Kernel crash dump features

Konrad Witaszczyk
def@FreeBSD.org

BSD-PL

February 28, 2019

Types of kernel crash dumps

Formats

- Minidump;
  Includes only pages in use by the kernel.
- Full dump;
  Includes all pages from physical memory.
- Textdump.
  Includes text files with debugging information.

Features

- Encryption;
  RSA/AES-CBC.
- Compression;
  gzip and Zstandard.
- Transport.
  Local device, netdump protocol.

Utilities

Configuration

- dumpon(8);
- sysctl(8);
- ddb(4);
- ddb(8);
- textdump(4).

Processing

- savecore(8);
- gzip(1);
- zstd(1);
- decryptcore(8);
- ftp/netdumpd.

Analysis

- crashinfo(8);
- btsockstat(1);
- dconschat(8);
- ddb(8);
- dmesg(8);
- fstat(1);
- fuser(1);
- ifmcstat(8);
- iostat(8);
- ipcs(1);
- kgdb(1);
- kgmon(8);
- ktrdump(8);
- netstat(1);
- nfsstat(1);
- pgrep(1);
- procstat(1);
- ps(1);
- pstat(8);
- vmstat(8);
- w(1).

Configuration

Kernel (NOTES)

- options KDB
- options KDB_UNATTENDED
- options KDB_TRACE
- options DDB
- options GDB
- options BREAK_TO_DEBUGGER
- options ALT_BREAK_TO_DEBUGGER
- options INVARIANTS
- options INVARIANT_SUPPORT
- options WITNESS
- options WITNESS_SKIPSPIN
- options WITNESS_KDB
- options SOCKBUF_DEBUG
- options DEBUG_VFS_LOCKS
- options DEBUG_MEMGUARD
- options DIAGNOSTIC
- options GZIO
- options ZSTDIO
- options EKCD
- options NETDUMP

Configuration

dumpon(8)

Configures kernel to specify where and how to store kernel crash dumps.

- `dumpon /dev/ada0s1b`
- `dumpon -k public.pem /dev/ada0s1b`
- `dumpon -z /dev/ada0s1b`
- `dumpon -Z /dev/ada0s1b`
- `dumpon -s 172.16.0.1 -c 172.16.0.2 vtnet0`

dumpon can be automatically executed at boot time:

- `dumpdev="/dev/ada0s1b"`
- `dumpon_flags='-k public.pem'`

Configuration

`sysctl(8)`

- `debug.minidump`
  - 0 - write full dumps;
  - 1 - write minidumps (default).

Configuration

ddb(8)

Configures kernel debugger to specify commands that should be executed when the debugger is entered, e.g. after panic.

- `ddb script kdb.enter.panic="textdump set; capture on; bt; ps; alltrace; show alllocks; textdump dump; reset"`

ddb can be automatically executed at boot time:

- `ddb_enable="YES"`
- `ddb_config="/etc/ddb.conf"`
  `script kdb.enter.panic=textdump set; capture on; bt; ps; alltrace; show alllocks; textdump dump; reset`

Configuration

ddb(4)

Interactive kernel debugger that can be used remotely with kgdb(1).

- Can be automatically entered after panic;
  `sysctl debug.debugger_on_panic=1`
- Can be manually entered by a user;
  `sysctl debug.kdb.enter=1`
  `sysctl debug.kdb.panic=1`
- Allows to define scripts for specific events.
  - kdb.enter.cam
  - kdb.enter.panic
  - kdb.enter.sysctl
  - kdb.enter.vfslock
  - kdb.enter.watchdog
  - kdb.enter.witness
  - kdb.enter.default

Configuration

`textdump(4)`

Kernel component that allows to capture human-readable information.

- A textdump is created using `ddb(4)`;
- Requires less space;
- Output can be reviewed by an administrator before being sent for analysis;
- Requires from an administrator to specify in advance what information should be collected;
- Does not include full kernel state.

Processing

savecore(8)

Reads a kernel crash dump from a dump device and stores the dump in a directory.

