Chrome/OS Security 2014

New and future hotness

Sumit Gwalani (sumit@chromium.org)
Senior Product Manager, Google

Member of Chrome Security Team

What do we do?

Everything Chrome/Chromium:

- Chrome Browser (Win/Mac/Linux/iOS/Android)
- Chrome OS (Chromebook/Chromebox/Chromebase)
- Chromecast

Thanks to everyone on the team as this deck is a small highlight of the great work by the entire team!

Core Principles

- Don’t get in the way
- Design for defense in depth (and more depth)
- Security is a team responsibility
- Speed matters
- Be transparent
- Engage the community
- Make the web safer for everyone

Chrome Browser Security

You'll know about Sandboxing, Safebrowsing, HSTS, CSP, SPDY, etc. Lets skip those!

- partitionAlloc
- Javascript Binding Integrity
- Improved Cert Pinning / SSL
- Download Safety
- ClusterFuzz
- Site Isolation

partitionAlloc

- Permanently partition allocations of certain object hierarchies into separate VM regions
- malloc() like impl→ supports different heaps at API level
- Heaps isolated from each other via guard pages
- Address space belongs to the same heap it began with
- Used in Blink to defend against use-after-free

partitionAlloc

Separate heaps for:

- DOM Nodes
- Rendering Nodes
- ArrayBuffers, strings and vector backing buffers

Note: Not a silver bullet, but some bugs are now harder to exploit

Javascript Binding Integrity

- New hardening feature added last year
- Type checking between the JS engine & C++ object layer
- How does it work?
  - C++ object creation → store unguessable representation of the V8 object type within C++ object
  - V8 wrapper objects creation → validate type representation set in C++ memory
  - Why only creation time → V8 wrapper objects contain reference pointers back to the blocks of C++ memory
- Would have stopped couple Pwn2Own/Pwnium entries

Improved Certificate Pinning/SSL

TLS1.2 in Chrome, including AES-GCM cipher

Chrome pins Google properties to Google's intermediate CA _only_, means that an attacker can compromise ANY OR EVEN ALL root CAs and Chrome <-> Google is has a layer of pinning defense

Google properties use a PFS negotiation and AES-GCM where possible (e.g. Chrome)

Google properties use 2k keys

Download Safety

- Blocking malicious downloads instead of click-through-able
- Relatively secure (well fuzzed/sandboxed) PDF reader
- Default download shelf action is to bounce PDFs into Chrome PDF viewer
- NPAPI blocked by default

ClusterFuzz

Automated crash detection, analysis and management

Fully reproducible and minimized testcases

Real-time regression and fixed testing

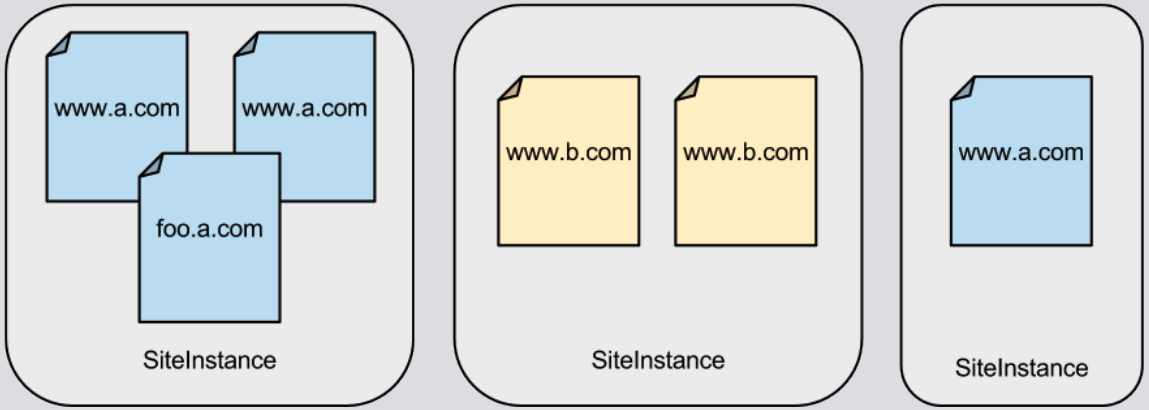
- Great to prevent security regressions as code changes on trunk
- Tons and tons of test cases processed per day
- Runs on multiple platforms (Linux, Win, Android, Mac)
- Over thousand (mostly pre-release) bugs found so far
- Coming up - code coverage, integration with clang tools, etc

