Dan Shallom Follow
May 15, 2020 · 5 min read · Listen

DLL Injection Attack in Kerberos NPM package

TL;DR

1. There is a need for awareness of the potential risks of using open source code
2. Introducing the *DLL preloading vulnerability* we discovered on Kerberos.
3. Mitigation & helpful tools and utilities.
4. https://www.npmjs.com/advisories/1514

Now it's official
CVE-2020–13110 ! Search for it !

Here we start
For those who are not familiar with NPM (Node Package Manager), it is a gigantic software registry that contains hundreds of thousands of open source Node.js projects in the form of packages. As a matter of fact, if a developer wanted to share their code with the world, NPM would be a good way to do it.

Open Source development brings much to the world by sharing ideas, perspectives and yes, eventually, helpful code. Unfortunately this code, in many cases, is not implemented with security in mind. Think of it like this: a machine is built to supply its clients with X, Y and Z which it does successfully, at least at the beginning. You see this machine was brought to market without ever undergoing a rigorous battery of testing under laboratory conditions. In time defects will begin to show and it may even become dangerous to its surroundings. So in addition to diminishing in reliability, it turns into a workplace liability, an accident just waiting to happen.

In general, it is not recommended to use open-source packages without fully reviewing and testing them to confirm they are flawless and well secured.

Figure 1- a concentrated gorilla

Let's dig in!

So let's get our hands dirty with some hot-off-the-press findings. While conducting research for one of our clients we came across a NPM Kerberos package. In practice, Kerberos is a ticket-based authentication protocol that offers a stronger encryption capability than its predecessor, NTLM.

This NPM package is responsible for supporting Kerberos authentication using the Generic Security Services Application Program Interface, on cross-platform services. While reviewing the package files, we found something of particular interest: "kerberos_sspi". SSPI (Security Support Provider Interface) is a Win32 API component that is essential for performing security operations such as the authentication itself.

Before we dive deeper into the technical details, consider this. So far there have been 28 versions of this package. 23 of them are vulnerable. Furthermore, the package has 3 dependencies. In order to understand the rate of spread (refreshingly, we're not talking about Covid-19), here's a graph that represents the number of downloads each week during the last year:

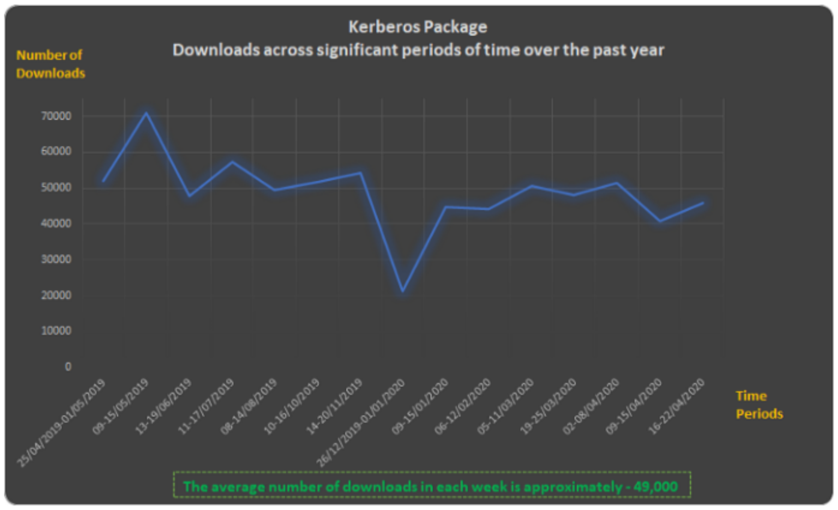

Figure 2 — Number of downloads across periods of times

Our research focused on one component used inside the package file that poses a great risk to the overall security of the user: the dynamic loading of DLLs.

The method responsible for this is called "LoadLibrary()" and it has two possible implementations: the first is supplying a fully qualified path. The second is searching for a named DLL by running through a list of directories.

```
_sspi_security_dll = LoadLibrary("security.dll");

if(_sspi_security_dll == NULL) {
   err = GetLastError();
   return err;
 }

 _sspi_secur32_dll = LoadLibrary("secur32.dll");

if(_sspi_secur32_dll == NULL) {
   err = GetLastError();
   return err;
 }
```

The LoadLibrary() searches for the named DLL in these directories in a set order and stops when it finds the DLL it is looking for. An attacker could therefore inject a legitimately named but malicious DLL into the first directory that is searched. The "LoadLibrary()" would find this DLL and stop its search. This malicious DLL would then be loaded into memory bringing with it potential for ACE (arbitrary code execution) and privilege escalation.

POC — Arbitrary code execution (ACE) and privilege escalation (PE)

Injecting a trojan DLL into an endpoint's memory can lead to the execution of malicious code with unlimited possibilities on the endpoint, but arguably the most interesting is privilege escalation. Using highest privileged access, an attacker gains full control over the system and can view, edit and modify system files.

