NAME

dumpon - specify a device for crash dumps

SYNOPSIS

dumpon [**-i** *index*] [**-r**] [**-v**] [**-C** *cipher*] [**-k** *pubkey*] [**-Z**] [**-z**] *device*
dumpon [**-i** *index*] [**-r**] [**-v**] [**-C** *cipher*] [**-k** *pubkey*] [**-Z**] [**-z**] [**-g** *gateway*] **-s** *server* **-c** *client iface*
dumpon [**-v**] **off**
dumpon [**-v**] **-l**

DESCRIPTION

The **dumpon** utility is used to configure where the kernel can save a crash dump in the case of a panic.

System administrators should typically configure **dumpon** in a persistent fashion using the rc.conf(5) variables *dumpdev* and *dumpon_flags*. For more information on this usage, see rc.conf(5).

Starting in FreeBSD 13.0, **dumpon** can configure a series of fallback dump devices. For example, an administrator may prefer netdump(4) by default, but if the netdump(4) service cannot be reached or some other failure occurs, they might choose a local disk dump as a second choice option.

General options

-i *index* Insert the specified dump configuration into the prioritized fallback dump device list at the specified index, starting at zero.

If **-i** is not specified, the configured dump device is appended to the prioritized list.

-r Remove the specified dump device configuration or configurations from the fallback dump device list rather than inserting or appending it. In contrast, "**dumpon** off" removes all configured devices. Conflicts with **-i**.

-k *pubkey* Configure encrypted kernel dumps.

A random, one-time symmetric key is automatically generated for bulk kernel dump encryption every time **dumpon** is used. The provided *pubkey* is used to encrypt a copy of the symmetric key. The encrypted dump contents consist of a standard dump header, the pubkey-encrypted symmetric key contents, and the symmetric key encrypted core dump contents.

As a result, only someone with the corresponding private key can decrypt the symmetric key. The symmetric key is necessary to decrypt the kernel core. The goal of the mechanism is to provide confidentiality.

The *pubkey* file should be a PEM-formatted RSA key of at least 2048 bits.

-C *cipher* Select the symmetric algorithm used for encrypted kernel crash dump. The default is "chacha20" but "aes256-cbc" is also available. (AES256-CBC mode does not work in conjunction with compression.)

-l List the currently configured dump device(s), or /dev/null if no devices are configured.

-v Enable verbose mode.

-Z Enable compression (Zstandard).

-z Enable compression (gzip). Only one compression method may be enabled at a time, so **-z** is incompatible with **-Z**.

Zstandard provides superior compression ratio and performance.

Netdump

dumpon may also configure the kernel to dump to a remote netdumpd(8) server. (The netdumpd(8) server is available in ports.) netdump(4) eliminates the need to reserve space for crash dumps. It is especially useful in diskless environments. When **dumpon** is used to configure netdump, the *device* (or *iface*) parameter should specify a network interface (e.g., *igb1*). The specified NIC must be up (online) to configure netdump.

netdump(4) specific options include:

-c *client* The local IP address of the netdump(4) client.

-g *gateway* The first-hop router between *client* and *server*. If the **-g** option is not specified and the system has a default route, the default router is used as the netdump(4) gateway. If the **-g** option is not specified and the system does not have a default route, *server* is assumed to be on the same link as *client*.

-s *server* The IP address of the netdumpd(8) server.

All of these options can be specified in the rc.conf(5) variable *dumpon_flags*.

Minidumps

The default type of kernel crash dump is the mini crash dump. Mini crash dumps hold only memory pages in use by the kernel. Alternatively, full memory dumps can be enabled by setting the

debug.minidump sysctl(8) variable to 0.

Full dumps

For systems using full memory dumps, the size of the specified dump device must be at least the size of physical memory. Even though an additional 64 kB header is added to the dump, the BIOS for a platform typically holds back some memory, so it is not usually necessary to size the dump device larger than the actual amount of RAM available in the machine. Also, when using full memory dumps, the **dumpon** utility will refuse to enable a dump device which is smaller than the total amount of physical memory as reported by the *hw.physmem* sysctl(8) variable.

IMPLEMENTATION NOTES

Because the file system layer is already dead by the time a crash dump is taken, it is not possible to send crash dumps directly to a file.

The loader(8) variable *dumpdev* may be used to enable early kernel core dumps for system panics which occur before userspace starts.

