Selfrando

Securing the Tor Browser against De-anonymization Exploits

Krishnakumar.C.N
August 24,2016
Roll No.:34

Department of Computer Science & Engineering,
College of Engineering, Trivandrum.

Overview

The Onion Router

What is Tor?

The Onion Router

- The Tor network is a group of volunteer-operated anonymity network that allows people to improve their privacy and security on the Internet.
- The Tor Network operates through a number of organizations and individuals who donate their internet bandwidth, processing power, and technical expertise.

Evolution of Onion Routing Project

1995- (Research)
1996 -2001 (Development)

Electronic Frontier Foundation
2004 - 2005

2006 - Present

TorProject.org

- A non-profit organization focused on research and education, 501(c)(3),since 2006
- Major contributors
  - U.S Department of Defense
  - U.S Broadcasting Board of Governors
  - U.S National Science Foundation (NSF)
- Annual Budget = $3 Million
  - 80% from United States Government
  - 20% Swedish Government and other Organizations

Need for Tor

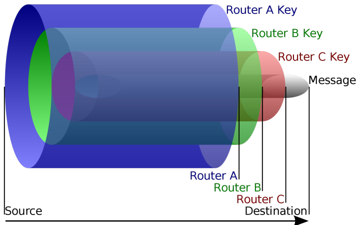
- Protection from traffic analysis
- Online Anonymity tool
    - Marketing based on browsing data
    - Censored internet access
    - Protection of Confidential sources
- Anonymity isn't Encryption.

Users of Tor

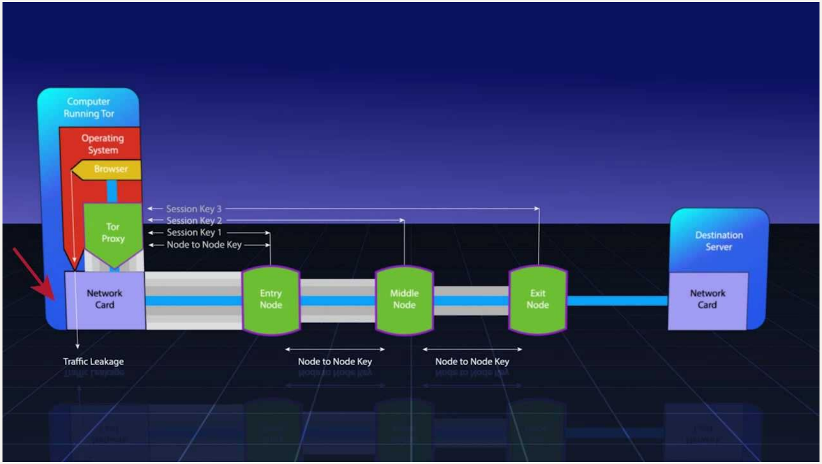
Military – Journalists – Activist – Human Right Groups – Foreign Embassies – Criminals – Normal Users

How Tor Works

- Wrapping data with consecutive layers of encryption that can be peeled back as it passes through a series of proxy computers around the world
- Each layer peeled contains information on where to send next
- The only link in the chain where the data is un-encrypted is between the exit node and the receiver



How Tor Works



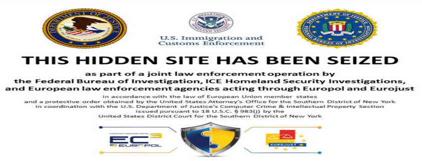
De-anonymization Attacks

De-anonymization Attack

A de-anonymization attack aims to disclose information, such as the identity or the location, of an anonymous user

- Weaknesses in the network protocol.
- Exploit security vulnerabilities in the software used to access the Tor network.

Major Attacks


U.S. Immigration and Customs Enforcement

THIS HIDDEN SITE HAS BEEN SEIZED

as part of a joint law enforcement operation by
the Federal Bureau of Investigation, ICE Homeland Security Investigations,
and European law enforcement agencies acting through Europol and Eurojust
in accordance with the law of European Union member states
and a protective order obtained by the United States Attorney's Office for the Southern District of New York
in coordination with the U.S. Department of Justice's Computer Crime & Intellectual Property Section
issued pursuant to 18 U.S.C. § 983(j) by the
United States District Court for the Southern District of New York

- 2013 FBI exploit using vulnerability on Firefox

  -Code Name:EGOTISTICALGIRAFFE
- Carnegie Mellon University researchers paid by FBI

  - The payment is about $1 Million
- Attack on SilkRoad, SilkRoad 2.0,Playpen portal

```
Yes, bad people need anonymity too. But they are
  already doing well. Honest people need more
      security, privacy,and anonymity.
```

Memory Corruption Exploits

- Buffer Overflow Attack
  - Inject malicious code into program and execute it
  - Not used anymore due to $W \oplus X$ policy
- Code Reuse Attack

Code Reuse Attack

- Reuse existing,legitimate code for malicious purpose
- Much harder attack than code injection
- Two types:
  1. Return-into-libc
  2. Return-oriented-programming
- Requirements:
  - Application has a Memory corruption vulnerability
  - Knowledge of the absolute address of the gadgets

Code Reuse Exploit Prevention

Adreess Space Layout Randomization

- Randomly choose base address of stack, heap, code segment
- Only randomizes the base address of the binary
- Easy to compute absolute addresses with a leaked code pointer

Address Space Layout Permutations

- Permuting the functions inside a library
- Improves Entropy

Selfrando

Selfrando

An enhanced and practical load-time randomization technique that defends against de-anonymization exploits.

