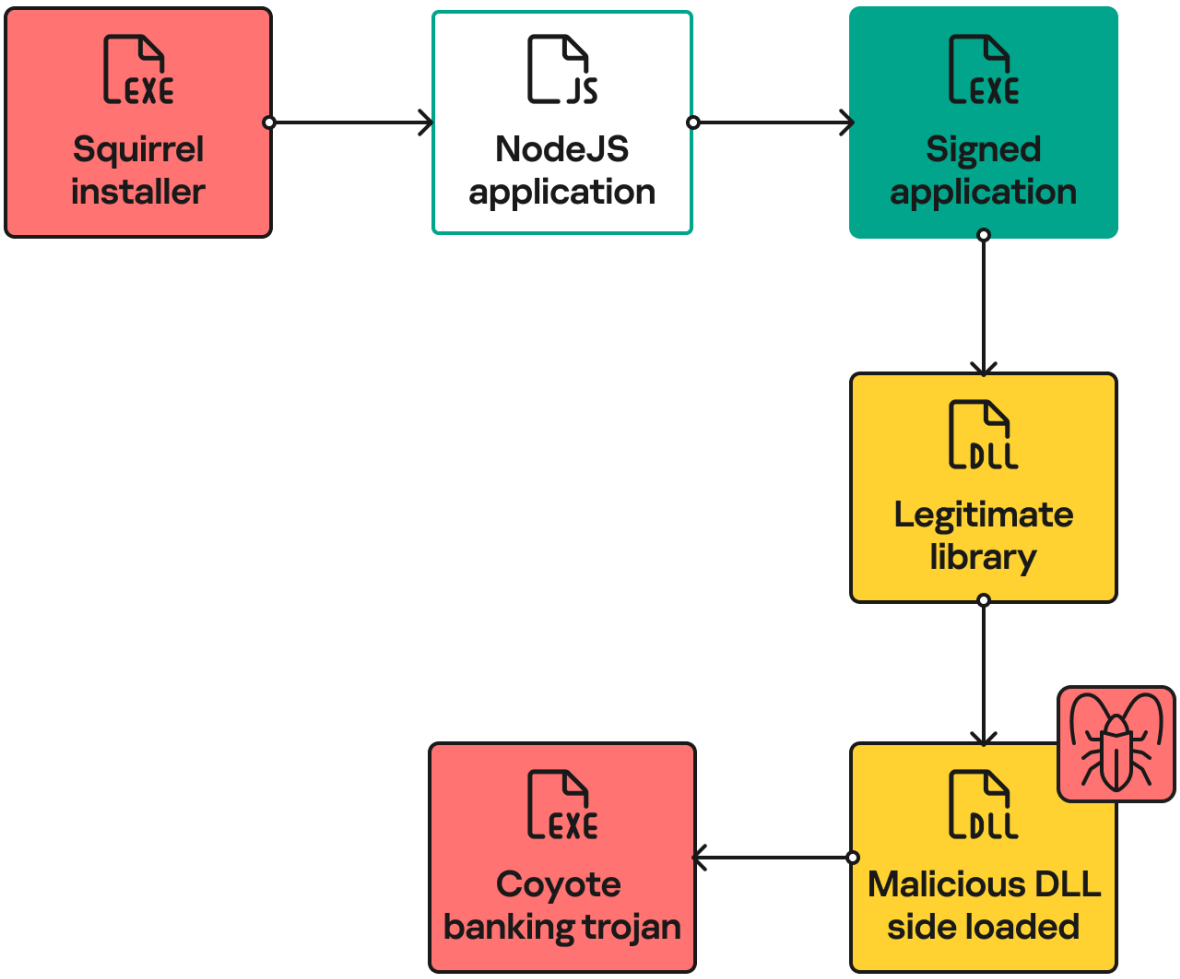
This malware utilizes the Squirrel installer for distribution, leveraging NodeJS and a relatively new multiplatform programming language called Nim as a loader to complete its infection. We have named this newly discovered Trojan "*Coyote*" due to the role of coyotes as natural predators of squirrels. The Nim language defines itself as a "statically typed compiled systems programming language that combines successful concepts from mature languages like Python, Ada and Modula". The adoption of less popular/cross-platform languages by cybercriminals is something we identified as a trend in our Crimeware and financial cyberthreats for 2024.

In this article, we will delve into the workings of the infection chain and explore the capabilities of this Trojan.

