Twitter Profile
GitHub Profile
Google+ Profile
Linkedin Profile
RSS feeds
Email

About me

I'm just a guy who has a strong interest in low-level computing, program analysis and software verification methods. My *"research"* is mostly focused on both static and dynamic program analysis. I also enjoy doing things in various fields like: OSdev, reverse engineering, bugs exploitation and others low-level stuffs. You will find the results of my different researches on this blog.

Short blog posts

Binary analysis

- 2016-10-28 - Automatic deobfuscation of the Tigress binary protection using symbolic execution and LLVM
- 2016-05-18 - What kind of semantics information Triton can provide?
- 2015-10-12 - Code coverage using a dynamic symbolic execution
- 2015-06-10 - Triton (concolic execution framework) under the hood
- 2013-10-14 - Stack and heap overflow detection at runtime via behavior analysis and Pin
- 2013-08-28 - Binary analysis: Concolic execution with Pin and z3
- 2013-08-17 - In-Memory fuzzing with Pin
- 2013-08-08 - Taint analysis and pattern matching with Pin
- 2013-06-10 - Concolic execution - Taint analysis with Valgrind and constraints path solver with Z3
- 2013-05-02 - A binary analysis, count me if you can

Linux Kernel

- 2013-05-26 - Trace and debug the Linux kernel functions
- 2013-03-25 - Linux Kernel runtime unpacker and binary signature
- 2013-01-29 - Linux process execution and the useless ELF header fields
- 2013-01-23 - Simple hook detection Linux module
- 2013-01-19 - ASLR implementation in Linux Kernel 3.7

Operating System

- 2013-02-03 - Physical page frame allocation with bitmap algorithms
- 2012-12-25 - Paging modes for the x86 32-bits architectures
- 2012-12-24 - Classical memory access optimization with the TLB

CTF Write-up

- 2017-02-06 - Hackover 2015 r150 *(outdated solving for Triton use cases)*
- 2016-08-02 - Defcamp 2015 r100 *(outdated solving for Triton use cases)*
- 2016-08-01 - Defcon Quals 2016 baby-re *(outdated solving for Triton use cases)*
- 2013-06-23 - Write-up NDH2k13 Final K1986
- 2012-06-04 - Defcon 20 quals 2012 - Forensic 400
- 2012-06-04 - Defcon 20 quals 2012 - Forensic 300
- 2011-09-21 - Hack.lu 2011 Antique Space Shuttle (300)
- 2011-05-30 - RSSIL 2011 - RCE encrypted file
- 2011-05-29 - RSSIL 2011 - RCE chimay_rouge
- 2011-04-25 - PlaidCTF 2011 - Another small bug
- 2011-04-25 - PlaidCTF 2011 - Calculator
- 2011-03-06 - Insomni'hack 2011 - Reverse 2

CVE exploitation

- 2011-07-04 - Analysis of CVE-2011-1938 - ROP exploitation in PHP 5.3.6

MISC

- 2011-10-02 - Polymorphism and Return Oriented Programming
- 2011-04-12 - Return Oriented Programming and ROPgadget tool
- 2010-11-25 - Shellcode on ARM architecture

Online services

- Online Assembler and Disassembler
- Shellcodes database for study cases

Presentations and publications

- **Deobfuscation of VM based software protection**
  Talk at SSTIC, Rennes, 2017. [french paper] [english slide] [french video] [bibtex]
  Abstract: *In this presentation we describe an approach which consists to automatically analyze virtual machine based software protections and which recompiles a new version of the binary without such protections. This automated approach relies on a symbolic execution guide by a taint analysis and some concretization policies, then on a binary rewriting using LLVM transition.*

- **How Triton can help to reverse virtual machine based software protections**
  Talk at CSAW SOS, NYC, 2016. [slide]
  Abstract: *The first part of the talk is going to be an introduction to the Triton framework to expose its components and to explain how they work together. Then, the second part will include demonstrations on how it's possible to reverse virtual machine based protections using taint analysis, symbolic execution, SMT simplifications and LLVM-IR optimizations.*

- **Dynamic Binary Analysis and Obfuscated Codes**
  Talk at St'Hack, Bordeaux, 2016. [slide]

Abstract: *At this presentation we will talk about how a DBA (Dynamic Binary Analysis) may help a reverse engineer to reverse obfuscated code. We will first introduce some basic obfuscation techniques and then expose how it's possible to break some stuffs (using our open-source DBA framework - Triton) like detect opaque predicates, reconstruct CFG, find the original algorithm, isolate sensible data and many more... Then, we will conclude with a demo and few words about our future work.*

- **How Triton may help to analyse obfuscated binaries**
  MISC magazine 82, 2015. [french article]
  Abstract: *Binary obfuscation is used to protect software's intellectual property. There exist different kinds of obfucation but roughly, it transforms a binary structure into another binary structure by preserving the same semantic. The aim of obfuscation is to ensure that the original information is "drown" in useless information that will make reverse engineering harder. In this article we will show how we can analyse an ofbuscated program and break some obfuscations using the Triton framework.*

- **Triton: A Concolic Execution Framework**
  Talk at SSTIC, Rennes, 2015. [french paper] [detailed english slide] [light french slide] [bibtex]
  Abstract: *This talk is about the release of Triton, a concolic execution framework based on Pin. It provides components like a taint engine, a dynamic symbolic execution engine, a snapshot engine, translation of x64 instruction to SMT2, a Z3 interface to solve constraints and Python bindings. Based on these components, Triton offers the possibility to build tools for vulnerabilities research or reverse-engineering assistance.*

