On the Practical Exploitability of Dual EC in TLS Implementations

Stephen Checkoway, Matt Fredrikson, Ruben Niederhagen, Adam Everspaugh, Matt Green, Tanja Lange, Tom Ristenpart, Dan Bernstein, Jake Maskiewicz, Hovav Schacham

Johns Hopkins, University of Wisconsin, TU Eindhoven, University of Illinois — Chicago, UCSD
“…[Dual EC] contains a weakness that can only be seen as a backdoor.” — Bruce Schneier, 2007

Exclusive: Secret contract tied NSA and security industry pioneer

IN WAKE OF LATEST CRYPTO REVELATIONS, ‘EVERYTHING IS SUSPECT’

Exclusive: NSA infiltrated RSA security more deeply than thought - study
Randomness and Bits

- Dual EC DRBG is a deterministic random bit generator included in NIST SP 800-90 until April 2014

- A DRBG turns a small random string (a seed) into a large (approximately) random one
Dual EC — A Provably Secure DRBG

*Not actually true

- Dual *Elliptic Curve* DRBG relies on two points on an elliptic curve ($P$ and $Q$)

- Based on “elliptic curve discrete logarithm problem”: given $P$ and $Q$, find $a$ such that $Q = aP$

- But if I can choose $P$, $Q$, then I can design $F$:

$$F(\text{block}) \rightarrow \text{seed}$$

- Runs in $\sim 2^{15}$

Shumow & Ferguson, 2007
Dual EC — A Provably Secure DRBG

*Not actually true

• Dual Elliptic Curve DRBG relies on two points on an elliptic curve ($P$ and $Q$)

• Relies on "elliptic curve discrete logarithm problem": given $P$ and $Q$, find $a$ such that $Q = aP$

The question driving the controversy is: did someone choose $P$ and $Q$ so that they could design $F$?

• But if I can choose $P$, $Q$, then I can design $G$:

  $G(P, Q, \text{block}) \rightarrow \text{seed}$

  Runs in $\sim 2^{15}$

Random Data

Seed for next time
Why is this a problem?

**Client**
- Generate client random
- Generate b

**Server**
- Generate session ID, server random, a, signature nonce
- bP, Finished
- Finished

Master Secret = PRF(x(abP), “master secret”, client || server random)

**DualEC**
- $\rightarrow$ [server random]

**F(server random)**
- $\rightarrow$ seed

**DualEC**
- $\rightarrow$ [ ]

Attacker and Server do the same thing
Maybe not?

- In reality, the server might...
  - Not release enough Dual EC output (we need 30 bytes)
  - Mix other entropy into Dual EC
  - Re-seed often
  - Share state with other sessions
  - Implement the spec wrong
Testing Practical Exploitability

• We tested RSA’s BSAFE, Microsoft’s SChannel, OpenSSL-FIPS

• Focused on server, ECDHE cipher suites

• We *assume* the attacker has $F$, sees network traffic

• Modified each implementation to use $P, Q$ such that we know $F$

• Significant amount of reverse-engineering on SChannel and BSAFE
Results: OpenSSL

• We found: OpenSSL is impossible to exploit!

List: openssl-announce
Subject: Flaw in Dual EC DRBG (no, not that one)
From: Steve Marquess <marquess ( ) opensslfoundation ! com>
Date: 2013-12-19 14:08:20
Message-ID: 52B2FDD4.5080701 () opensslfoundation ! com
[Download message RAW]

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is an unusual bug report for an unusual situation. I'm using it as an opportunity to point out some considerations that have not been widely reported.

The nature of the bug shows that no one has been using the OpenSSL Dual EC DRBG.

Given the current status of Dual EC DRBG (now disowned[1] by the NIST CMVP[2] and pretty much toxic for any purpose) we do not plan to correct the bug. A FIPS 140-2 validated module cannot be changed
Results: OpenSSL

- Uses Dual EC to generate a 32-byte session ID
- ...but it also uses 20—45 bits of other entropy per call
  - \(\text{time secs} \|| \text{time \(\mu\text{secs}\)} \|| \text{counter} \|| \text{PID}\)
- Complexity depends on guessability of counter, PID
- If we know (time secs, counter, PID), attack runs in 1.2 seconds
Results: SChannel

- Only sends 28 bytes of Dual EC output
- Does not use additional entropy or re-seed
- Also has an implementation flaw
  - …that happens to make the attack slightly faster
- Leveraging attack on future sessions more difficult
- Attack possible in ~63 minutes
Results: BSAFE

- Dual EC is the default RNG!
- Does not use additional entropy or re-seed often
- BSAFE-C caches unused outputs
- Releases variable amounts of output:
  - 31-60 bytes for BSAFE-C
  - 28 bytes for BSAFE-Java

BSAFE-C: 2.4 seconds
BSAFE-Java: 64 minutes

Good enough for massive surveillance
Fine for targeted sessions
Conclusions

• Dual EC *might* contain a backdoor

  • Need concrete evidence that someone knows $F$

• If it does, they should hire better hackers

  • Fragile to low-level implementation choices

  • Only works for certain types of protocols

• Design was suspicious from the start

• **Read the full paper at dualec.org**