Forget old Delphi and MSI

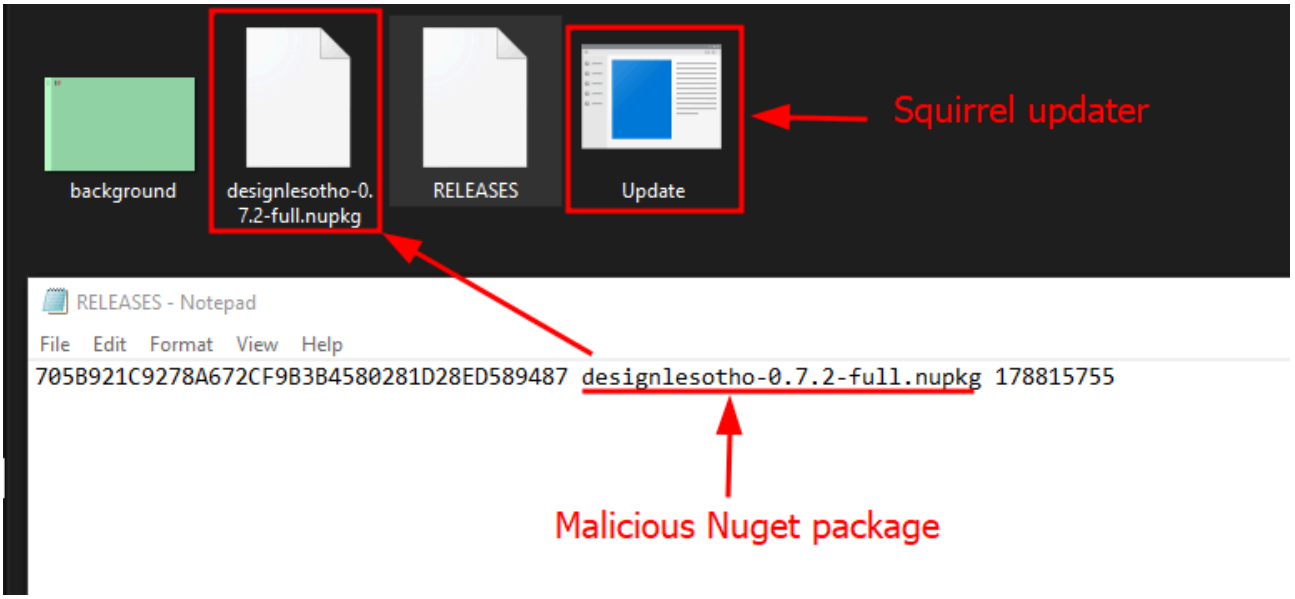
In the banking Trojan landscape, the use of the Delphi language or MSI installers is a recurring trend among malware creators. It's a well-known fact in the cybersecurity community that this method serves as a widely used initial infection vector.

Coyote does things a little differently. Instead of going down the usual route with MSI installers, it opted for a relatively new tool for installing and updating Windows desktop applications: Squirrel. As the authors explain, "*Squirrel uses NuGet packages to create installation and update packages, which means that you probably already know most of what you need to create an installer.*"



Coyote infection chain

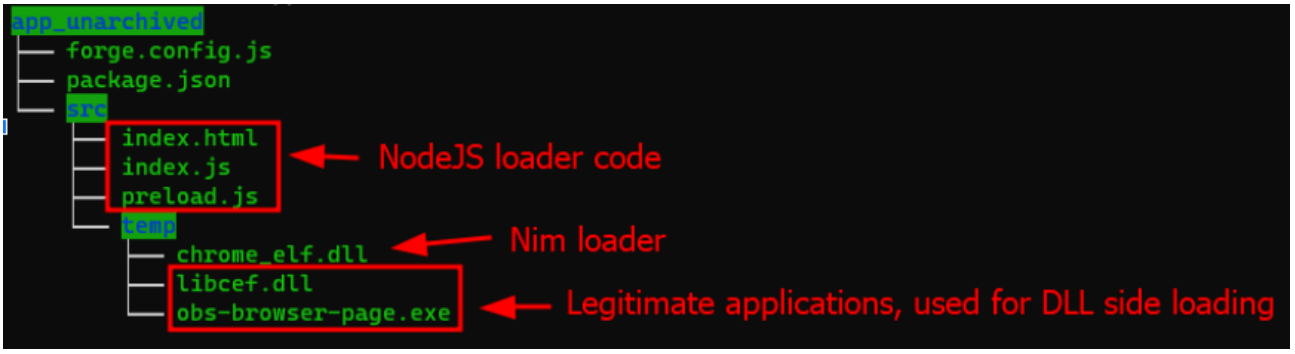
By using this tool, Coyote hides its initial stage loader by presenting it as an update packager.



