SEE TEXT ONLY

svn UPDATE

by Steven Kreuzer

Like everything else in life, security is a trade-off, and it usually comes at the expense of the end user losing functionality or having to endure a decrease in performance or throughput. What this usually means is that unless you have people and/or an organization that have decided to take security seriously, some of the first things which get turned off are the firewalls or the access control security policies. To make matters even worse, sometimes the security will cause your application to break in a nonobvious way, and taking the sledgehammer approach of turning everything off until it works again is all too common. At the other end of the spectrum, you can be left with a false sense of security and have machines that are actually vulnerable because of a very subtle typo you made in a complex security subsystem with difficult-to-understand syntax.

Security is hard, and it's even harder to get it right. One of the great things about the FreeBSD Project is that system security has always been a major focus. Developers spend a lot of time and effort to provide a secure and reliable system so that someone with little to no experience using FreeBSD can build a new machine and connect it directly to the Internet and be reasonably confident that nothing bad will happen. Over the past few years, FreeBSD has been used as a proving ground for some new and innovative concepts in computer security that are both lightweight and mostly transparent, resulting in a much better experience for both system administrators and end users alike. Since the topic of this issue is security, I wanted to use this installment of svn update to highlight some of the recent improvements designed to keep you and your data safe, be it improvements in encryption, extending FreeBSD to support hardware security features that are being baked into the CPU itself, or just the ongoing work to explicitly define what an application can and cannot do with Capsicum.

Add ipfw_pmod kernel module (https://svnweb.freebsd.org/changeset/base/316435)

This new module is designed for modification of packets from any protocol, only implementing TCP MSS modification at this time. It adds the external action handler for "tcp-setmss" action. A rule with tcp-setmss action performs an additional check for protocol and TCP flags. If SYN flag is present, it parses TCP options and modifies the MSS option if its value is greater than the configured value in the rule.

Capsicumize pom (https://svnweb.freebsd.org/changeset/base/317165)

I had a good laugh when I saw this commit because I was not aware that out-of-the-box FreeBSD provided the ability to report the phases of the moon. While this appears to be a fairly trivial application with little attack surface, it has been converted to run in a capsicum sandbox so you can rest easy at night.

Set the arm64 Execute-never bits in more places (https://svnweb.freebsd.org/changeset/base/316761)

Each memory region on arm64 can be tagged as not containing executable code. If the Execute-never, XN, bit of the descriptor is set to 1, any attempt to execute an instruction in that region results in a permission fault. Set the Execute-never bits when mapping device memory, as the hardware may perform speculative instruction fetches. Set the Privileged Execute-never bit on userspace memory to stop the kernel if it is tricked into executing it.

Enable the process state bit to disable access to userspace from the kernel on ARMv8.1 (https://svnweb.freebsd.org/changeset/base/316756)

ARMv8.1 introduced a new Privileged Access-never (PAN) state bit. This bit provides control that prevents privileged access to user data unless explicitly enabled, which provides additional security against possible software attacks.

3BSD-licensed implementation of the ChaCha20 stream cipher, intended or used by the upcoming arc4random replacement (https://svnweb.freebsd.org/changeset/base/316982)

The ChaCha20 cipher is a high-speed cipher published by Daniel J. Bernstein in 2008. It is considerably faster than AES in software-only implementations due to its use of CPU-friendly Addition-rotation-XOR operations.

Replace the RC4 algorithm for generating in-kernel secure random numbers with ChaCha20 (https://svnweb.freebsd.org/changeset/base/317015)

Writing software is hard but we are pretty fortunate that contributions being made to FreeBSD are coming from a pool of very talented people. However, these developers are human, and humans sometimes make mistakes. But we have a very active community of people who continually audit the code base looking for potential security vulnerabilities, both large and small. While not as exciting as some of the other changes I mentioned, I thought it would be interesting to take a look at a few small snippets of code that appear innocuous, but could have potentially been used as an avenue to compromise your system.

Use strlcpy to appease static checkers in dummynet (https://svnweb.freebsd.org/changeset/base/316777)

Prevent some heap overflows in restore(8) (https://svnweb.freebsd.org/changeset/base/316799)

Prevent possible sscanf() overflow in mixer(8) (https://svnweb.freebsd.org/changeset/base/317596)

Fix some trivial argv buffer overruns in ctm(1) (https://svnweb.freebsd.org/changeset/base/316795)

Avoid possible overflow via environment variable in loader(8) (https://svnweb.freebsd.org/changeset/base/316771)

Fix an out-of-bounds write when a zero-length buffer is passed (https://svnweb.freebsd.org/changeset/base/316768)

STEVEN KREUZER is a FreeBSD Developer and Unix Systems Administrator with an interest in retro-computing and air-cooled Volkswagens. He lives in Queens, New York, with his wife, two daughters, and dog.