Monthly Research

Building Secure Linux Application With Privilege Separation

F F R I , Inc
http://www.ffri.jp

Ver 1.00.02

Background

- Privilege separation is a key technology to achieve “Principle of least privilege”

- In secure programming:
  - Privilege separated application limits an impact of a vulnerability
  - Real world application
    - tcpdump, vsftpd, OpenSSH, Google Chrome

Privilege Separation

- A design of secure application architecture
  - Dividing execution units and minimizing privilege each process
  - Attacker obtains only few privileges even if the exploit is successful

- Merit of privilege separated server application
  - Strong user isolation in multi-user service
  - Limited intruder hostile action on internet services

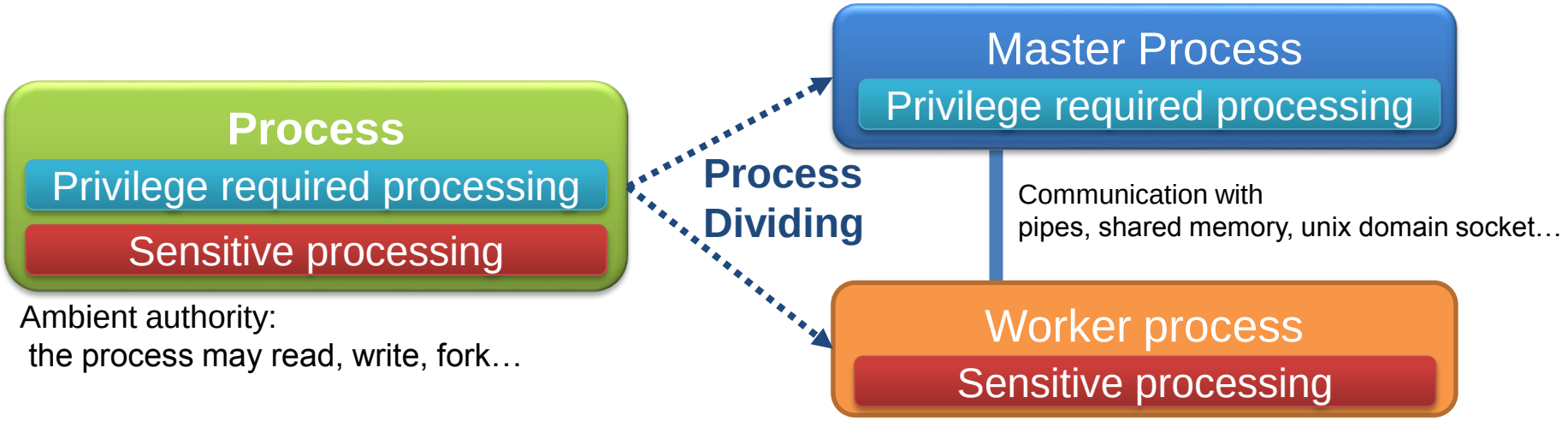
- Merit of privilege separated client application
  - Secure execution environments for untrusted remote script like javascript
    - e.g. Web browser needs a lot of privileges while running untrusted remote script

Key Technology

- Process dividing
  - Dividing a process into some processes
- Process sandboxing
  - Granting least privilege to each process
- Inter-process communication(IPC)
  - For inter-communication between divided processes
  - In Linux: Pipe, POSIX Shared memory, Unix domain socket…