Sneak Peek - Site Isolation

"Site-per-process" policy that ensures each renderer process contains page from at most one web site

Improve security for:

- Cross site iframes
- Too many renderers
- Renderer initiated navigations



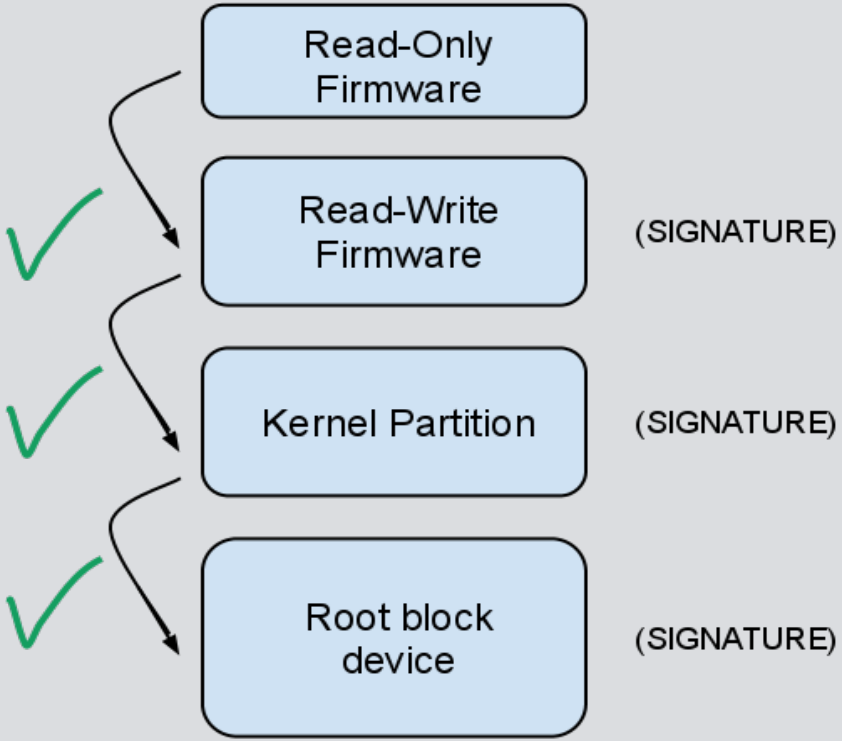
Chrome OS Security

- Verified Boot
- Encrypted storage
- Hardened OS
- seccomp-bpf
- kASLR
- Verified Access

