Challenges and Implications of Verifiable Builds for Security-Critical Open-Source Software

Xavier de Carné de Carnavalet          Mohammad Mannan

Concordia University, Montreal, Canada
Motivation

1. TrueCrypt:
   - End of volume: 64k of encrypted random data (Win) vs. 64k of encrypted zeros (Linux)
   - Reverse-engineering is hard
   - Authors are anonymous
Motivation

1. **TrueCrypt:**
   - End of volume: 64k of encrypted random data (Win) vs. 64k of encrypted zeros (Linux)
   - Reverse-engineering is hard
   - Authors are anonymous

2. **NSA surveillance programs, Torvalds and backdoors**
Motivation

1. TrueCrypt:
   - End of volume: 64k of encrypted random data (Win) vs. 64k of encrypted zeros (Linux)
   - Reverse-engineering is hard
   - Authors are anonymous

2. NSA surveillance programs, Torvalds and backdoors

3. IsTrueCryptAuditedYet? (OCAP)
   - “Perform and document repeatable, deterministic builds of TC 7.1a from source code […]”
Contributions

1. Positively match the binary files of TrueCrypt for Windows v5.0–7.1a (16 versions) with the available source code
2. Uncover several issues leading to non-determinism through the compilation of TrueCrypt
3. Clarify “deterministic”, “reproducible” and verifiable builds
4. Summarize lessons learned for open-source software
Non-deterministic build process
Non-deterministic build process
Non-deterministic build process
Verifiable build
A build is verifiable if any two instances of the build process produce identical results.

How?
1. Deterministic process: builds are byte-by-byte identical
2. Matching builds at a higher semantic level (by ignoring unimportant differences)
Signatures with deterministic builds

1. Same source $\rightarrow$ same output files $\rightarrow$ same hashes
2. Let’s sign the hashes and publish them
3. Advanced users can contribute to the signatures list
   - Recompile and publish the signed hashes
4. Normal users compare their hashes with the list
   - No recompilation
   - Simple hash comparison (by majority? trusted signers?)
Implications

Verifiability provides end-users with defenses against:

1. **Targeted attacks on binaries**
   - MiTM when downloading the files
2. **Untrusted authors**
   - Benign source and malicious binaries
TrueCrypt

- is an encryption software (AES, Serpent, Twofish)
- protects data by password/keyfile/token
- provides Full-Disk Encryption (FDE)
- supports Plausible Deniable Encryption (PDE)
- is cross-platform and portable

We analyze v5.0–7.1a (16 versions, 2008–2012).
Where could we find backdoors?
Where could we find backdoors?

NOT CONSIDERED
(source code audit)
Where could we find backdoors?

NOT CONSIDERED
(trusting trust attack)
Where could we find backdoors?

CONSIDERED
(untrusted authors threat)
Requirements (v7.1a)

1. Microsoft Visual C++ 2008 SP1 (Professional Edition or compatible)
2. Microsoft Visual C++ 1.52 (available from MSDN Subscriber Downloads)
3. Microsoft Windows SDK for Windows 7 (configured for Visual C++)
4. Microsoft Windows Driver Kit 7.1.0 (build 7600.16385.1)
5. RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) 2.20 header files
6. NASM assembler 2.08 or compatible
7. gzip compressor
8. (dd tool)
Naive approach

Requirements are old, let’s use the most recent version of the tools!
Naive approach

Requirements are old, let’s use the most recent version of the tools!

Wrong! 57% of difference* by using VS2012

* over the 4 unpacked binaries altogether (3.42MiB)
Let’s download exactly what is written in the Readme file so as to be close to the developers environment.
Forensic approach

Let’s download exactly what is written in the Readme file so as to be close to the developers environment.

