



Installing or migrating to Hardened Gentoo

Tomáš Chvátal <scarabeus@gentoo.org>

2013/03/02





Who the hell is Tomáš Chvátal

- Gentoo developer since fall 2k8
- Council member since Jan 2010
- KDE team member (former lead)
- Libreoffice maintainer and also upstream developer
- Formely also working on X11, Overlays, Clustering, QA, ...





Basic informations

- Gentoo project focusing on providing latest security measures.
- Most of the features are accepted to the main project
- The reason for separate project is that security can have speed impact and make maintenance bit harder

More to be described later in the presentation.



Install from scratch

At first the plan was just to show up the instalation process but during the at home tryout i found out that it takes 5 hours from scratch.

The difference is only in one step of the normal guide

<http://www.gentoo.org/doc/en/handbook/index.xml>

You have to use stage3 at:

<http://distfiles.gentoo.org/releases/amd64/autobuilds/current-stage3/hardened/>

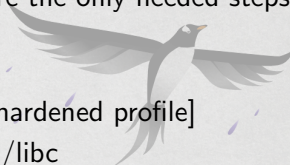
And when prompted for setting the profile by `eselect`, you just use hardened one.



Migration

I expected this to be bit complex, but the results are contrary my assesments, behold these are the only needed steps:

- eselect profile list
- eselect profile set [number of hardened profile]
- emerge -1v binutils gcc virtual/libc
- gcc-config -l
- gcc-config [the gcc from hardened profile]
- emerge -1v gcc virtual/libc
- source /etc/profile
- emerge -1evD @world





Desktop troubles

There are two issues when migrating your desktop:

- Usage of binary drivers where nvidia does not work at all and fglrx only with crazy hacks.
- Some PAX options does not allow running X at all, broken features are: CONFIG_PAX_KERNEXEC, CONFIG_GRKERNSEC_IO.
- CONFIG_GRKERNSEC_IO: ioperm and iopl calls return error when enabled, but X won't start at all.
- CONFIG_PAX_KERNEXEC: protects against code injections into kernel memory, but causes input devices like mouse to hang.





Hardened features

We can split the features into 4 independent areas:

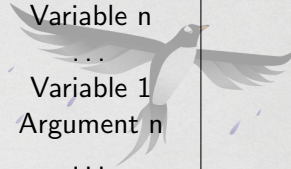
- Compilation protection
- Link time protection
- Kernel level
- GCC plugins from kernel





Stack-Smashing Protector

SSP	NORMAL GCC
Variable n	Variable n
...	...
Variable 1	Variable 1
Argument n	Argument n
...	...
Argument 1	Argument 1
Return addr	Return addr
Canary	Local variables
Local variables	—





-DFORTIFY_SOURCE=2

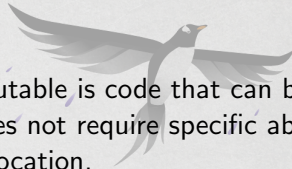
Various operations detecting software vulnerabilities in compile and run time.





PIE/PIC

Position-independent code/executable is code that can be put anywhere in the memory and does not require specific absolute address allocation.





RELRO

Linker marks some critical sections of the object like GOT and PLT tables as read only.

You can use this with linker flags `-Wl,-z,relro`

PLT means Procedure Linkage Table.

GOT stands for Global Offset Tables.



BIND_NOW

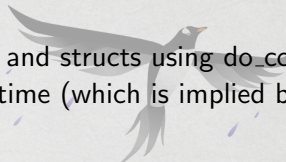
Ensures that all symbols are loaded during application start so we can have GOT and PLT tables completely read-only. Sideeffect is that apps can take more time to fire up.

You can use this with linker flags `-Wl,-z,now`



GCC plugins

- kernexec: forces function pointer calls and function returns to jump into kernel space.
- constify: makes the ops structs and structs using `do_const` parm as read only at compile time (which is implied by this for runtime too).
- stackleak: sanitizes kernel stack by providing estimate of how much deep was the kernel stack used by the syscall.
- interoverflow: detects calls which could have suffered from under/overflow by detecting these and prevents usage of incorrect sizes.





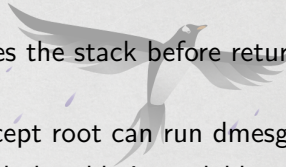
Features part 1

- ASLR, RANDMMAP: Safeguards that provide random addresses to mmap calls unless required otherwise by the call.
- RANDSTACK: Inserts random address for the stack between system calls.
- /proc only for user: Restricts the access to process informations only to owner (and group if needed).
- Brute force: If daemon aobrts or gets interesting signal it can't fork during next 30 seconds.
- UDEREF: prevents accessx to user space functions and data.
- MIN_MMAP_ADDR: forbids memory reservation on lower addresses.



Features part 2

- Memory sanitization: wipes mem pages when they are freed so nobody can read stored data.
- Kernel stack sanitization: wipes the stack before returning from syscall.
- Dmesg restriction: nobody except root can run dmesg.
- Kernel symbol hiding: the symbols table is readable only by root and `SYS_MODULE` can provide the needed info via syscalls.
- Kernel process hiding: prevents user from seeing kernel threads.





TODO

NX memory Mprotect restriction Reference counter overflow protection Free mem sanitization Kernel Stack sanitization VM86 mode restriction Disabled privileged I/O sysfs/debugfs restrictions linking restrictions fifo restrictions chroot jail restrictions enforce process limit on execs (not only on forks) dmesg restrictions TCP/UDP blackholing and LAST_ACK DoS protection Code execution on non trusted folders ptrace denial on non readable sugid binaries socket restrictions consistent multithreaded privilege enforcement active kernel exploit response kernel auditing



Links to interesting stuff about hardened

- Project page - <http://www.gentoo.org/proj/en/hardened/>
- Svens blog - <http://blog.siphos.be/category/gentoo/hardened/>





Questions

Nobody wants to ask anything. Right?





Thank you

Thank you for your attention!