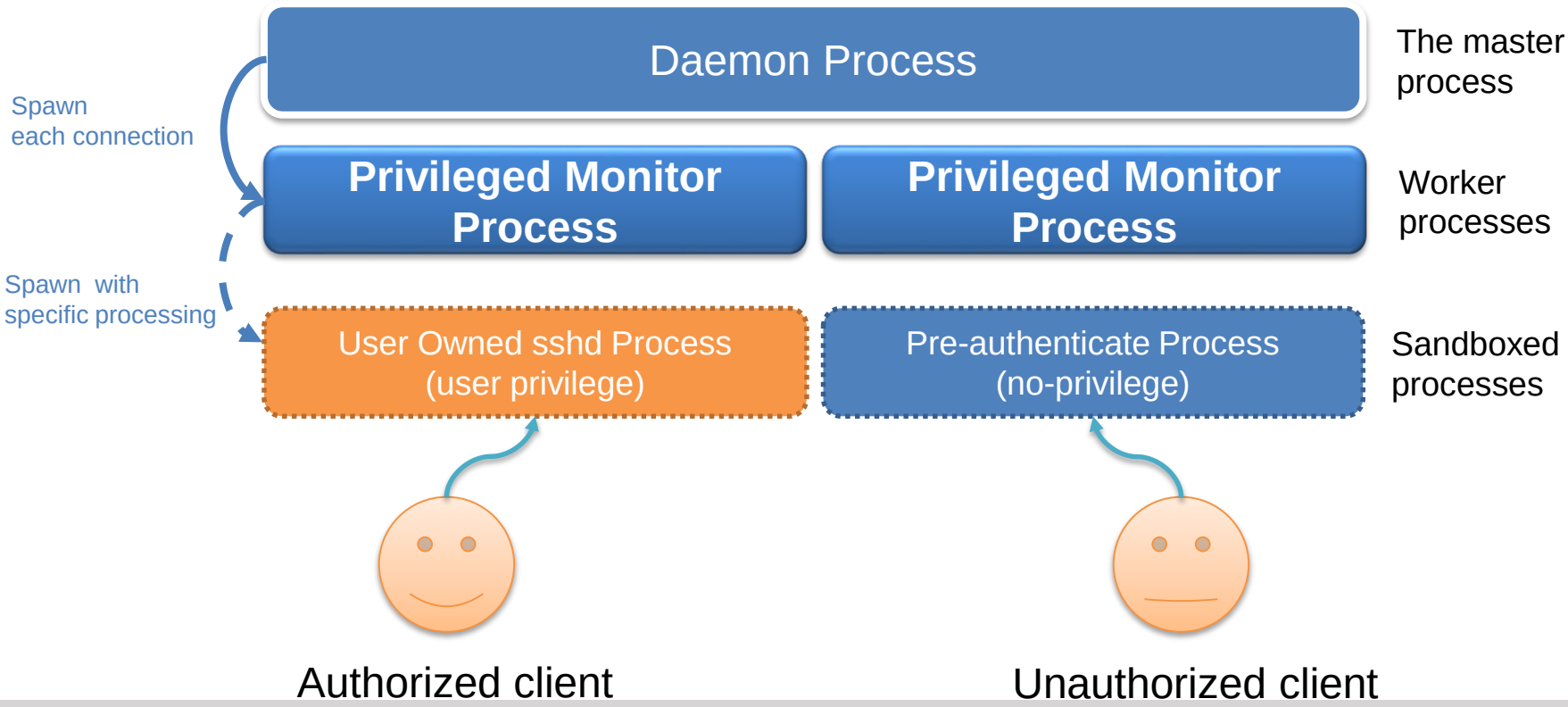
Process Dividing

- To separate between privilege required processing(like process management) and sensitive processing
  - Divided processes communicate using IPC



Example: OpenSSH

- OpenSSH daemon spawns privileged worker process per session
  - Authentication processing and authenticated user processing execute in the non-privilege process



Sandboxing on Linux

- Access Control based sandboxing
  - Using Discretionary Access Control(DAC)
    - UID, Permissions
  - Using Mandatory Accesss Control(MAC)
    - SELinux, AppArmor
  - Using Namespace
    - Chroot

