Binary Code Analysis

Core dump analysis

Petr Holášek <pholasek@kerio.com>
Kerio Control team, Kerio Technologies

Core dump vs. crash dump

- core dump - dump of userland process state
- crash dump - whole system's state

Enabling the core dump

```
$ ulimit -c unlimited
$ ulimit -c
unlimited
# echo -e "*\t-\tcore\tunlimited" >> /etc/security/limits.conf
$ cat /proc/sys/kernel/core_pattern
|/usr/lib/systemd/systemd-coredump %P %u %g %s %t %c %e
```

- Warning for our VMs: The arguments in core_pattern file are different from those expected by systemd
- fix:
- ```
  # dnf update --releasever=25 systemd && reboot
  ```
  ```

Creating the core dump

hello.c

```
#include <stdio.h>

int foo(int val1, int val2, int val3, int val4)
{
    *(int *)NULL = val1;

    return val1;
}

int main(int argc, char *argv[]) {
            int x = 5;

    foo(x, 6, 7, 8);
    printf("Hello world\n");
    return 0;
}
```

- ```
  # gcc -O0 -Wall -g hello.c -o hello
  ```
- ```
  # ./hello
  ```
- What is output?

Where is the core dump?

- ```
  # coredumpctl list
  ```
- ```
  # coredumpctl info <PID>
  ```
- ```
  # coredumpctl gdb <PID>
  ```

OR

- ```
  mv /var/lib/systemd/coredump/xyz.lz4 /tmp/dump.lz4
  ```
- ```
  lz4 -d /tmp/dump.lz4
  ```
- ```
  gdb ./hello /tmp/dump
  ```

Core dump analysis using gdb

- ```
  (gdb) info threads
  ```
- ```
  (gdb) info registers
  ```
- ```
  (gdb) info bt
  ```
- ```
  (gdb) disassemble $rip
  ```
- ```
  (gdb) disassemble main
  ```
- How was variable sent to foo()?
- ```
  (gdb) print $rsp
  ```
- ```
  (gdb) print $rbp
  ```
- ```
  (gdb) x/10xg $rsp
  ```
- What addresses/values you can see on the stack?

- How to get argv[0] content (program name)? Verify then with bt command.

Explore stack on your own

- Write small C program reading the line from file given as cmd argument
- Crash it after reading line
- Try to obtain line of file from coredump with gdb
- Hint: reading subfunction, local char* buffer, fgets()