Objectives

- Substantially raise the costs for attackers to exploit memory-corruption vulnerabilities.
- Support complex C/C++ programs without modifying their source code
- Full compatibility with current build systems
- Avoid any modification to compilers, linkers, and other operating system component
- Should not substantially increase the size of the program in memory or on disk.

Design

Compile Time Randomization

- Impractical for introducing diversity
- Randomized builds complicate the deterministic build process
- Increase the feasibility of a de-anonymization attack

Load Time Randomization

- Only need to compile and test a single binary
- Users can verify the authenticity of the downloaded binary
- Support Deterministic build process

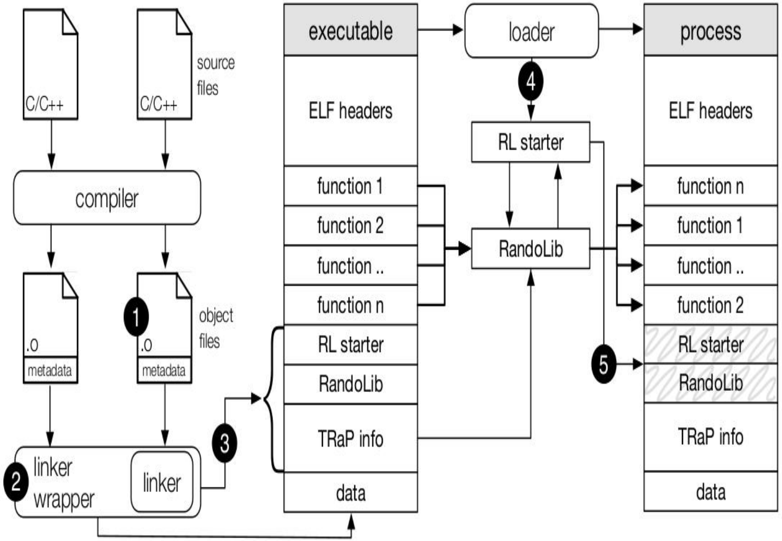
Framework

- Randomization granularity - Function Permutation
  - Increase entropy
  - Low overheads
- Translation and Protection (TRaP)
  - Load-time function boundary discovery is unreliable
  - Pre-compute function boundaries statically and store the necessary information in each binary
- Linker wrapper: wraps the system linker,extracts all function boundaries from the object files used to build the binary, then appends the necessary TRaP information and RandoLib to the binary itself

Implementation

Extracting TRaP Information

- Selfrando requires accurate information about function boundaries
- When a function moved
    - All references and pointers to it become invalid
    - Needs to update all references to the moved function
    - Complete list of all locations that reference a function is needed
- Intermediate object file contains these information.
- The Linker Wrapper intercept the linking process to extract the TRaP information
- Direct jumps between two functions
- Add a linker argument to generate a map file
- Map file with metadata give final locations and references



C/C++
C/C++
source files
compiler
.o
metadata
.o
metadata
object files
1
2
linker wrapper
linker
3
executable
ELF headers
function 1
function 2
function ..
function n
RL starter
RandoLib
TRaP info
data
loader
4
RL starter
RandoLib
5
process
ELF headers
function n
function 1
function ..
function 2
RL starter
RandoLib
TRaP info
data

Embedding TRaP Information

Embedding the data in a space efficient and binary format compatible way without modifying the linker is challenging

- Some metadata only available after the completion of linking
- Cannot pre-allocate space for the data as space needed is unknown until completion of linking
- Add an empty segment header in the beginning of the binary.
- When the linker finished, we append TRaP and RandoLib to the end of the binary and update the header
- Finally, change the starting address to RandoLib

Load-Time Function Permutation

- Function permutation performed by RandoLib
- Fisher-Yates shuffling algorithm for permutation
- RandoLib fix all references
- After RandoLib returns, the helper stub (RL Starter) makes selfrando’s data inaccessible and jumps to the original entry point

Stack Unwinding

SELFRANDO needs to support stack unwinding

- Iterate through active stack frames
- Used for stack traces,exception handling etc.
- Linked list method traditionally
- Modern compiler generate additional metadata.
- Function permutations invalidate the metadata
- RandoLib updates them