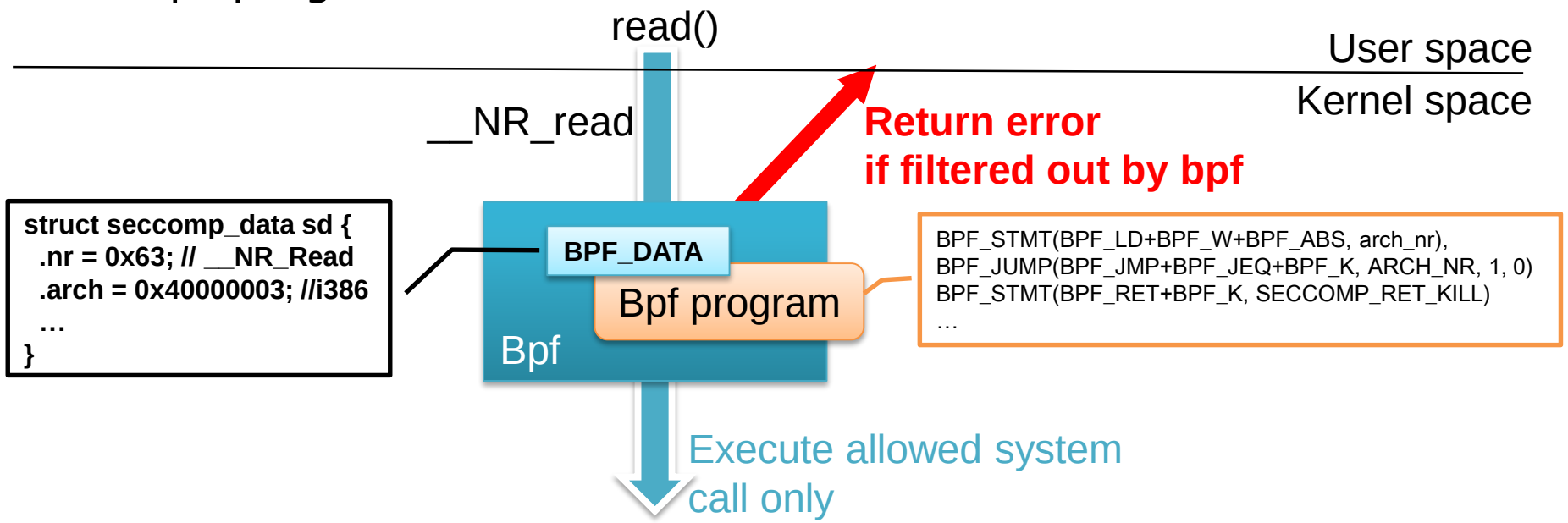
- Capability based sandboxing
  - Linux kernel capabilities (based on POSIX Capability)
  - **Linux secure computing mode**
    - State-of-the-art of sandboxing on Linux

Linux Secure Computing Mode(seccomp)

- Secure computing mode process renounces execution privileges of system calls
  - Developer has to concern themselves about “least privilege” design
- Seccomp Mode 1 (Available since Linux 2.6.12~)
  - Mode 1 permits only read(), write(), exit(), sigreturn()
- Seccomp Mode 2 (Available since Linux 3.5~)
  - Mode 2 can configures permit/denied system calls

Seccomp Mode 2(a.k.a. Seccomp-bpf)

- Seccomp Mode 2 filtered out violated system calls at system call execution
  - Kernel calls bpf(Berkeley packet filter) backend with translated bpf filter program
  - Seccomp Mode 2 configuration forces developer to describe bpf-program

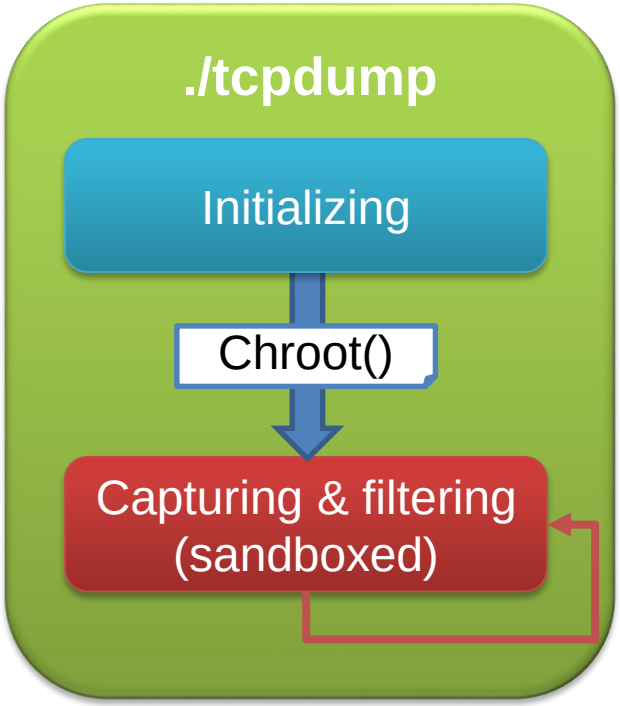


Case study

- tcpdump
  - Reducing own privilege
    - the process not divided
- vsftpd
  - Restricted accounts in multi-user services
- Google Chrome
  - Running script engine with untrusted code

