A Few Thoughts on Cryptographic Engineering

Some random thoughts about crypto. Notes from a course I teach. Pictures of my dachshunds.

Tuesday, December 22, 2015

On the Juniper backdoor

You might have heard that a few days ago, Juniper Systems announced the discovery of "unauthorized code" in the ScreenOS software that underlies the NetScreen line of devices. As a result of this discovery, the company announced a pair of separate vulnerabilities, CVE-2015-7755 and CVE-2015-7756 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. This is a pretty fantastic vulnerability, if you measure by the impact on security of NetScreen users. But on the *technological awesomeness* scale it only rates only about a two out of ten, maybe a step above 'hit the guy with a wrench'.

The *second* vulnerability is a whole different animal. The advisory 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:

> **Matthew Green** @matthew_d_green Follow
>
> I'm really invested in the idea that this Juniper encryption vulnerability is going to be amazing. Like, Flame-level amazing.
>
> 11:30 AM - 18 Dec 2015
>
> 48 59

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, 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 and Ralf Philipp Weinmann. 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, 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

About Me

Matthew Green

I'm a cryptographer and professor at Johns Hopkins University. I've designed and analyzed cryptographic systems used in wireless networks, payment systems and digital content protection platforms. In my research I look at the various ways cryptography can be used to promote user privacy.

My website
My twitter feed
Useful crypto resources
RSS
Bitcoin tipjar
Matasano challenges

Journal of Cryptographic Engineering (not related to this blog)

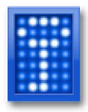
View my complete profile

Popular Posts

On the NSA
Let me tell you the story of my tiny brush with the biggest crypto story of the year . A few weeks ago I received a call from a reporter a...

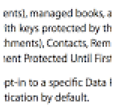
What's the matter with PGP?
Image source: @bcrypt . Last Thursday, Yahoo announced their plans to support end-to-end encryption using a fork of Google's end-to-...

Attack of the week: FREAK (or 'factoring the NSA for fun and profit')
Cryptography used to be considered 'munitions'. This is the story of how a handful of cryptographers 'hacked' the NSA....



Truecrypt report
A few weeks back I wrote an update on the Truecrypt audit promising that we'd have some concrete results to show you soon. Thanks to so...

Here come the encryption apps!
It seems like these days I can't eat breakfast without reading about some new encryption app that will (supposedly) revolutionize our c...

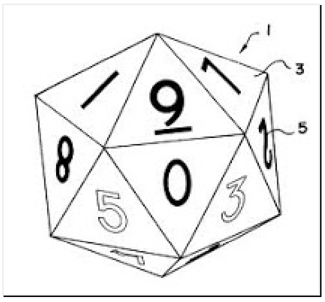


Why can't Apple decrypt your iPhone?
Last week I wrote about Apple's new default encryption policy for iOS 8 . Since that piece was intended for general audiences I mostly ...

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



Pretty much every cryptographic system depends on a secure random number generator, 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.

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 the possibility of a backdoor in a NIST standard called Dual_EC_DRBG.

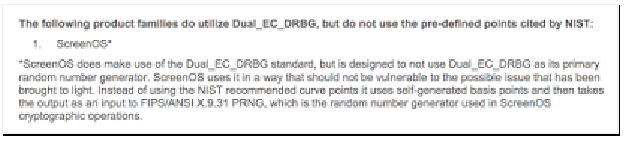
I've written extensively about the design of the Dual_EC generator and the process that led to it its standardization. Omitting the mathematics, the short version is that Dual EC relies on a special 32-byte constant called *Q*, which -- *if generated by a malicious attacker* -- can allow said attacker 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 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. Prior to this week, the only place you might have learned about this was on this tiny page posted by the company after the Snowden leaks.

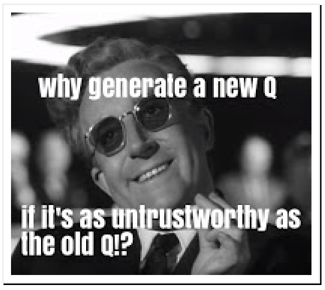


The following product families do utilize Dual_EC_DRBG, but do not use the pre-defined points cited by NIST:

1. ScreenOS*

*ScreenOS does make use of the Dual_EC_DRBG standard, but is designed to not use Dual_EC_DRBG as its primary random number generator. ScreenOS uses it in a way that should not be vulnerable to the possible issue that has been brought to light. Instead of using the NIST recommended curve points it uses self-generated basis points and then takes the output as an input to FIPS/ANSI X.9.31 PRNG, which is the random number generator used in ScreenOS cryptographic operations.

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



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 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.

The problem with this argument is that it assumes that *no* other code could every

Attack of the week: RC4 is kind of broken in TLS
Update: I've added a link to a page at Royal Holloway describing the new attack. Listen, if you're using RC4 as your primary c...

Attack of the week: OpenSSL Heartbleed
Ouch. (Logo from heartbleed.com) I start every lecture in my security class by asking the students to give us any interesting security ...

Dear Apple: Please set iMessage free
Normally I avoid complaining about Apple because (a) there are plenty of other people carrying that flag, and (b) I honestly like Apple ...

RSA warns developers not to use RSA products
In today's news of the weird, RSA (a division of EMC) has recommended that developers desist from using the (allegedly) 'backdoore...

My Blog List

Schneier on Security
"The Medieval Origins of Mass Surveillance"
17 hours ago

Bentham's Gaze | Information Security Research, University College London
Nicolas Courtois - Algebraic cryptanalysis is not the best way to break something, but sometimes it is the only option
5 days ago

ellipticnews
Asiacrypt 2015, Auckland, New Zealand
1 week ago

Shtetl-Optimized
Ask an unbounded question, get an uncomputable answer
1 week ago

Bristol Cryptography Blog
Secure Computation from Millionaire
2 weeks ago

Cryptanalysis
Bypassing certificate checks in OpenSSL 1.0.2c (CVE-2015-1793)
5 months ago

root labs rdist
Was the past better than now?
1 year ago

The MPC Lounge
5th Bar-Ilan Winter School 2015: Advances in Practical Multiparty Computation
1 year ago

Chargen

Subscribe **Followers**

Posts

Comments

"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 that such a bug does exist in many versions of ScreenOS, dating to both before and after the "unauthorized code" noted by Juniper. This issue was discovered by Willem Pinckaers and can be illustrated by the following code snippet, which Steve Checkoway decompiled (see full context here):

```
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 them to passively decrypt and ScreenOS VPN sessions they were able to eavesdrop.

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 attacker 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.

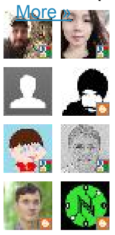
So why does this matter?

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 "back doors" 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 back door 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

Join this site
with Google Friend Connect

Members (206) More »



Already a member? Sign in

Blog Archive

- ▼ 2015 (14)
  - ▼ December (1)
    - On the Juniper backdoor
  - ► November (1)
  - ► October (1)
  - ► September (1)
  - ► August (1)
  - ► July (1)
  - ► May (1)
  - ► April (2)
  - ► March (1)
  - ► February (3)
  - ► January (1)
- ► 2014 (13)
- ► 2013 (23)
- ► 2012 (48)
- ► 2011 (39)

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 is not that they're the *only* way that an attacker can break intro 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.

Posted by Matthew Green at 3:22 AM

G+1 +19 Recommend this on Google

No comments:

Post a Comment

Enter your comment...

Comment as: ggyy (Google)

Sign out

Publish Preview ☐ Notify me

Home Older Post

Subscribe to: Post Comments (Atom)