The reason ACE and PE attacks are mentioned in one breath is that when they are chained together, their effect causes a severe impact on the system. When combined, the attacker can execute code **remotely.** Thus ACE evolves into RCE (remote code execution).

Here's how an attacker can exploit this attack:

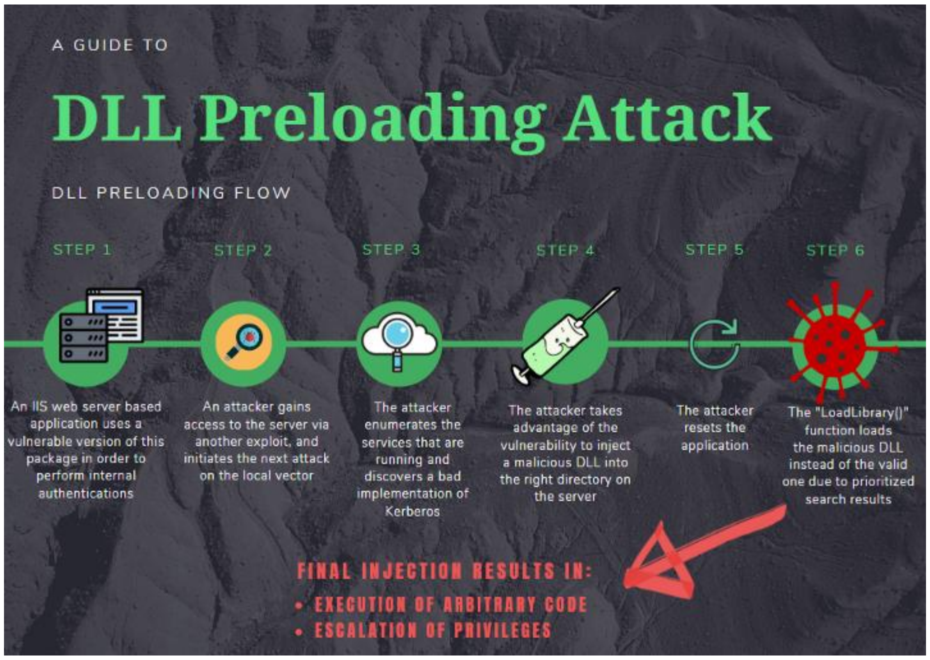
A GUIDE TO
DLL Preloading Attack
DLL PRELOADING FLOW
STEP 1
An IIS web server based application uses a vulnerable version of this package in order to perform internal authentications
STEP 2
An attacker gains access to the server via another exploit, and initiates the next attack on the local vector
STEP 3
The attacker enumerates the services that are running and discovers a bad implementation of Kerberos
STEP 4
The attacker takes advantage of the vulnerability to inject a malicious DLL into the right directory on the server
STEP 5
The attacker resets the application
STEP 6
The "LoadLibrary()" function loads the malicious DLL instead of the valid one due to prioritized search results
FINAL INJECTION RESULTS IN:
• EXECUTION OF ARBITRARY CODE
• ESCALATION OF PRIVILEGES

Figure 3- Attack's flow

Mitigation

When using functions such as "LoadLibrary()" it is essential to provide a full path for the assembly file in order to avoid preloading. With that being said, it is essential to update the Kerberos package to the latest version (that's true of all packages, of course) before deploying this software in your product.

Vulnerable versions: 0.0.1–0.0.24

Secured versions: 1.0.0–1.1.3

Figure 4- Drake's recommendation

Take away

So, now that we're done mitigating the vulnerability described above, there are some great players out there that have proved themselves in this kind of scenario:

- **Whitesource** brings open source security and a license compliance management platform to the table that helps detect vulnerable open source libraries and offers some other great utilities. Also, from my personal experience, they are very reliable when it comes to their research results ;) .
- **Veracode** offers a great "Software Composition Analysis" solution for discovering new open source vulnerabilities along with establishing an open source catalog.
- **NPM CLI**- The command line utility of NPM offers a bunch of useful commands, one of which, "npm-audit", is pretty relevant here. This command can scan a project, find vulnerabilities and automatically install updates to vulnerable dependencies. If that's not enough, it can also generate a pretty good looking report!
- **SonarQube**- A great static analyzer for inspecting code quality and identifying security vulnerabilities, along with bugs and code smells, on up to 20 programming languages.

About us

OP Innovate was established in 2014 to defend global enterprises from the increasing challenges of organizational cybersecurity. Our team has unmatched expertise in cyber research, penetration testing, incident response, training and forensics. Our team members are exposed to cutting-edge responses to today's most critical cybersecurity concerns allowing us and our partners to remain ahead of the bad guys.

Check out our blog and LinkedIn page!

Written by: Dan Shallom, Cybersecurity expert at OP Innovate.

Cybersecurity NPM Vulnerability Code Review Open Source Software