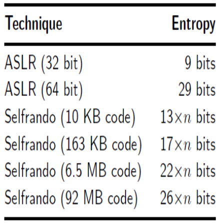
AddressSanitizer

- Tool to detect memory corruption bug
- Stack trace is generated when detects a memory corruption
- Symbolizer stores information-function name and address
- Randomization creates problem
- Selfrando emits a map file and Symbolizer is modified to use it

Experimental Evaluation

Security Analysis

Randomization Entropy



Technique	Entropy
ASLR (32 bit)	9 bits
ASLR (64 bit)	29 bits
Selfrando (10 KB code)	$13 \times n$ bits
Selfrando (163 KB code)	$17 \times n$ bits
Selfrando (6.5 MB code)	$22 \times n$ bits
Selfrando (92 MB code)	$26 \times n$ bits

Table 1. Randomization entropy of ASLR and selfrando for different address space sizes and function counts. For selfrando, we report the number of bits the attacker needs to guess for each function address the attacker needs.

Debian 8.4 machine using GCC 6.1.0 and Clang 3.5.0

Security Analysis

Real-world Exploits

- All modern attacks exploit heap-based vulnerabilities
- ROP payload on the heap is executed using stack pivot gadgets
- Stack pivots available through virtual functions
- Objects on the heap —> virtual table —> virtual functions
- Virtual table not on heap,so cannot disclose the gadget addresses
- When only ASLR is applied, the address of the virtual table randomized with the same offset as the ROP gadgets.Such an attack can bypass aslr but not selfrando

Performance Overhead

Average Load-time

- Normal version: 2.046s
- Selfrando version: 2.400s

Run-time overhead

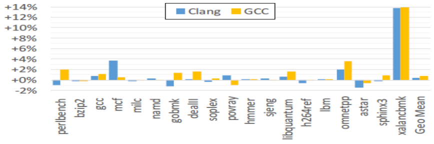


Fig. 4. Run time overhead on the benchmarks in the SPEC CPU2006 suite (full selfrando).

- Average case: 2.02%
- Worst case 2.5%

Conclusion

Summary

- Securing the Tor network from de-anonymization attack is necessary to preserve privacy.
- SELFRANDO, an enhanced and practical load-time randomization technique for the Tor Browser defends against de-anonymization exploits.
- It has negligible run-time overhead, and a perfectly acceptable load-time overhead.
- Selfrando significantly improves security over standard address space layout randomization (ASLR) techniques currently used.

References

Mauro Conti, Stephen Crane, Tommaso Frassetto, Andrei Homescu, Georg Koppen, Per Larsen, Christopher Liebchen, Mike Perry, and Ahmad-Reza Sadeghi.
Selfrando: securing the tor browser against de-anonymization exploits.
The annual Privacy Enhancing Technologies Symposium in Darmstadt, Germany, 2016.

C.Kil, J.Jun, C.Bookholt, J.Xu, and P.Ning.
Address space layout permutation (aslp): Towards fine-grained randomization of commodity software.
22nd Annual Computer Security Applications Conference, 2006.

Erik Buchanan, Ryan Roemer,Stefan Savage, Hovav Shacham.
Return-oriented programming: Exploitation without code injection.
University of California, San Diego.

Tor overview.
https://www.TorProject.org.

Selfrando: Q and a with georg koppen.
http://www.TorProject.org.

Did the fbi pay a university to attck tor users?
https://blog.torproject.org/blog/did-fbi-pay-university-attack-tor-users.

How tor works – a compute cycle deep dive.
https://www.excivity.com/ComputeCycle/howtorworks/.

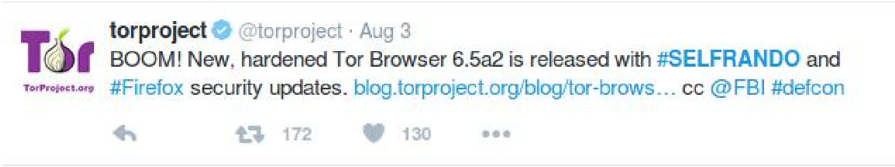
M.Tran, M.Etheridge, T.Bletsch, X.Jiang, V.W.Freeh, and P.Ning.
On the expressiveness of return-into-libc attacks.
14th International Symposium on Research in Attacks, Intrusions and Defenses, 2011.

Data execution prevention (DEP).

Microsoft.
http://support.microsoft.com/kb/875352/EN-US/.

Konstantin Serebryany, Derek Bruening, Alexander Potapenko, Dmitry Vyukov.
Addresssanitizer: A fast address sanity checker.

, Ben Pfaff, Eu-Jin Goh, Nagendra Modadugu, Dan Boneh Hovav Shacham, Matthew Page.
On the effectiveness of address-space randomization.


torproject @torproject · Aug 3

BOOM! New, hardened Tor Browser 6.5a2 is released with #SELFRANDO and #Firefox security updates. blog.torproject.org/blog/tor-brows… cc @FBI #defcon

THANK YOU!