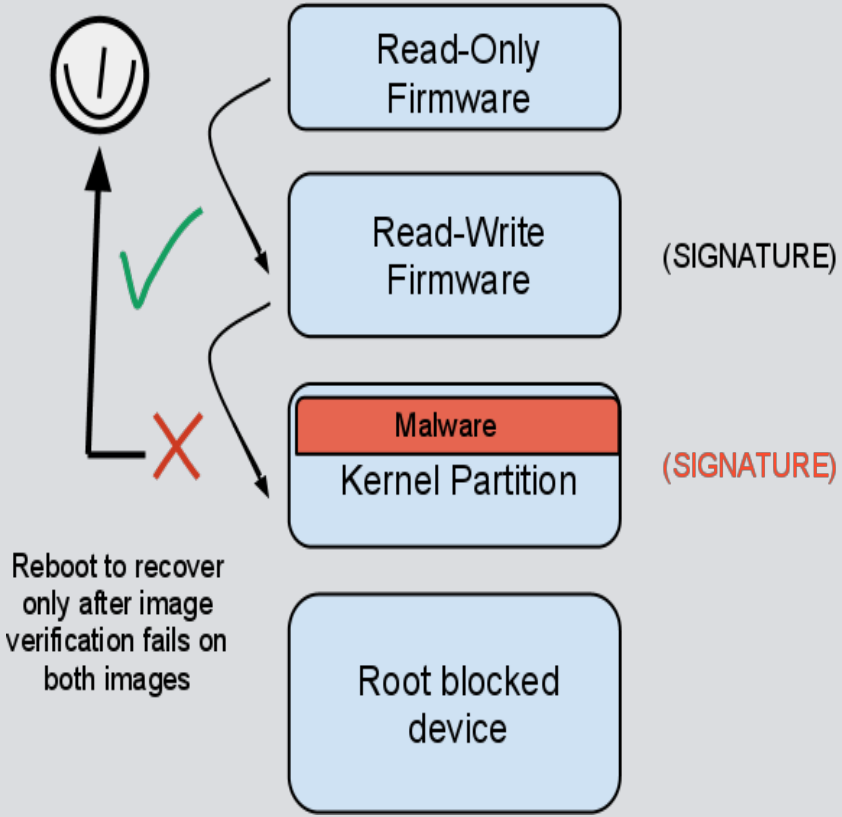
Verified Boot

- Static root of trust model for providing a verified boot flow
- Read-only firmware checks mutable firmware
  - Only the header containing the signature is parsed
- Mutable firmware checks the OS kernel and boot data
  - Partition table (GPT) is parsed
  - Kernel header containing the signature is parsed
- Kernel uses boot data to validate root filesystem

Verified boot flow

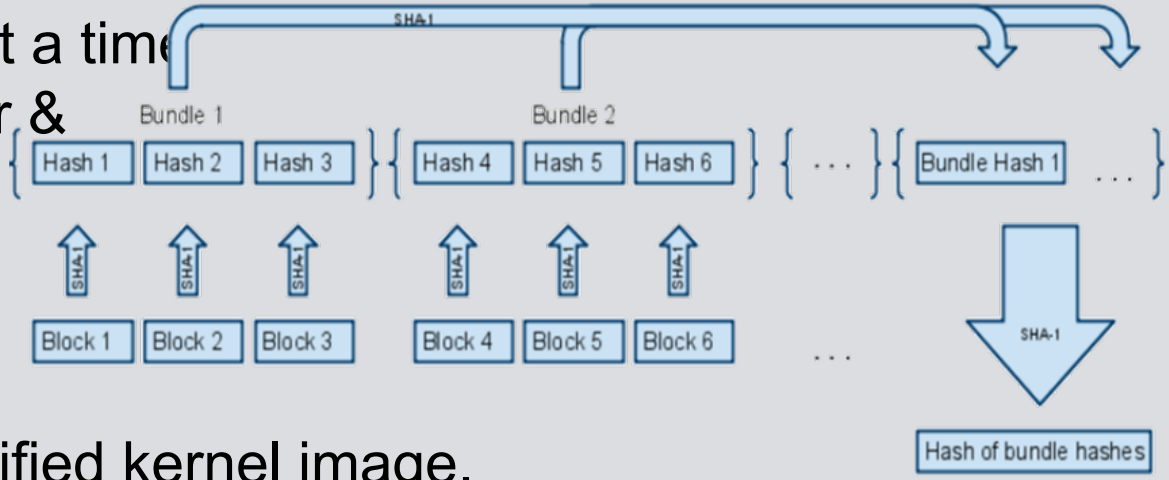


Verified boot flow



Verified Boot - Kernel

- Uses a hash tree to assert authenticity
- Verified kernel uses known good parent hash
- The second-level hashes (bundles) are verified against it
- Block-level hashes are then verified on-demand a bundle at a time
- Built on device-mapper & compatible with other trusted boot solutions
- Parent hash passed in kernel command-line, which is part of the verified kernel image.



Encrypted Storage

System storage encrypted with per-system unique key tied to TPM (uses dm-crypt)

User storage encrypted with per-user key

- Encrypted home directory per user to help protect user cached data
- Data encryption is done using eCryptfs
- TPM+password wrapped disk encryption key
- 128 bit AES symmetric encryption in block chaining mode
- Keys itself are encrypted with > 128 bit encryption

Encrypted Storage

All data at rest is encrypted, reducing risk in the event of device theft

Coming soon: evaluate dm-crypt again instead of eCryptfs

Hardened OS

- On Chrome OS, we are responsible for the entire stack
- Sandboxing and privilege-dropping helper utility: Minijail
  - User/group changes
  - Linux capabilities
  - PID/VFS namespacing
  - chroot()'ing, bind-mounting
  - `no_new_privs`
  - seccomp and seccomp-bpf filter
- Very few services running as root