- `savecore /var/crash /dev/ada0s1b`

`savecore` can be automatically executed at boot time:

- `savecore_enable="YES"`
- `dumpdir="/var/crash"`

Processing (normal case)

```
 1 root@vm:~ # dumpon /dev/ada0
 2 root@vm:~ # sysctl debug.minidump=1
 3 debug.minidump: 0 -> 1
 4 root@vm:~ # sysctl debug.kdb.enter=1
 5 debug.kdb.enter: 0KDB: enter: sysctl debug.kdb.enter
 6 [ thread pid 743 tid 100089 ]
 7 Stopped at       kdb_sysctl_enter+0x98:  movq    $0,kdb_why
 8 db> call doadump
 9 Dumping 135 out of 217 MB:..12%..24%..36%..48%..59%..71%..83%..95%
10 Dump complete
11 = 0
12 db> continue
13  -> 0
14 root@vm:~ # savecore /var/crash /dev/ada0
15 savecore 744 - - reboot
16 savecore 744 - - writing core to /var/crash/vmcore.0
17 Feb 27 22:21:08 vm savecore[744]: reboot
18 root@vm:~ #
```

Encrypted kernel crash dumps

Problem

- Kernel crash dumps might include sensitive information available in memory at the time a panic occurred.
- By default they are written to unencrypted storage that can be accessed by unauthorized users.

Encrypted kernel crash dumps

Goal

- We would like to encrypt a kernel crash dump before it is written to a dump device;
- After reboot the dump should be written in an encrypted form to storage;
- An administrator should not be able to decrypt the core dump;
- A person who analyses the core dump should be able to decrypt it.

Encrypted kernel crash dumps

Design

- A security officer generates a private RSA key;
- The public RSA key is sent to an administrator;
- The administrator configures a dump device with the public RSA key;
- System generates a one-time AES key, encrypts it with the public key and passes the key in both forms to kernel;
- When kernel wants to write a crash dump, it writes the encrypted key to the dump device, encrypts and writes the crash dump to the dump device;
- When possible, the administrator reads the encrypted key, the crash dump and writes them to storage;
- The administrator sends the encrypted key and the dump for analysis.

Processing (EKCD)

`decryptcore(8)`

Decrypts a core dump using a private RSA key.

- `decryptcore -p private.pem -k key.0 -e vmcore_encrypted.0 -c vmcore.0`

Processing (EKCD)

```
 1 root@vm:~ # dumpon -k /etc/public.pem /dev/ada0
 2 root@vm:~ # sysctl debug.minidump=1
 3 debug.minidump: 0 -> 1
 4 root@vm:~ # sysctl debug.kdb.enter=1
 5 debug.kdb.enter: 0KDB: enter: sysctl debug.kdb.enter
 6 [ thread pid 801 tid 100089 ]
 7 Stopped at       kdb_sysctl_enter+0x98:  movq    $0,kdb_why
 8 db> call doadump
 9 Dumping 135 out of 217 MB:..12%..24%..36%..48%..59%..71%..83%..95%
10 Dump complete
11 = 0
12 db> continue
13  -> 0
14 root@vm:~ # savecore /var/crash /dev/ada0
15 savecore 802 - - reboot
16 Feb 27 23:02:29 vm savecore[802]: reboot
17 savecore 802 - - writing encrypted core to /var/crash/vmcore_encrypted.1
18 root@vm:~ # decryptcore -p /etc/private.pem -k /var/crash/key.1 -e
       /var/crash/vmcore_encrypted.1 -c /var/crash/vmcore.1
19 root@vm:~ #
```

Compressed kernel crash dumps

- There are two formats available: gzip and Zstandard;
- Zstandard provides better compression ratio and performance;
- Encrypted kernel crash dumps cannot be compressed;
- Textdumps cannot be compressed;
- `savecore` allows to compress kernel crash dumps when writing to storage (the `-z` flag).