Malicious Squirrel installer contents

The Node.js loader script

When Squirrel is executed, it eventually runs a NodeJS application compiled with Electron. This application executes obfuscated JavaScript code (**preload.js**), whose primary function is to copy all executables found in a local folder named ***temp*** to the user's ***captures*** folder inside the Videos folder. It then runs a signed application from that directory.

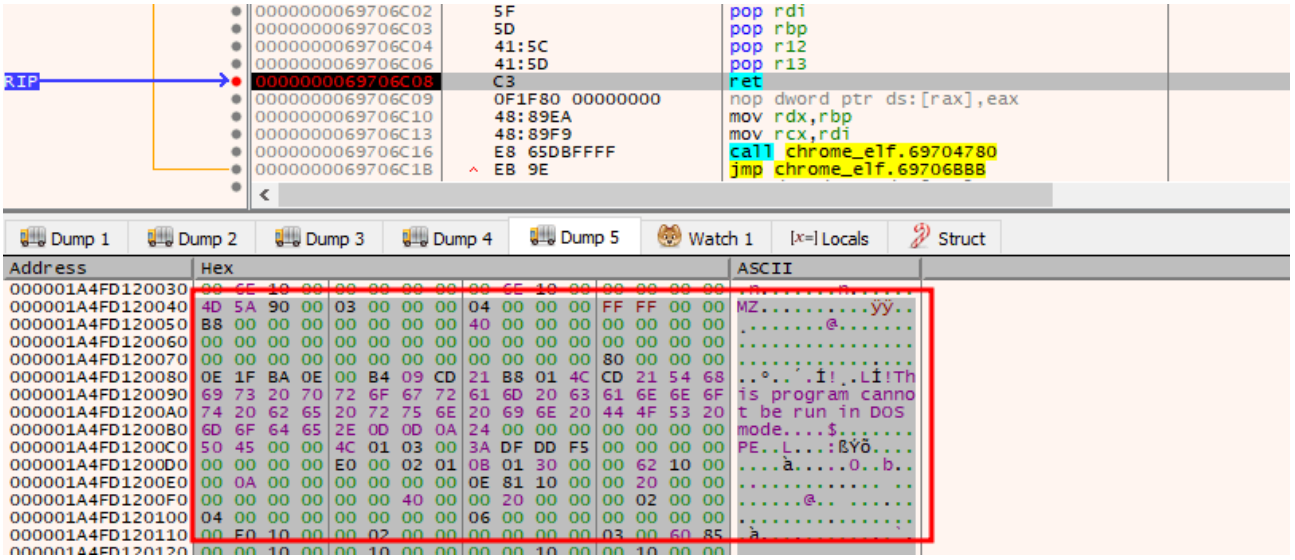


NodeJS project structure

Several executables have been identified in use, including those associated with Chrome and OBS Studio. The banker is loaded through DLL sideloading of a dependency of these executables. In all cases analyzed by our team, DLL sideloading occurs in the *libcef.dll* library.

The Nim loader

An intriguing element of the infection chain is the use of Nim, a relatively new programming language, to load the final stage. The loader's objective is to unpack a .NET executable and execute it in memory using the CLR. This implies that the loader aims to load the executable and execute it within its process, reminiscent of how Donut operates.



Unpacked .Net executable

It's worth noting that the same entry point, *obs-browser-page.exe*, is utilized for every machine reboot, serving as a means of persistence.