Almost! Still 3.1% of difference :(
Forensic approach: updates matter

Microsoft Visual Studio 2008 SP1 updates:

- KB971092 (2009-08-03)
- KB973675 (2009-08-10)
- KB972222 (2009-10-12)
- KB2538241 (2011-06-07)
- KB2669970 (2012-03-12)
Forensic approach: updates matter

Microsoft Visual Studio 2008 SP1 updates:

- KB971092 (2009-08-03)
- KB973675 (2009-08-10)
- KB972222 (2009-10-12)
- KB2538241 (2011-06-07)
- KB2669970 (2012-03-12) TrueCrypt 7.1a was released on 2012-02-07
Comparing the binaries

Less than 0.9% of difference (mostly the embedded certificates)
Sources of non-determinism (1/2)

1. Timestamps (PE header, resources, ...)
2. Binaries checksums
3. Debug info (GUID)
4. Build path
5. Signatures (embedded X.509 certificate)
Sources of non-determinism (2/2)

6. Setup package’s custom integrity checksum
7. Processor-specific NOP alignment instructions
   - AMD:
     66 66 90  data32 xchg ax,ax
     66 90  xchg ax,ax
   - Intel:
     0F 1F 44 00 00  nop DWORD PTR [rax+rax*1+0x0]
8. Not-so-precise info in readme file (exact VS updates)
9. Missing info in readme file (dd tool)
10. Wrong info in readme file (non-working version of NASM)
11. Incomplete code (repackaged version v5.1a-bis)
How I compiled TrueCrypt 7.1a for Win32 and matched the official binaries

TrueCrypt is an open-source encryption software capable of on-the-fly encryption on file-, partition- or disk-based virtual disks. It supports various ciphers, including AES, Serpent, Twofish or some combination of them; provides a full disk encryption (FDE) feature under Windows environment with pre-boot authentication; and even allows plausible deniability.

Hence TrueCrypt seems to be a perfect solution to protect sensitive files. However, the recent news about the NSA programs enables all conspiracy theorists to imagine the worst of all. What if TrueCrypt was backdoored? What if the binaries provided on the website were different than the source code and they included hidden features?

We show in this article how to reproduce a deterministic compilation process specific to TrueCrypt 7.1a for Windows that matches the official binaries, and relieve the world from at least some concerns.

Article versions changelog

- 2013-10-27: Added in appendix the checksums of the files downloaded in this analysis, provided better understanding of the installer checksum difference, verified MVC++ 1.52 found on the web with the original, fixed typos
- 2013-10-24v2: Clarified few sentences about backdoors, explained the PDB info difference, made clear my results are meant to be reproduced
- 2013-10-24: Added analysis results of v7.0a and v6.3a
- 2013-10-23: Explained differences in more details, added assembly comparison
- 2013-10-22: Added PGP/X.509 screenshots, clarified some comparison comments
- 2013-10-21: First version
45K visitors within 2d after Slashdot post
Important visitors within 2 days

Submitted on Slashdot

DoD

NIST

Posted on Slashdot

US Department of State

US Senate

US Securities & Exchange Commission

UTC-4

12pm 3pm 6pm 9pm 12am 3am 6am 9am 12pm 3pm 6pm

24 Oct 2013

25 Oct 2013
Current approaches for the future (1/3)

Gitian for Tor

- Originally developed for the Bitcoin Core client
- Controlled VM with input and output descriptors for files and packages to be used
- Signs output files
- “Find a way to control the environment”
Current approaches for the future (2/3)

Debian (for its packages)
- Custom toolchain for reproducible builds
- Stop including timestamps, ...
- 13213 (61.4%) of 21448 packages are reproducible (2014-11-11)
- “Treat the problem at the root”
Current approaches for the future (3/3)

Fedora & OpenSUSE
- Scripts to compare builds (decompose them, handle file-type-specific comparisons)

Initiatives
- Reproducible Build Manager (RBM)
- NixOS
- Firefox
Concluding thoughts

1. Security by randomization vs. determinism?
2. Which approach for future verifiable builds?
3. How to automate verification of old software?
4. Is finding nothing, something?

Questions?

x_decarn@ciise.concordia.ca
https://madiba.enscs.concordia.ca