Fuzzing and TaintScope

Kalle Kosteila

Helsinki 11.04.2011

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos
**Fuzzing and TaintScope**

Fuzzing is a very effective tool for testing software security. There are many fuzzing methods, and each method has many types. Fuzzing is not limited to just testing but also plays a very important part of fuzzing. Different fuzzing methods and their types have many advantages and disadvantages. To be effective, a fuzzer needs to be automatic, and monitoring also needs to be automatic.

TaintScope is a checksum-aware directed fuzzing tool which goes into the in-memory fuzzing area. In TaintScope, everything is done automatically, and that is why it is said to be very effective. Anyway, I didn't find any public software version that this paper is based on just reading the TaintScope report.

**ACM Computing Classification System (CCS):**

D.4.6 [Security and Protection]

---

**Keywords**

Fuzz, Fuzzing
**Contents**

Introduction ....................................................................................................................... 4  
Fuzzing ............................................................................................................................ 5  
Fuzzing methods and types ............................................................................................. 6  
  Methods ......................................................................................................................... 6  
  Types ............................................................................................................................. 7  
TaintScope ......................................................................................................................... 9  
  Fuzzing result .............................................................................................................. 9  
Conclusion ....................................................................................................................... 10  
References ...................................................................................................................... 11
Introduction

Security testing is done in many different ways and methods. Which method is actually chosen depends who does testing. In this paper I concentrate fuzzing technics. At first I explain what fuzzing is and what kind of fuzzing methods there are. Last part of paper concentrate to the taintscope and what it is. There is white box, grey box and black box testing areas.

In white box testing source code is available and basically vulnerabilities are find from source code. This white box method sounds to be very simple but actually it is very difficult because there is typically quite a lot source code and that's why there is also many so called false positives. It is very time consuming way. Also must be noticed that even if source code is available there can be includes of the libraries (lib or dll) which are closed. So anyway in most of cases in white box method all of the possibility source code is not readable format [SGA07].

In the grey box testing binary file or files are reversed back to more readable mode [SGA07]. This method is used example of malware testing and hackers who try to solve software keys (code). Typically there is some debugger software as IDA or ollydbg which can reverse binary code. At this point must be remembered that most software licencing does not allow reversing, so this might be against licence.

In black box testing in testers view tested area or system is kind of in the black box. Source code is not known and tester have to find out by him self how software actually work.

Fuzzing is test method where software's parameters or other arguments or values are tested by random or then special generated values. Fuzzing is very popular and effective method to do back box kind of testing. With fuzzing can find out software bugs and so called vulnerabilities. Anyway fuzzing methods are separated into many different areas depends on what is tested. Fuzzing is belonging in so called black box testing methods. In generally in fuzz testing tester input values to the program that might not be excepted. Values can be larger as usual or values can include some special character.

Basically there is at least one method to make fuzzing all most impossible or then very
difficult. In software can be check expected and allowed values by checksums. This fuzzing area is going between grey box and black box testing. In Taint Scope paper this check sum aware testing is tried to test. In Taint scope paper binary file is reversed to find out so called hot bits the check sum check and then this check is tried to denied.

Fuzzing

Fuzzing is basically method where input abnormal values into software and checking what happens. But that is not so easy as it sounds. First consider if one is doing fuzzing with random values and target system crash how one knows which random value actually did crashing. There must be some kind of validating and recording method for show what caused a crash. But also consider if we have 10 test cases, in every test case target system must be monitored at the same time that running tests. But if we mix test cases we can get 10! different test runs. Well consider what happens if there are 10 000 different test cases. Some times in certain software test cases impact some later test case which actually does crashing. So monitor as right as possible is very important for fuzzing. At all fuzzing is very effective but it doesn't give 100% prov about security.

Also planning fuzz test is very important. Always must be know what fuzzer actually is fuzzing. Example of fuzzing network protocols wrong value at the wrong place can cause that check sum control loose the packet before it reaches the target system.

There are also some limitations which fuzzing method wont find.

Access Control Flaws

If software has many different user levels as administrators and users and power users. It might be very difficult to show out what functionality of software is connected to which user layers. Example if fuzzing is done as regular user rights and some vulnerability gives administrators right this might not be seen at all [SGA07].

Poor design logic

It is nearly impossible show out by fuzzing if software uses insecure functions
or connections. Example if some software's authentication is poor. Then there is error in logic [SGA07].

Backdoors

Basically backdoors is seen no differently than other software [SGA07].

Memory corruption

In some cases example of format string exploits memory area is corrupted.
Process might continue or then process can be killed ans new process can be spawn. Depending of monitoring this might not seen by fuzzer [SGA07].

Multistage vulnerabilities

Vulnerabilities that are complex or contains many simpler attack [SGA07].

**Fuzzing methods and types**

Fuzzing has at least two categories Mutation-based fuzzing and Generation-based fuzzing. In mutation-based fuzzing test cases are mutations from data examples. Generation-based fuzzing test cases are build from some protocol or file format [SGA07].