Last but not least, the Coyote banking Trojan

After all these steps, the Trojan is successfully executed. Coyote does not implement any code obfuscation and only uses string obfuscation with AES encryption.

```
private static string dtgmwiuxtypja(int oytvjgacr)
{
    Program.xvnxpsoety = new List<string>();
    Program.xvnxpsoety.Add("PCwSXvFXj5yA3VQXdxGOpF5dWqAzXOhKAdhzALtv2DQ=");
    Program.xvnxpsoety.Add("AJOjNowxwv7ej9KrQYYYq8+huGIDFognerEgHq3P1Kg=");
    Program.xvnxpsoety.Add("PauHwLWvmLchYGzecnFUZ4ggjxPkHvyClMEo1J3L0uY=");
    Program.xvnxpsoety.Add("pdRIAM+ub0WhDn8pUsDH61qKFJtq0ENbzIJjOxXnqVk=");
    Program.xvnxpsoety.Add("OXtnpFMvWr2UR45AwX91XDmEvdFuaOoiftHdVqdRDtq/djXlc7XXYBn3OylLGwYR");
    Program.xvnxpsoety.Add("WZmI6rev3UjJpep5sjysXXo+lfUkZpZE2SqZSwfbJos=");
    Program.xvnxpsoety.Add("ljCKvzONZZyylx8eJVHwAaQO6B/gn9rEwizuMjCAtoRnHSe7w1AjgFlVV6ijxCT9");
    Program.xvnxpsoety.Add("sKWVUD4MDX1P2ZDP5fZa3Q8gCIJL2sjSnPt7v3s6Wsw=");
    Program.xvnxpsoety.Add("fesYzSYbP6T/Z5mYisjwJYzRzJC3KmMalFG9cI2xvWA=");
    Program.xvnxpsoety.Add("f4HM0+WoaK/mQonST6Ml2LV3Y8QpbEUHfrydMk90Frk=");
    Program.xvnxpsoety.Add("wKkZIABa+Uyi/Wdl6Lucbgse6G9posdAue2mon4KJFNbqpKCbdCJzTmv7SaQrgYz");
    Program.xvnxpsoety.Add("4extOoBFkXNEZICs5QF7W909HC1Hhjsi4BvExTiQ0UdTQ6D8if/Gpn5UIBxMV8MQcPRA/
    Program.xvnxpsoety.Add("Ukf5f6M4COrxT6sELxV3y+SKgiLaW2pRliu4GIe+BEg=");
    Program.xvnxpsoety.Add("MWLrjScz3dlWciewjH2HpFmRuf5OPBf3PvLLPKfOyFk=");
    Program.xvnxpsoety.Add("bUIIDGMsk9Li1ZUhQiVwI+BuCCV2AO8nfVFqeumVtwE=");
```

Encrypted string table building

To retrieve a specific string, it calls a decryption method with the string index as a parameter. The decryption method works by creating a table of base64-encoded data. The first 16 bytes of each decoded data item serve as the *IV* (Initial Vector), while the rest is the encrypted data later used in the AES decryption routine.