Processing (gzip)

```
 1 root@vm:~ # ddb script kdb.enter.sysctl='call doadump'
 2 root@vm:~ # dumpon -z /dev/ada0
 3 root@vm:~ # sysctl debug.kdb.enter=1
 4 debug.kdb.enter: 0KDB: enter: sysctl debug.kdb.enter
 5 [ thread pid 848 tid 100089 ]
 6 Stopped at       kdb_sysctl_enter+0x98:  movq    $0,kdb_why
 7 db:0:kdb.enter.sysctl> call doadump
 8 Dumping 135 out of 217 MB:..12%..24%..36%..48%..59%..71%..83%..95%
 9 Dump complete
10 = 0
11 db> continue
12  -> 0
13 root@vm:~ # savecore /var/crash /dev/ada0
14 savecore 849 - - reboot
15 savecore 849 - - writing core to /var/crash/vmcore.3.gz
16 Feb 27 23:12:30 vm savecore[849]: reboot
17 root@vm:~ # time gzip -kd /var/crash/vmcore.3.gz
18 0.719u 0.139s 0:01.29 65.1%      31+185k 150+1087io 5pf+0w
19 root@vm:~ # ls -lh /var/crash/vmcore.3 /var/crash/vmcore.3.gz
20 -rw-------  1 root  wheel   136M Feb 27 23:12 /var/crash/vmcore.3
21 -rw-------  1 root  wheel    25M Feb 27 23:12 /var/crash/vmcore.3.gz
22 root@vm:~ #
```

Processing (Zstandard)

```
root@vm:~ # ddb script kdb.enter.sysctl='call doadump'
root@vm:~ # dumpon -Z /dev/ada0
root@vm:~ # sysctl debug.kdb.enter=1
debug.kdb.enter: 0KDB: enter: sysctl debug.kdb.enter
[ thread pid 854 tid 100089 ]
Stopped at      kdb_sysctl_enter+0x98:  movq    $0,kdb_why
db:0:kdb.enter.sysctl> call doadump
Dumping 135 out of 217 MB:..12%..24%..36%..48%..59%..71%..83%..95%
Dump complete
= 0
db> continue
 -> 0
root@vm:~ # savecore /var/crash /dev/ada0
savecore 855 - - reboot
Feb 27 23:13:12 vm savecore[855]: reboot
savecore 855 - - writing core to /var/crash/vmcore.4.zst
root@vm:~ # time zstd -d /var/crash/vmcore.4.zst
/var/crash/vmcore.4.zst: 142577664 bytes
0.243u 0.294s 0:01.27 41.7%     71+179k 25+485io 13pf+0w
root@vm:~ # ls -lh /var/crash/vmcore.4 /var/crash/vmcore.4.zst
-rw-------  1 root  wheel   136M Feb 27 23:13 /var/crash/vmcore.4
-rw-------  1 root  wheel    24M Feb 27 23:13 /var/crash/vmcore.4.zst
root@vm:~ #
```

Kernel crash dumps transmitted over network

- Kernel is a client;
- `netdumpd` from ports is a server;
- Sides communicate using netdump, a UDP-based protocol;
- Compressed and encrypted kernel crash dumps can be transmitted using netdump;
- May only be used after kernel has panicked;
- Only IPv4 is supported.

Processing (netdump)

Server:

```
1 def@tpad:~ % netdumpd -D -a 172.16.0.1 -d /usr/home/def/crash -P /usr/home/def/crash/pid
2 netdumpd: listening on IP 172.16.0.1
3 Waiting for clients.
```

Processing (netdump)

