**On the Juniper backdoor**

[](https://matthewdgreen.files.wordpress.com/2015/12/a826e-juniperberries.png)You might have heard that a few days ago, Juniper Systems [announced the discovery](http://www.wired.com/2015/12/juniper-networks-hidden-backdoors-show-the-risk-of-government-backdoors/) of “unauthorized code” in the ScreenOS software that underlies the [NetScreen](http://www.juniper.net/techpubs/en_US/release-independent/screenos/information-products/pathway-pages/netscreen-series/product/) line of devices. As a result of this discovery, the company announced a pair of separate vulnerabilities, [CVE-2015-7755 and CVE-2015-7756](https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10713&cat=SIRT_1&actp=LIST) and urged their customers to patch immediately.

The first of these CVEs (#7755) was an authentication vulnerability, caused by a malicious hardcoded password in SSH and Telnet. [Rapid7 has an excellent writeup of the issue.](https://community.rapid7.com/community/infosec/blog/2015/12/20/cve-2015-7755-juniper-screenos-authentication-backdoor) This is a pretty fantastic vulnerability, if you measure by the impact on security of NetScreen users. But on the *technological* *awesomeness* scale it rates about a two out of ten, maybe a step above ‘[hit the guy with a wrench](https://xkcd.com/538/)‘.

The *second* vulnerability is a whole different animal. The [advisory](https://kb.juniper.net/InfoCenter/index?page=content&id=JSA10713&cat=SIRT_1&actp=LIST) notes that CVE-7756 — which is independent of the first issue — “may allow a knowledgeable attacker who can monitor VPN traffic to decrypt that traffic.”This is the kind of vulnerability that makes applied cryptographers cry tears of joy. It certainly did that for me:

And while every reasonable person knows you can’t just drop “passive decryption vulnerability” and expect the world to go on with its business, this is exactly what Juniper tried to do. Since they weren’t talking about it, it fell to software experts to try to work out what was happening by looking carefully at firmware released by the company.

Now I want to be clear that *I* was not one of those software experts. IDA scares the crap out of me. But I’m fortunate to know some of the right kind of people, like [Steve Checkoway](https://www.cs.uic.edu/~s/), who I was able to get on the job, mostly by encouraging him to neglect his professional obligations. I also follow some talented folks on Twitter, like [H.D. Moore](https://twitter.com/hdmoore) and [Ralf Philipp Weinmann](https://twitter.com/esizkur?lang=en). So I was fortunate enough to watch them work, and occasionally (I think?) chip in a helpful observation.

And yes, it was worth it. Because what Ralf and Steve *et al.*found is beyond belief. Ralf’s excellent post [provides all of the technical details](https://rpw.sh/blog/2015/12/21/the-backdoored-backdoor/), and you should honest just stop reading now and go read that. But since you’re still here, the TL;DR is this:

*For the past several years, it appears that Juniper NetScreen devices have incorporated a potentially backdoored random number generator, based on the NSA’s Dual\_EC\_DRBG algorithm. At some point in 2012, the NetScreen code was further subverted by some unknown party, so that the very same backdoor could be used to eavesdrop on NetScreen connections. While this alteration was not authorized by Juniper, it’s important to note that the attacker made no major code changes to the encryption mechanism — they only changed parameters. This means that the systems were potentially vulnerable to other parties, even beforehand. Worse, the nature of this vulnerability is particularly insidious and generally messed up.*

In the rest of this post I’m going to try to fill in the last part of that statement. But first, some background.

**Dual EC DRBG**

[](https://matthewdgreen.files.wordpress.com/2015/12/93863-dice.jpeg)Pretty much every cryptographic system depends on a secure [random number generator](https://blog.cryptographyengineering.com/2012/02/random-number-generation-illustrated.html), or RNG. These algorithms produce the unpredictable random bits that are consumed by cryptographic protocols. The key word in this description is *unpredictable*: if an attacker can predict the output of your RNG, then virtually everything you build on it will [end up broken](https://blog.cryptographyengineering.com/2012/03/surviving-bad-rng.html).

