CerbNG – New Era for FreeBSD Security

(Presentation for WIP session of BSDCon, September 2003)

Paweł Jakub Dawidek <jules@garage.freebsd.pl>

Why??

Requirements

Architecture

Capabilities

Example

Future

Availability

close

Why??

- too generous UNIX security model for privileged applications, and not only them
- avoiding complex methods to secure applications, like:
  - rewriting programs to degrade their privileges
  - creating chroot'ed or jail'ed environments
  - *OpenSSH*-like privilege separation
- securing proprietary (closed-source) applications
- most of "trendy" security solutions (like compiler encorced process stack protection) do not protect system resources and therefore are not sufficient

Why??
Requirements
Architecture
Capabilities
Example
Future
Availability
close

- lazy or ignorant developers, “audit-proof” code
- evil hackers; cruel world !!

Why??

Requirements

Architecture

Capabilities

Example

Future

Availability

close

Requirements

- need to enforce resource protection as the most complete solution
- need to stay application-independent and transparent
- need to be flexabile
- need to monitor process behavior in depth
- need to be able to modify privileges in the run-time
- need to implement selective logging

Why??
Requirements
Architecture
Capabilities
Example
Future
Availability
close

Architecture

The components:

- kernel module (the main part) – interprets and executes the rules
- userland policy parser and compiler
- many working policies
- plenty of regression tests
- detailed documentation

Why??
Require-ments
Archi-tecture
Capa-bilities
Example
Future
Avail-ability
close

Capabilities

What CerbNG can do:

- non-exec mechanism based on group membership (group name specified in sysctl) including removal of `LD_⋆` environment variables
- pathname-based non-exec mechanism
- hardlink creation limited to own files
- adding permission checks for sysctl access variables (like `kern.msgbuf`, `machdep.msgbuf`)
- restricting access to debug syscalls (ptrace(2), ktrace(2))
- extending jailed process privileges (allowing ping(8) inside jail)

Why??
Requirements
Architecture
Capabilities
Example
Future
Availability
close

- logging all execve(2) calls (or any chosen syscall), including it's arguments
- allowing unprivileged users to chroot to selected directory
- calling nearly arbitratry syscall with any arguments in the run-time
- run-time sysctl creation/deletion in `cerb.user.⋆` tree and reading/writing to any sysctl
- decrease privileges on application start and increase them on selected actions (opening ICMP socket, binding to privileged ports, etc.)
- ... and a lot more

Why??
Requirements
Architecture
Capabilities
Example
Future
Availability
close

Example

```
#define        LUSERS          CB_SYSCTL("bsdcon.gid")
#define        LUSERS_LOG      CB_SYSCTL("bsdcon.log")


ADD_SYSCALLS(SYS_open);
if (INITRUN()) {
    crsysctl("bsdcon");
    crsysctl("bsdcon.gid", GET_GID("lusers"));
    crsysctl("bsdcon.log", 1);
}
if (syscall == SYS_open && ismember(LUSERS, groups) >= 0) {
    fullpath = realpath(arg[0]);
    if (fullpath @ "/var/mail/*" && arg[1] == O_RDONLY) {
       arg[0] = "/dev/null";
       if (LUSERS_LOG) {
           log(LOG_INFO, "User %s isn’t permitted to "
           "open any mailbox!", login);
       }
       return call();
    }
}
```

Why??

Requirements

Architecture

Capabilities

Example

Future

Availability

close

Future

- porting to FreeBSD 5.x and DragonFlyBSD
- extend functionality of configuration language (`for`, `else if`, `goto` constructs)
- integrating CerbNG with MAC framework (per-process policies based on process label, etc.)
- loading rules in jails (limited, “secure” functionality subset)

Why??
Require-ments
Archi-tecture
Capa-bilities
Example
Future
Avail-ability
close

Availability

- the homepage: http://cerber.sourceforge.net
- sourceforge project page: http://www.sourceforge.net/projects/cerber/
- policies: http://cerber.sourceforge.net/policies/

Why??
Requirements
Architecture
Capabilities
Example
Future
Availability
close

The End

Why??

Requirements

Architecture

Capabilities

Example

Future

Availability

close