Client (panicked kernel):

```
root@vm:~ # dumpon -k /etc/public.pem -s 172.16.0.1 -c 172.16.0.2 vtnet0
root@vm:~ # sysctl debug.kdb.panic=1
debug.kdb.panic: 0panic: kdb_sysctl_panic
cpuid = 0
time = 1551311684
KDB: stack backtrace:
db_trace_self_wrapper() at db_trace_self_wrapper+0x2b/frame 0xfffffe000040f670
vpanic() at vpanic+0x1a3/frame 0xfffffe000040f6d0
panic() at panic+0x43/frame 0xfffffe000040f730
kdb_sysctl_panic() at kdb_sysctl_panic+0x61/frame 0xfffffe000040f760
sysctl_root_handler_locked() at sysctl_root_handler_locked+0x7b/frame 0↩
     xfffffe000040f7a0
sysctl_root() at sysctl_root+0x20b/frame 0xfffffe000040f820
userland_sysctl() at userland_sysctl+0x17b/frame 0xfffffe000040f8d0
sys___sysctl() at sys___sysctl+0x5f/frame 0xfffffe000040f980
amd64_syscall() at amd64_syscall+0x28c/frame 0xfffffe000040fab0
fast_syscall_common() at fast_syscall_common+0x101/frame 0xfffffe000040fab0
--- syscall (202, FreeBSD ELF64, sys___sysctl), rip = 0x8004181ea, rsp = 0↩
     x7fffffffd588, rbp = 0x7fffffffd5c0 ---
KDB: enter: panic
[ thread pid 699 tid 100091 ]
Stopped at      kdb_enter+0x3b: movq    $0,kdb_why
db> call doadump
netdump: overwriting mbuf zone pointers
netdump in progress. searching for server...
netdumping to 172.16.0.1 (02:60:ad:59:0d:00)
Dumping 131 out of 217 MB:..13%..25%..37%..49%..61%..73%..85%..98%
netdump finished.

Dump complete
= 0
db>
```

Processing (netdump)

Server:

```
1  def@tpad:~ % netdumpd -D -a 172.16.0.1 -d /usr/home/def/crash -P /usr/home/def/crash/pid
2  netdumpd: listening on IP 172.16.0.1
3  Waiting for clients.
4  Empty bounds file for client 172.16.0.2 [172.16.0.2]
5  New dump from client 172.16.0.2 [172.16.0.2] (to ./vmcore.172.16.0.2.0)
6  ........(KDH from 172.16.0.2 [172.16.0.2])
7  (EKCD key from 172.16.0.2 [172.16.0.2])
8  Completed dump from client 172.16.0.2 [172.16.0.2]
9  ^C
10 def@tpad:~ % ls -lh /usr/home/def/crash
11 total 135292
12 -rw-------  1 def  def     2B Feb 27 23:54 bounds.172.16.0.2
13 -rw-------  1 def  def   351B Feb 27 23:55 info.172.16.0.2.0
14 lrwxr-xr-x  1 def  def    17B Feb 27 23:55 info.172.16.0.2.last -> info↩
       .172.16.0.2.0
15 -r--------  1 def  def   4.0K Feb 27 23:55 key.172.16.0.2.0
16 -rw-------  1 def  def     5B Feb 27 23:53 pid
17 -rw-------  1 def  def   132M Feb 27 23:55 vmcore_encrypted.172.16.0.2.0
18 lrwxr-xr-x  1 def  def    29B Feb 27 23:55 vmcore.172.16.0.2.last -> ↩
       vmcore_encrypted.172.16.0.2.0
19 def@tpad:~ %
```

Security considerations

`dumpon`, `savecore`, `netdumpd`, `decryptcore` are using Capsicum!

However, we need to remember that:

- EKCD does not provide integrity nor authentication;
- `netdumpd` does not perform any client authentication.

We should address these problems.

Analysis

crashinfo(8)

Analyses a core dump using the following commands:
`ps -axlww`, `vmstat -s`, `vmstat -m`, `vmstat -z`, `vmstat -i`, `pstat -T`, `pstat -s`, `iostat`, `ipcs -a`, `ipcs -T`, `nfsstat`, `netstat -s`, `netstat -m`, `netstat -anA`, `netstat -aL`, `fstat`, `dmesg`, `config`.

`crashinfo` can be automatically executed at boot time:

- `crashinfo_enable="YES"`
- `crashinfo_program="/usr/sbin/crashinfo"`

We would like to allow to configure `crashinfo` and define user scripts, similarly to `ddb(8)`.

Analysis

`kgdb(1)`

Modified `gdb` that is able to read FreeBSD kernel crash dumps.
It can also use GDB remote serial protocol to debug remote kernel.

Analysis

Other utilities

There are many utilities that implement `-M core` and `-N system` flags and allow to perform operations as they were executed at a system at the time the crash dump was created.

Questions?