- **Dynamic Behavior Analysis Using Binary Instrumentation**
  Talk at St'Hack, Bordeaux, 2015. [slide]
  Abstract: *This talk can be considered like the part 2 of my talk at SecurityDay. In the previous part, I talked about how it was possible to cover a targeted function in memory using the DSE (Dynamic Symbolic Execution) approach. Cover a function (or its states) doesn't mean find*

all vulnerabilities, some vulnerability doesn't crashes the program. That's why we must implement specific analysis to find specific bugs. These analysis are based on the binary instrumentation and the runtime behavior analysis of the program. In this talk, we will see how it's possible to find these following kind of bugs : off-by-one, stack / heap overflow, use-after-free, format string and {write, read}-what-where.

- **Covering a function using a Dynamic Symbolic Execution approach**
  Talk at Security Day, Lille, 2015. [slide]
  Abstract: *This talk is about binary analysis and instrumentation. We will see how it's possible to target a specific function, snapshot the context memory/registers before the function, translate the instrumentation into an intermediate representation,apply a taint analysis based on this IR, build/keep formulas for a Dynamic Symbolic Execution (DSE), generate a concrete value to go through a specific path, restore the context memory/register and generate another concrete value to go through another path then repeat this operation until the target function is covered.*

- **An introduction to the Return Oriented Programming and ROP-chain generation**
  Course lecture at Bordeaux University, 2014. [slide]
  Abstract: *This course lecture is about an introduction to the return oriented programming and its variants like JOP, SOP... It also describes some techniques and active works about ROP-chain generation based on backtracking or symbolic execution.*

- **An introduction to the Return Oriented Programming**
  MISC Magazine HS-09, 2014. [link]
  Abstract: *This article is about an introduction to the return oriented programming. In this article we describe the Operating Systems' protections in detail and explains why the ROP exploitation is useful. We also provides a detailled step-by-step example of the ROP exploitation on the CVE-2011-1938 vulnerability.*

- **Software testing and concolic execution**
  Talk at LSE Summer Week, Paris, 2013. [slide]
  Abstract: *This talk is about an introduction to the concolic execution using Valgrind and Z3. Concolic execution is a technique that uses both symbolic and concrete execution. In this talk we introduce a little tool which breaks a dumb crackme.*

Mini projects

- **Triton - A Dynamic Binary Analsyis Framework**, *2015-06-03*
  Abstract: *Triton is a dynamic binary analysis (DBA) framework. It provides internal components like a Dynamic Symbolic Execution (DSE) engine, a Taint Engine, an intermediate representation based on SMT2-Lib of the x86 and x86-64 instructions set, SMT simplification passes, an SMT Solver Interface and, the last but not least, Python bindings. Based on these components, you are able to build program analysis tools, automate reverse engineering and perform software verification. .*
- **Kaminou-Kernel - Another (unfinished) mini Kernel from scratch**, *2012-11-11*
  Abstract: *Just another (unfinished) kernel from scratch... At first I didn't plan on releasing this project, but it's dying in a directory. This project was just for understanding how the kernel works and contains these following features: Protected mode 32 bits, Multitasking, Paging, Memory Allocation, Interrupts, Exceptions, Syscalls.*
- **Useless emulator for fun (VMNDH-2k12)**, *2012-03-26*
  Abstract: *This emulator is totally useless, but it was created for the CTF NDH 2012. Some challenges was on the NDH architecture. The NDH architecture is a new architecture which look like a mix between ARM and x86. The project contains a compiler, a debugger and a virtual machine.*

- **ROPgadget - Gadgets finder and auto-roper**, *2011-03-12*
  Abstract: *This tool lets you search your gadgets on your binaries to facilitate your ROP exploitation. ROPgadget supports the ELF, PE and Mach-O format on x86, x64, ARM, ARM64, PowerPC, SPARC and MIPS architectures. It also offers a dumb ROP-chain generation.*

Vulnerabilities publicly disclosed

- CVE-2015-1801 - Samsung S4 (GT-I9500) - Multiple Kernel memory corruption in the video driver
- CVE-2015-1800 - Samsung S4 (GT-I9500) - Kernel memory disclosure in the video driver
- CVE-2013-6392 - Kernel MSM < 3.10 - Kernel memory disclosure in the Genlock driver
- CVE-2013-6122 - Goodix gt915 Android touchscreen driver - Kernel race condition
- CVE-2013-4740 - Goddix gt915 Android touchscreen driver - Multiple Kernel memory corruption
- CVE-2013-4739 - Android Kernel msm-3.4/jb_3* - Kernel memory disclosure in the Gemini JPEG/Jpeg1.0 engines
- CVE-2013-4738 - Android Kernel msm-3.4/jb_3* - Kernel Stack overflow in camera post processing driver (CPP)
- CVE-2013-2239 - OpenVZ kernel 2.6.32 (042stab080.1) - Multiple kernel memory disclosure
- CVE-2013-2164 - Linux Kernel 3.9.5 - Kernel memory disclosure in cdrom driver
- CVE-2009-4800 - Sysax FTP server 4.5 - DELE request handling Traversal arbitrary file deletion
- CVE-2009-1031 - Serv-U FTP server 7.4 - MKD request handling Traversal arbitrary directory creation