**Methods**

**Pregenerated test cases**

In pregenerated test cases all test cases are generated before test. Test cases are build from specification so that possible values are changed to violate certain specification. PROTOS and its spin of company named CODENOMICON is doing this kind of fuzzing [SGA07]. Prgenerated test is very effective and fuzzers knows how testing is going on. But there is some disadvantages that random element is missing. This will limit possible test cases [SGA07].
Random
Random fuzzing might be very easy to build up. Because all data that is passed to software is generated by random element. Random testing is not quite effective and there is problem in crashing cases to validate what test case actually caused crashing. Also in this test protocol structures and specification must be studied that random test pass to the target [SGA07].

Manual protocol mutation testing
This testing is typically done manually by fuzzer. Tester try to generate some unexpected behaviours to the target by manually [SGA07].

Mutation or brute force testing
This kind of fuzzers are mixing random and pergenerated types. Cases are done on fly and those are changed case by case and byte by byte or bit by bit starting from right format [SGA07].

Automatic protocol generation testing
This is also mix of mutation or brute force testing. Tester need to study specification and then he need to generate grammar or definition to the fuzzing engine [SGA07]. Also this method of fuzzing is depending how aware fuzzer is about tested specification and its limits.

Types

Local fuzzers
Local fuzzing is done in computer and basically with this type of fuzzing is trying to get root or administrator access from normal user account. In linux or unix machines typical target is programs which are ran so normal user can start a program but running is done by root account [SGA07].

Command-Line Fuzzers
Command-Line fuzzing is testing softwares command-line arguments [SGA07].
Environment variable fuzzers
This type of fuzzing is testing or fuzzing environment variables that software might use [SGA07].

File format fuzzers
This type is fuzzing file types. Fuzzer generate malformated files and tries how software is dealing with those [SGA07].

Remote fuzzers
This type of fuzzing is done by remotely. Target is listening some TCP/IP port and fuzzing is done by remote [SGA07].

Network protocol fuzzers
This can be either local or remote fuzzing. This is tests how target handles network protocols that is being supported [SGA07].

Web application fuzzers
This fuzz tests target is web application or web site. By this fuzzer try to find if there is example of SQL injection or XSS type of vulnerabilities [SGA07].

Web browser fuzzers
This type of fuzzer tests web browsers. There is HTML, CSS, Javascript, COM, ActiveX etc which is fuzzed. Very simple method is to add tag <META REFRESH> in html code [SGA07].
At the latest security threats is focusing very much to the security of the web browsers.

In-memory fuzzers
This fuzzing type is not trivial. Main idea is that process is freezed and taken snapshot and malformat data is input to its routines. After this process can be run again [SGA07].
With this type of fuzzing checksum checks can be pass by. Because fuzzing can be done after checksum validating.
**TaintScope**

At first TaintScope checksum aware directed fuzzing tool was not tested because it is not public. In paper TaintScope writer call it checksum-aware fuzzing and directed fuzzing. Basically in directed fuzzing is meaning that taint analysis can be done at byte level. TaintScope can find security-sensitive points as some system calls. Also TaintScope is fully automatic [WWG10].

![TaintScope system overview](image)

Dynamic Taint Tracing is done in Execution monitor. Execution monitor records next two steps 1. which input bytes pollute the arguments of specified API functions; and 2. which input bytes each conditional jump instruction (JZ, JE, JB) depends on.1 is hot bytes information and 2 is checksum information [WWG10].

Checksum detector is detecting potentially checksum points [WWG10].

Fuzzer module generates malformed test cases and pass those back to program. All malformed test cases are based hot bytes information [WWG10].

**Fuzzing result**

In paper said that TaintScope has already detected 27 vulnerabilities in applications and libraries as Microsoft Paint, Adobe Acrobat, Google Picasa, Imagemagick and Libtiff. Paper also said that vulnerabilities is analysed by manually to belonging five different vulnerability types, including buffer overflow, integer overflow, double free, null pointer dereference, and infinite loop [WWG10].
Conclusion

In this paper I presented different fuzzing methods and types. Fuzzing is very effective security testing method. Area of fuzzing is separated in many different fuzzing methods. Different fuzzing methods can be distributed different fuzzing types. Fuzzing is at all very challenging testing area.

In software there is used checksums also to prevent fuzzing. In memory fuzzing there is couple tools which can pass by checksum control.

TaintScope is one tool to do checksum-aware directed fuzzing. By TaintScope program is runned inside fuzzer. TaintScope is also said to be effective and automatic fuzzer.

At all in fuzzing monitoring target system is very important. Most difficult section of fuzzing is find out which test case crashed a program. Fuzzing is not just running fuzzer as in some automatic process but also tester need to be aware how different specification works and what are limitations of specifications, protocols and fuzzing tools. There are many kind of potentially very good fuzzers but there are also many potential test cases and those differential variables. I think software can't be 100% tested not eve fussing. But fuzzing is very effective tool to raise security of softwares.
References

SGA07  Sutton Micheal, Greene Adam, Amini Pedram: Fuzzing Brute Force Vulnerability Discovery, 2007

WWG10  Wang Tielei, Wei Tao, Gu Guofei, Zou Wei: TaintScope: A Checksum-aware Directed Fuzzing tool for automatic software vulnerability detection, 2010