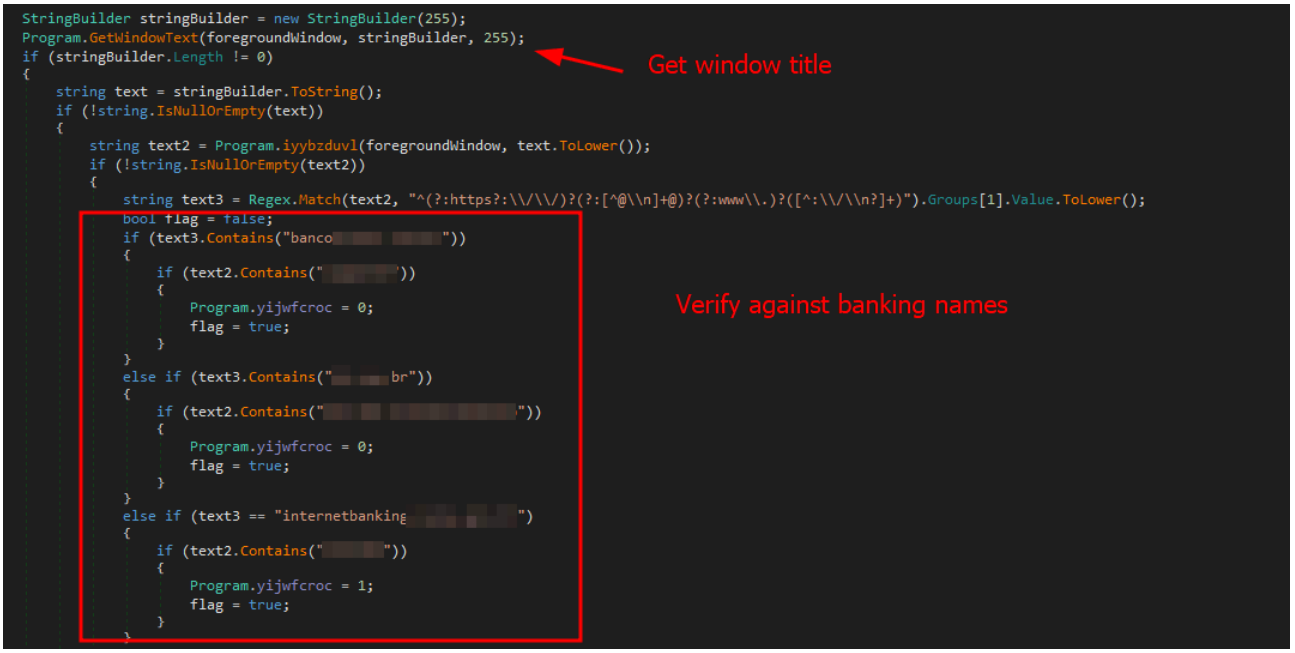
Encrypted data structure

The key is randomly generated by each executable, and the AES decryption algorithm uses the official .Net encryption interfaces. With this approach, for each string access that Coyote needs, it searches inside the table and decrypts each string with a custom *IV*.

Persistence and goals

Coyote achieves persistence by abusing Windows logon scripts; it first checks if **HKCU\Environment\UserInitMprLogonScript** exists, and if so, it inserts the registry value as the full path to the signed application, in this case, *obs-browser-page.exe*.

The Coyote Trojan's objective is consistent with typical banking Trojan behavior. It monitors all open applications on the victim's system and waits for the specific banking application or website to be accessed.



Application monitoring routine

In our analysis we identified at least 61 related applications, all originating from Brazil. This strongly suggests that Coyote is indeed a Brazilian banking Trojan, exhibiting behavior similar to that previously reported in our Tetrade blog post.

C2 communication and control

When any banking-related application is executed and utilized, the Coyote banker contacts the C2 with this information. The C2 then responds with various actions on the machine, ranging from keylogging to taking screenshots. Communication with the attacker server will be explained in the following sections.

The Trojan establishes communication with its command and control server using SSL channels with a mutual authentication scheme. This implies that the Trojan possesses a certificate from the attacker-controlled server and uses it during the connection process.

The certificate is stored as a resource in an encrypted format that is decrypted by the X509 library from .Net. Once the malware verifies that the connection is indeed with the attacker, it proceeds to send the information collected from the infected machine and banking applications to the server. The information transmitted includes:

- Machine name
- Randomly generated GUID

- Banking application being used

With this information, the attacker sends a response packet that contains specific actions. To process these actions, the attacker transmits a string with a random delimiter. Each position of the string is then converted to a list, with the first entry representing the command type.

To determine the desired command, it checks the length of the string in the first parameter, which is a random string. In other words, the only difference between commands is the size of the string.

The most important available commands are:

Length	Description
12	Take a screenshot
14	Show an overlay window of a fake banking app
15	Show a Window that is in the foreground
17	Kill a process
18	Show a full-screen overlay
21	Shut down the machine
27	Block machine with a fake banking image displaying: “Working on updates…”
31	Enable a keylogger
32	Move mouse cursor to specific X, Y position

The Trojan can also request specific bank card passwords and create a phishing overlay to capture user credentials.

Conclusion

Coyote marks a notable change in Brazilian banking Trojans. Unlike its counterparts, which often use older languages like Delphi, the developers behind Coyote are skilled in modern technologies such as Node.js, .NET, and advanced packaging techniques.

The addition of Nim as a loader adds complexity to the Trojan’s design. This evolution highlights the increasing sophistication within the threat landscape and shows how threat actors are adapting and using the latest languages and tools in their malicious campaigns.

Our telemetry data reveals that up to 90% of infections originated from Brazil. All Kaspersky products detect the threat as **HEUR:Trojan-Banker.MSIL.Coyote.gen**.

A more detailed analysis of the latest Coyote versions is available to customers of our private Threat Intelligence Reports. For more information, please contact crimewareintel@kaspersky.com.

Reference IoCs (indicators of compromise)

Host-based (MD5 hash)

03 eacccb664d517772a33255dff96020
071b6efd6d3ace1ad23ee0d6d3eead76
276f14d432601003b6bf0caa8cd82fec
5134e6925ff1397fdda0f3b48afec87b
bf9c9cc94056bcdae6e579e724e8dbbd

C2 domain list

atendesolucao[.]com
servicoasso[.]com
dowfinanceiro[.]com
centralsolucao[.]com
traktinves[.]com
diadaacaodegraca[.]com
segurancasys[.]com