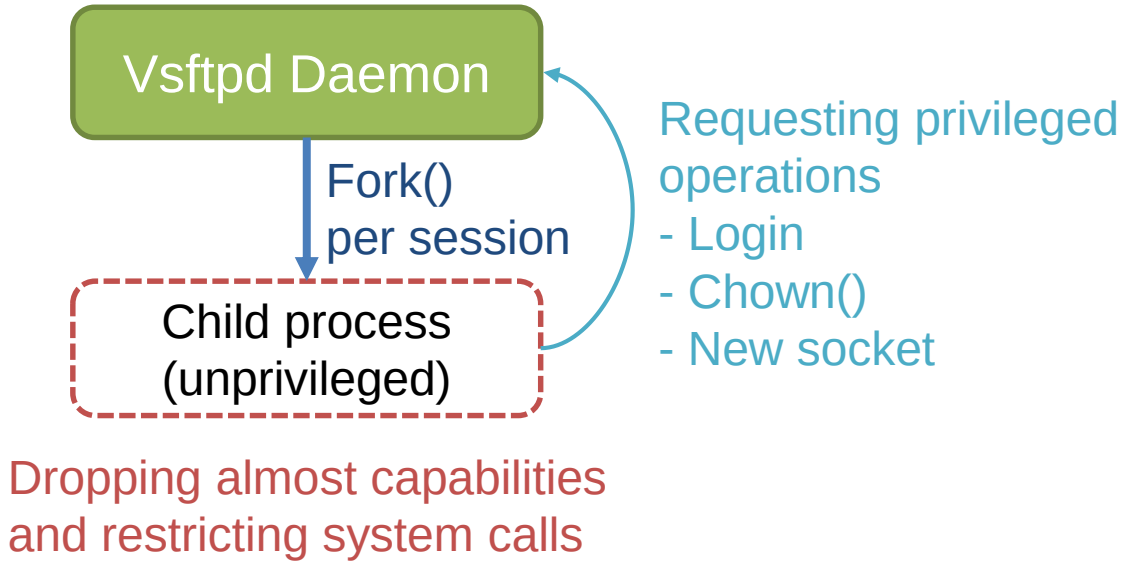
tcpdump

- tcpdump dropped own privileges before actual packet filtering
  - Sandboxing is achieved due to change own user from privileged to non-privileged user



