CS380S: Project Proposal

Rushi Shah Andrew Russell

1 Project Idea

1.1 Motivation

One common misuse of cryptography is the misuse of entropy. Without proper random inputs, many cryptographic algorithms are vulnerable to basic forms of cryptanalysis. Some cryptographic schemes, such as DSA, can even disclose long-term secrets like the signing key when the random per-message input is low entropy, made public, or nonunique.

We have two motivating examples. First, we would like our tool to be able to detect the issue with the Debian/OpenSSL pseudo-random number generator that was exposed in 2008. Second, we would like to identify potential vulnerabilities in current cryptocurrency wallet code as many cryptocurrency protocols rely on DSA.

1.2 Goals

We plan to use data dependency tools to determine how entropic inputs in a given program are used by various cryptographic algorithms. That will allow us to identify if and when entropy is too low or is misused. We plan to either produce this as a code integration tool for developers to use as part of a compiler toolchain, or use this tool to analyze a large number of codebases found "in the wild," such as those written by amateurs.

Our tool will seek to generate a human-checkable dependency graph from source code using taint analysis on functions and function inputs manually specified via annotation by the developer. This graph will allow the developer to manually verify that entropy is used properly; a stretch goal of ours would be to automate the identification of a subset of known misuses of entropy.

2 Rough Plan

1. Determine data dependency tool (for C/C++) to use, conduct background research (*2 weeks*)
2. Adapt tool to identify the OpenSSL bug (*5 weeks*)
3. Test on other cases like Bitcoin wallets (*3 weeks*)
4. Prepare presentation/writeup (*1 week*)
5. Integrate tool into compiler toolchain like clang (*optional*)

3 Research Hypotheses

We principally have two hypotheses we would like to test.

1. An automated tool can detect entropy bugs in real-world programs.
2. Entropy is insufficiently propagated in programs that rely on cryptography.

4 Related Work

Static code analysis has a history of identifying security vulnerabilities at a source code level. Some examples include SQL injection, cross-site scripting exploits, and buffer overflow attacks. However, there has not been any attempt in the literature to statically analyze source code for cryptographic vulnerabilities stemming from entropy misuse, which we seek to do.

1. Debian/OpenSSL Bug
   (a) `https://www.schneier.com/blog/archives/2008/05/random_number_b.html`
   (b) `https://research.swtch.com/openssl`

   (c) `https://freedom-to-tinker.com/2013/09/20/software-transparency-debian-openssl-bug/`

   (d) `https://www.cs.umd.edu/class/fall2017/cmsc818O/papers/private-keys-public.pdf`

2. Data flow

   (a) `https://en.wikipedia.org/wiki/Data-flow_analysis`

   (b) `https://www.seas.harvard.edu/courses/cs252/2011sp/slides/Lec02-Dataflow.pdf`

3. Static Program Analysis

   (a) `https://cs.au.dk/~amoeller/spa/spa.pdf`

   (b) `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6859783`