Hardened OS

- Read-only root filesystem
- Module loading restrictions
  - Kernel modules limited to read-only rootfs
  - Coming soon: limit device firmware loading to read-only rootfs
- Hibernation and kexec disabled
- No Option ROMs supported
- Mount target restrictions
- Further glibc hardening
- Yama link restrictions

seccomp-bpf

Seccomp-legacy: allow read()/write()/sigreturn()/_exit() syscalls

- Provides a tight sandbox but hard to use in practice

Improving Seccomp: syscall filtering

- Attack surface reduction
- Relatively "small" change to kernel
- Leaves much of real sandbox implementation to userspace
- Past ideas: ptrace(), syscall filter bitmask, system call sets + exception signals, ftrace
- Will Drewry came-up with the idea of system call policy as a filter with Berkeley Packet Filters and using exception signals facility

seccomp-bpf summary

No loops

Used to inspect system call arguments

- That are not in user land memory & limited to arguments in registers
- Pointers cannot be followed

Decides to either:

- Kill the process (SECCOMP_RET_KILL)
- Return a specified errno (SECCOMP_RET_ERRNO)
- Trap, raising synchronous SIGSYS signal (SECCOMP_RET_TRAP)
- Notify a ptrace-er (SECCOMP_RET_TRACE)
- Allow (SECCOMP_RET_ALLOW)

Requires putting your thread in new_new_priv mode

Where is seccomp-bpf being used in Chrome OS?

- Chrome renderer (along with setuid sandbox for access control semantics)
- GPU sandboxing
- All services accessing devices (USB, etc)
- Minijail support

Coming soon: more system daemon support

Kernel ASLR (kASLR)

ASLR well established in userspace to randomize various parts of the executable (stack, mmap region, heap and program text itself)

kASLR randomizes where kernel code is at boot-time. It's a start and we can randomize address regions in future

Limitations include hibernation, entropy and info leaks

Proper care to handle info leaks:

- kptr_restrict,
- dmesg_restrict,
- log files,
- kernel objects as API handles

Kernel ASLR

Boot steps:

- Figure out lowest safe address location
- Walk E820 regions, counting kernel-sized slots
- Choose slot randomly using best available method
  - RDRAND, RDTSC, or timer IO ports
- Decompress, handle relocation, and start kernel

Coming soon: ARM support

Sneak Peek - Verified Access

Problem - High-value resources:

- Corporate network resources
- Premium A/V content

TPM protected keys validated anonymously by a service to generate per-origin certificates

The device leverages the TPM to prove that it is a legitimate Chromebook running in an approved mode (e.g. Verified Boot is enabled)

Security Rewards Program

Better protect our users and build relationship with security researchers!

Chromium VRP/Pwnium - Chromium project

- Browser
- OS
- 3rd party s/w
- Apps and Extensions by Google

Google VRP - google.com service with sensitive user-data

- *.google.com, *.youtube.com, *.blogger.com, etc

There's more - Patch Rewards!

Submit a patch for:

- Improvements to privilege separation
- Memory allocator hardening
- Cleanups of integer arithmetics
- Systematic fixes for various types of race conditions
- Elimination of error-prone design patterns or library calls.

What's covered:

- Open source: Chromium, AOSP, Blink
- security-critical components of Linux Kernel (including KVM)
- Select network services, web/mail servers, data parsers, libraries
- Toolchain security improvements for GCC, binutils, and llvm

Total Rewards

> $2 Million USD

Chromium VRP + Pwnium > $1M USD

Google VRP > $1 M USD

Pwnium 4

Chrome OS - x86 and ARM

- $110,000 USD: browser or system-level compromise in guest mode or as a logged-in user, delivered via a web page.
- $150,000 USD: compromise with device persistence: guest to guest with interim reboot, delivered via a web page.

Total pot - $2.71828 million USD
CanSecWest in Vancouver
Registrations close: March 10th, 2014

Core Principles

- Don’t get in the way
- Design for defense in depth (and more depth)
- Security is a team responsibility
- Speed matters
- Be transparent
- Engage the community
- Make the web safer for everyone

References

http://www.chromium.org/Home/chromium-security

http://www.chromium.org/chromium-os/chromiumos-design-docs

Thanks!

Questions?