Vulnerability in whoopsie (previously "reporterd")

The parse_report function in whoopsie.c allows attackers to cause a denial of service (memory leak) via a crafted file. Exploitation of this issue causes excessive memory consumption which results in the Linux kernel triggering OOM killer on arbitrary process. This results in the process being terminated by the OOM killer.

Background

What is the whoopsie?

whoopsie is the "Ubuntu Error Reporting" daemon, and is installed by default in both desktop/server installations. When something crashes, whoopsie does two things: collects the crash report generated by Apport and can send them to Ubuntu/Canonical (specifically to https://daisy.ubuntu.com)

Basic operation

When a program has been crashed, Linux system tries to create a '.crash' file on '/var/crash/' directory with python script located in '/usr/share/apport/apport'. The file contains a series of system crash information including core dump, syslog, stack trace, memory map info, etc. After then, whoopsie parses key-value pairs in '.crash' file and encodes it into binary json (bson) format. Lastly, whoopsie forwards the data to a remotely connected Ubuntu error report system.

Vulnerability Description

I have found a memory leak vulnerability during the parsing the crash file, when a collision occurs on GHashTable through g_hash_table_insert(). According to [1], if the key already exists in the GHashTable, its current value is replaced with the new value. If 'key_destory_func' and 'value_destroy_func' are supplied when creating the table, the old value and the passed key are freed using that function. Unfortunately, whoopsie does not handle the old value and the passed key when collision happens. If a crash file contains same repetitive key-value pairs, it leads to memory leak as much as the amount of repetition and results in denial-of-service.

- [1] https://developer.gnome.org/glib/stable/glib-Hash-Tables.html#g-hash-table-insert

Attack

1. Generate a certain malformed crash file that contains same repetitive key-value pairs.
2. Trigger the whoopsie daemon to read the generated crash file.
3. After then, the attack results in the daemon being terminated by the OOM killer.

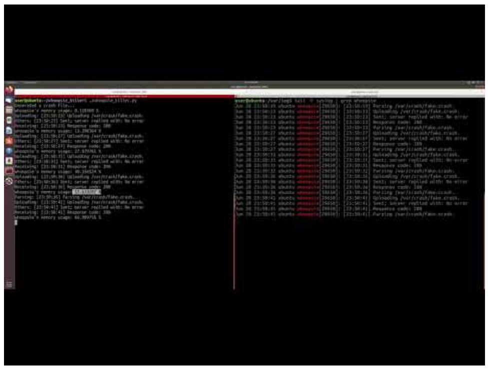
How to run

You require the following modules to run whoopsie_killer.py:

- argparse
- psutil (for retrieving information on memory utilization)

Demo video

- Let's check whoopsie_killer.poc



Releases

No releases published

Packages

No packages published

Languages

- **Python** 100.0%