EXAMPLES

In order to generate an RSA private key, a user can use the genrsa(1) tool:

```
# openssl genrsa -out private.pem 4096
```

A public key can be extracted from the private key using the rsa(1) tool:

```
# openssl rsa -in private.pem -out public.pem -pubout
```

Once the RSA keys are created in a safe place, the public key may be moved to the untrusted netdump client machine. Now *public.pem* can be used by **dumpon** to configure encrypted kernel crash dumps:

```
# dumpon -k public.pem /dev/ada0s1b
```

It is recommended to test if the kernel saves encrypted crash dumps using the current configuration. The easiest way to do that is to cause a kernel panic using the ddb(4) debugger:

```
# sysctl debug.kdb.panic=1
```

In the debugger the following commands should be typed to write a core dump and reboot:

```
db> dump
db> reset
```

After reboot savecore(8) should be able to save the core dump in the "dumpdir" directory, which is */var/crash* by default:

```
# savecore /dev/ada0s1b
```

Three files should be created in the core directory: *info.#*, *key.#* and *vmcore_encrypted.#* (where "#" is the number of the last core dump saved by savecore(8)). The *vmcore_encrypted.#* can be decrypted using the decryptcore(8) utility:

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

or shorter:

```
# decryptcore -p private.pem -n #
```

The *vmcore.#* can be now examined using kgdb(1) (*ports/devel/gdb*):

```
# kgdb /boot/kernel/kernel vmcore.#
```

or shorter:

```
# kgdb -n #
```

The core was decrypted properly if kgdb(1) (*ports/devel/gdb*) does not print any errors. Note that the live kernel might be at a different path which can be examined by looking at the *kern.bootfile* sysctl(8).

The **dumpon** rc(8) script runs early during boot, typically before networking is configured. This makes it unsuitable for configuring netdump(4) when the client address is dynamic. To configure netdump(4) when dhclient(8) binds to a server, dhclient-script(8) can be used to run dumpon(8). For example, to automatically configure netdump(4) on the vtnet0 interface, add the following to */etc/dhclient-exit-hooks*.

```
case $reason in
BOUND|REBIND|REBOOT|RENEW)
        if [ "$interface" != vtnet0 ] || [ -n "$old_ip_address" -a \
            "$old_ip_address" = "$new_ip_address" ]; then
                break
        fi
        if [ -n "$new_routers" ]; then
                # Take the first router in the list.
```

```
                gateway_flag="-g ${new_routers%% *}"
        fi
        # Configure as the highest-priority dump device.
        dumpon -i 0 -c $new_ip_address -s $server $gateway_flag vtnet0
        ;;
esac
```

Be sure to fill in the server IP address and change the interface name if needed.

SEE ALSO

gzip(1), kgdb(1) (*ports/devel/gdb*), zstd(1), ddb(4), netdump(4), fstab(5), rc.conf(5), config(8), decryptcore(8), init(8), loader(8), rc(8), savecore(8), swapon(8), panic(9)

HISTORY

The **dumpon** utility appeared in FreeBSD 2.0.5.

Support for encrypted kernel core dumps and netdump was added in FreeBSD 12.0.

AUTHORS

The **dumpon** manual page was written by Mark Johnston <*markj@FreeBSD.org*>,
Conrad Meyer <*cem@FreeBSD.org*>,
Konrad Witaszczyk <*def@FreeBSD.org*>, and countless others.

CAVEATS

To configure encrypted kernel core dumps, the running kernel must have been compiled with the EKCD option.

Netdump does not automatically update the configured *gateway* if routing topology changes.

The size of a compressed dump or a minidump is not a fixed function of RAM size. Therefore, when at least one of these options is enabled, the **dumpon** utility cannot verify that the *device* has sufficient space for a dump. **dumpon** is also unable to verify that a configured netdumpd(8) server has sufficient space for a dump.

-Z requires a kernel compiled with the ZSTDIO kernel option. Similarly, **-z** requires the GZIO option.

BUGS

Netdump only supports IPv4 at this time.

SECURITY CONSIDERATIONS

The current encrypted kernel core dump scheme does not provide integrity nor authentication. That is, the recipient of an encrypted kernel core dump cannot know if they received an intact core dump, nor can they verify the provenance of the dump.

RSA keys smaller than 1024 bits are practical to factor and therefore weak. Even 1024 bit keys may not be large enough to ensure privacy for many years, so NIST recommends a minimum of 2048 bit RSA keys. As a seatbelt, **dumpon** prevents users from configuring encrypted kernel dumps with extremely weak RSA keys. If you do not care for cryptographic privacy guarantees, just use **dumpon** without specifying a **-k** *pubkey* option.

This process is sandboxed using capsicum(4).