This fact has not been lost on attackers! The most famous (alleged) example of *deliberate* random number generator subversion was discovered in 2007 by Dan Shumow and Neils Ferguson from Microsoft, when they [announced](http://rump2007.cr.yp.to/15-shumow.pdf) the possibility of a backdoor in a NIST standard called [Dual\_EC\_DRBG](https://en.wikipedia.org/wiki/Dual_EC_DRBG).

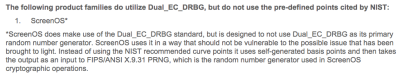
I’ve written extensively about the [design of the Dual\_EC generator](https://blog.cryptographyengineering.com/2013/09/the-many-flaws-of-dualecdrbg.html) and the [process that led to it its standardization.](https://blog.cryptographyengineering.com/2015/01/hopefully-last-post-ill-ever-write-on.html) Omitting the mathematics, the short version is that Dual EC relies on a special 32-byte constant called *Q,*which — *if generated by a malicious attacker* — can allow said attacker to predict future outputs of the RNG after seeing a mere 30 bytes of raw output from your generator.

The NIST specification of Dual\_EC comes with a default value for *Q* that was generated by the NSA. Nobody has ever been able to determine how NSA generated this value, but [leaks by Edward Snowden in 2013](http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html) provide strong evidence that NSA may have generated it maliciously. It certainly doesn’t smell good.

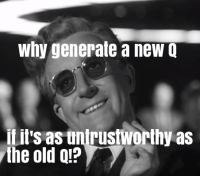
But enough with the ancient history. We’re here to talk about Juniper.

**Juniper ScreenOS — *before* the “unauthorized code”**

Although it was not widely publicized before this week, Juniper’s ScreenOS devices have used Dual EC for some time — probably since before Juniper acquired [NetScreen Technologies](https://en.wikipedia.org/wiki/NetScreen_Technologies). Prior to this week, the only place you might have learned about this was [on this tiny page](https://kb.juniper.net/InfoCenter/index?page=content&id=KB28205&pmv=print&actp=LIST) posted by the company after the Snowden leaks.

[](https://matthewdgreen.files.wordpress.com/2015/12/b32a3-whacko.png)

The decision to use Dual EC is strange all by itself. But it gets weirder.

[](https://matthewdgreen.files.wordpress.com/2015/12/33080-jtbmlh1450766921.jpg)First, ScreenOS *doesn’t* use the NSA’s default *Q.*Instead, they use an alternative *Q* value that was generated by Juniper and/or NetScreen. If Juniper had really wanted to rule out the possibility of surveillance, this would have been a good opportunity to do so. They could have generated the new constant in a “provably” safe way — *e.g.,* by hashing some English-language string. Since we have no evidence that they did so, a conservative view holds that the Juniper/NetScreen constant may *also* have been generated maliciously, rendering it useful for eavesdropping.

Next, ScreenOS uses Dual EC in a strange, non-standard way. Rather than generating all of their random numbers with Dual EC (which would be slow), they only use Dual EC to generate a seed for a fast 3DES-based generator called ANSI X9.17. Since that generator is actually [FIPS-140 approved](https://en.wikipedia.org/wiki/FIPS_140) and generally believed to be sufficient to the purpose, it’s not clear what value Dual EC is really adding to the system in the first place — except, of course, its usefulness as a potential backdoor.

The good news here is that the post-processing by ANSI X9.17 *should* kill the Dual EC backdoor, since the attack relies on the attacker seeing *raw* output from Dual EC. The ANSI generator appears to completely obfuscate this output, thus rendering Dual EC “safe”. This is indeed the argument Juniper made in 2013 when it [decided to leave the Dual EC code in ScreenOS.](https://kb.juniper.net/InfoCenter/index?page=content&id=KB28205&pmv=print&actp=LIST)

