

Bypassing Linux Executable Space Protection using 20+ years old tools.

 MIT license


 4 stars  1 fork

 Star

 Notifications

 Code


 Issues

 Pull requests

 Actions

 Projects

 Security

 Insights

 main ▾

Go to file



x0reaxeax Fixed typo ...

on Feb 19  7

[View code](#)

 README.md

Executable Space Protection Bypass (CVE-2022-25265)

This POC demonstrates execution of bytes located in supposedly non-executable region of binary, therefore completely bypassing executable-space protection.

The root cause of this can be found here:

<https://github.com/torvalds/linux/blob/master/arch/x86/include/asm/elf.h#L280>

Brief

As it turns out, binary files built on either systems lacking NX or IA32 systems with NX, which do NOT contain the `PT_GNU_STACK` header will be marked with `exec-a11` . This allows for complete RWX to/from everywhere in the binary.

To achieve this, we use "historical" building tools.

In this case, gcc 3.2.2 running on x86 Slackware9 with Linux 2.4.20

We will end up with a binary file which can be executed on modern Linux systems, in this case **Linux 5.16.1**

The very same effect MIGHT be achievable with specific linker arguments/scripts, although I have NOT verified this.

The following code will copy assembled bytes of function `dummy()` to character array `harmless_str_buf` and execute the destination array as function.

[Demo with reverse shell](#)

*** DISCLAIMER ***

This demonstration serves completely for educational purposes. Under no circumstances can the author of this code be held responsible for any direct or indirect damage caused by misusing any provided code and/or information.

See [LICENSE](#) for more details

Releases

No releases published

Packages

No packages published

Languages

- C 100.0%