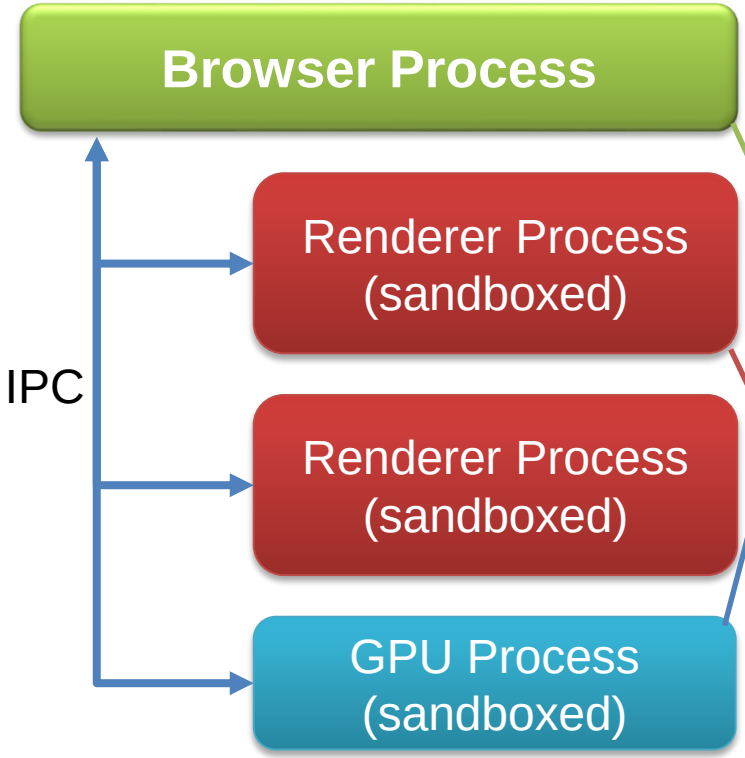
vsftpd

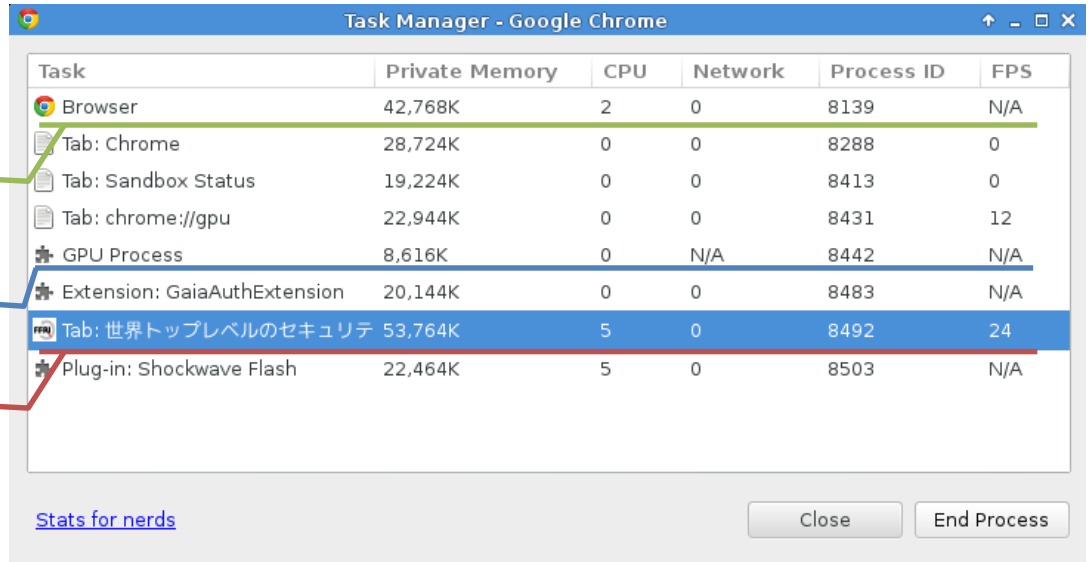
- Remote user restricted action with own privilege
  - If user needs privilege action, child process calls privileged process's function
  - Reinforcing sandbox with Seccomp Mode 2 since version 3.0.0



Google Chrome

- Renderer separates main process and its sandboxing
  - Because renderer executes untrusted remote script





Suitable a part of program for privilege separation

- Parser with untrusted data
  - e.g. Packet filtering
- Interpreter with untrusted code
  - e.g. javascript engine
- Authentication processing on multi-user service

Concerns

- Increase complexity of source code by process dividing
- Decrease portability by sandboxing
  - A number of privilege separation related component depends on OS environment
    - Process management, DAC/MAC, capabilities, IPCs..
- Deteriorate memory space effectiveness
  - Divided processes consume memory larger than a single process application

Conclusion

- Privilege separation limits incursion into your application
- Show key technology of privilege separation as follows:
  - Process dividing
  - Process sandboxing
  - Inter-process communications

- Seccomp Mode 2 is state-of-the-art of Linux sandboxing
- Some security-critical open source software has been armed process diving and sandboxing
- Privilege separation increases security, but a development cost increase again

References

- Syscall Filters
  https://fedoraproject.org/wiki/Features/Syscall_Filters
- The Chromium Projects: Design documents
  http://dev.chromium.org/developers/design-documents/
- Using simple seccomp filters
  http://outflux.net/teach-seccomp/
- Vsftpd
  https://security.appspot.com/vsftpd.html
- OpenSSH
  http://www.openssh.com/
- Preventing Privilege Escalation[Niels Provos et al, USENIX Security 2003]
  http://niels.xtdnet.nl/papers/privsep.pdf
- Capsicum[Robert R.M.W et al, USENIX Security 2010]
  http://static.usenix.org/event/sec10/tech/full_papers/Watson.pdf

Contact Information

E-Mail ： research—feedback@ffri.jp
Twitter： @FFRI_Research