The problem with this argument is that it assumes that *no* other code could ever “accidentally” exfiltrate a few bytes bit of raw Dual EC output. Yet this is exactly the kind of threat you’d worry about in a deliberately backdoored system — the threat that, just maybe, the developers are *not your friend*. Thus Dual EC is safe only if you assume no tiny bug in the code could accidentally leak out 30 bytes or so of raw Dual EC output. If it did, this would make all subsequent seeding calls predictable, and thus render *all numbers generated by the system predictable.* In general, this would spell doom for the confidentiality of VPN connections.  
And unbelievably, amazingly, *who coulda thunk it,* it appears thatsuch a bug does exist in many versionsof ScreenOS, dating to both before and after the “unauthorized code” noted by Juniper. This issue was discovered by [Willem Pinckaers](https://twitter.com/_dvorak_) and can be illustrated by the following code snippet, which Steve Checkoway decompiled (see full context [here](http://pastebin.com/GcN8QKXT)):

void prng\_generate()  
{  
  int index; // eax@4  
  unsigned int bytes\_generated; // ebx@4  
  int v2; // eax@9  
  int v3; // eax@12  
  char v4; // ST04\_1@12  
  int time[2]; // [sp+10h] [bp-10h]@1  
  
  time[0] = 0;  
  time[1] = get\_cycles();  
  prng\_output\_index = 0; **// this is a global**  
  ++blocks\_generated\_since\_reseed;  
  if ( !do\_not\_need\_to\_reseed() )  
    **// the routine below fills a buffer with raw Dual EC output**  
    prng\_reseed(); **// internally sets prng\_output\_index to 32**  
  for ( ; (unsigned int)prng\_output\_index <= 0x1F; prng\_output\_index += 8 )  
  {  
**// this loop is supposed to process the same buffer**  
**// through the ANSI (3DES) generator. However, since the**  
**// value of prng\_output\_index was set to 32 above, it never executes**  
    memcpy(prev\_prng\_seed, prng\_seed, 8);  
    memcpy(prev\_prng\_block, prng\_block, 8);  
    ANSI\_X9\_31\_generate\_block(time, prng\_seed, prng\_key, prng\_block);  
…

Thus what comes out from this function is 32 bytes of raw Dual EC output, which is all you need to recover the internal Dual EC generator state and predict all future outputs.

**So if this was the authorized code, what the hell was the *unauthorized* code?**

The creepiest thing about CVE-2015-7756 is that there doesn’t *seem to be* any unauthorized code. Indeed, what’s changed in the modified versions is simply the value of the *Q* point. According to Ralf this point changed in 2012, presumably to a value that the hacker(s) generated themselves. This would likely have allowed these individuals to passively decrypt ScreenOS VPN sessions.

In the more recent Juniper patch to fix the vulnerability, *Q*is simply set back to the the original Juniper/NetScreen value.

The attacker also replaced some test vectors. But that appears to be it.

To sum up, some hacker or group of hackers noticed *an existing backdoor* in the Juniper software, which may have been intentional or unintentional — you be the judge! They then piggybacked on top of it to build a backdoor of their own, something they were able to do because all of the hard work had already been done for them. The end result was a period in which someone — maybe a foreign government — was able to decrypt Juniper traffic in the U.S. and around the world.

And all because Juniper had already paved the road.

**So why does this matter?**  
For the past several months I’ve been running around with various groups of technologists, doing everything I can to convince important people that the sky is falling. Or rather, that the sky *will* fall if they act on some of the *very bad, terrible*ideas that are currently bouncing around Washington — namely, that our encryption systems should come equipped with “backdoors” intended to allow law enforcement and national security agencies to access our communications.

One of the most serious concerns we raise during these meetings is the possibility that encryption backdoors could be subverted. Specifically, that a backdoor intended for law enforcement could somehow become a backdoor for people who we don’t trust to read our messages. Normally when we talk about this, we’re concerned about failures in storage of things like escrow keys. What this Juniper vulnerability illustrates is that the danger is much broader and more serious than that.

The problem with cryptographic backdoors isn’t that they’re the *only* way that an attacker can break into our cryptographic systems. It’s merely that they’re *one of the best*. They take care of the hard work, the laying of plumbing and electrical wiring, so attackers can simply walk in and change the drapes.

*This post made possible by Ralf Philipp Weinmann, H. D. Moore, Steve Checkoway, Willem Pinckaers, Nadia Heninger, Shaanan Cohney and the letter Q.*