ActiveState

PERL NOVEMBER 2023 CVES

On November 25, 2023 the Perl language developers released information about two vulnerabilities in recent language releases. Patched versions are available which address these vulnerabilities.

To protect your systems, it's safest to upgrade to a patched release. You will lose no features from these patches.

If you cannot upgrade immediately, you should ensure that you are not exposed to these vulnerabilities by auditing your code and configuration as well as securing your environment to reduce the possibility of exploitation. Please note that auditing all of your systems can be difficult and time-consuming, so we strongly recommend upgrading as soon as possible.

These vulnerabilities relate to two features:

- **CVE-2023-47038:** using regular expressions which refer to specific Unicode properties
- **CVE-2023-47039:** launching external processes from Perl on Windows systems

CVE-2023-47038

Write past buffer end via illegal user-defined Unicode property

What can go wrong?

This attack can corrupt or crash your program. Combined with another unknown or potential vulnerability in Perl or a library running with Perl where users can inject data into your running process, it may be possible for a dedicated attacker to execute malicious code remotely. If your program is running with elevated privileges, the damage may be greater.

How to mitigate, if you can't upgrade

To mitigate this vulnerability, ensure that you are not using custom Unicode character properties in regular expressions in your application or any Perl library your application uses. More importantly, ensure that you do not allow unfiltered user-provided input within your application as part of regular expressions, such as to search for data or filter a data set.

What's the problem?

A well-crafted regular expression can cause the Perl interpreter to overflow a section of memory, potentially causing undefined behavior, including crashes or other exploitable behavior.

When does this apply?

Any regular expression which uses a user-defined Unicode character property in affected Perl versions is susceptible to this vulnerability.

What triggers this behavior?

To trigger this vulnerability, an attacker needs to convince your system to execute a regular expression with a custom user-defined character property embedded in it. For example, if you take untrusted input from a user or files on your system, and that input becomes part of a regular expression like:

```
if ($input =~ /\p{ExploitNamespace::IsExploit}/) { ... }
```

… then your code may be vulnerable to this problem. If you do not use this construct within your code (it's not very common) and if you do not allow users to provide unmodified input used within regular expressions in your application or within Perl libraries used by your application, then you can avoid this vulnerability.

ActiveState

CVE-2023-47039
Perl for Windows binary hijacking vulnerability

What can go wrong?

When Perl on Windows runs shell commands–or uses the shell–it executes a program called `cmd.` `cmd.exe` Perl looks for this file in the current process's system path. It also looks in the running program's current working directory.

An attacker who has permission to create a file in that directory can place a file named `cmd.exe` there. If Perl attempts to use this command to execute shell commands, Perl will execute the attacker's file.

How to mitigate, if you can't upgrade

To mitigate this vulnerability, ensure that your system's file permissions do not allow untrusted users to write files. Audit your program as well as the source code of libraries that you use to ensure you are not executing external commands or shell commands from Perl.

What's the problem?

Perl uses a program on Windows called `cmd.exe` to run external programs. When vulnerable versions of Perl search for this program, they look for that file in the program's current working directory. For example, if your program is currently running from C:\Program Files\, a vulnerable version of Perl will launch any program named `cmd.exe` in that directory even if that executable is *not* the cmd.exe you expect.

When does this apply?

Any code which launches an external program or invokes the shell from Perl may be vulnerable to this exploit, if malicious users or programs have the ability to add files to your filesystem.

What triggers this behavior?

To trigger this vulnerability, an attacker needs access to a directory that your Perl application will use. This means the attacker needs access to the machine running Perl or a network drive your Perl application will access. Your program also needs to execute Windows commands which trigger the execution of `cmd.exe` while its current working directory contains that malicious `cmd.exe.`

What can go wrong?

The malicious program can do anything your Perl program has permission to do: exfiltrate data, open a connection to a malicious external system, overwrite data, infect your network, et cetera.

www.activestate.com
Toll-free in NA: 1-866.631.4581
solutions@activestate.com