Discovering security defects in binary only software by fingerprinting

Introduction

Regularly security defects are found in electronics such as routers or NAS storage devices. With a shift from all purpose desktop computers/laptops to specialized devices (consumer electronics, SCADA, industrial automation) and a massive growth of electronics it is expected that these devices will get closer attention by people who want to exploit bugs in these devices.

As soon as these devices are deployed it is hard to fix bugs in them, either because it is hard to get users to install firmware updates (because the users don't want to, or because it is hard to get them to notice there is a firmware update), or it is impossible, impractical or dangerous to interrupt services and install an update (SCADA, industrial automation).

This document describes a method for proactively discovering if binary software is vulnerable by first detecting which files were used to build a binary and by then combining it with information about security bugs extracted from source code.

Tags

Linux, FreeBSD, *BSD, Unix, QNX, Solaris, Android, Java, software, software development, security, defect discovery, reverse engineering

Detailed description: using fingerprinting and combine it with security information

The method uses a combination of fingerprinting and security information obtained using static source code analysis. The first part of this method consists of analysing source code and finding out about security flaws in the source code. Common flaws are documented in for example the CERT secure coding standards. Source code can be analysed in a number of ways. One way is to use regular expressions to get the interesting bits of code, the other is to use a parser to build an abstract syntax tree (AST) and walk the AST to search for possible vulnerable code and further process results. After identification of possible vulnerable code the file name, file hash (MD5, SHA1, SHA256 or another hash), package, version and bugs found are stored in a database, together with information about identifiers (string constants, variable names, function names, method names, and so on) found in the file.

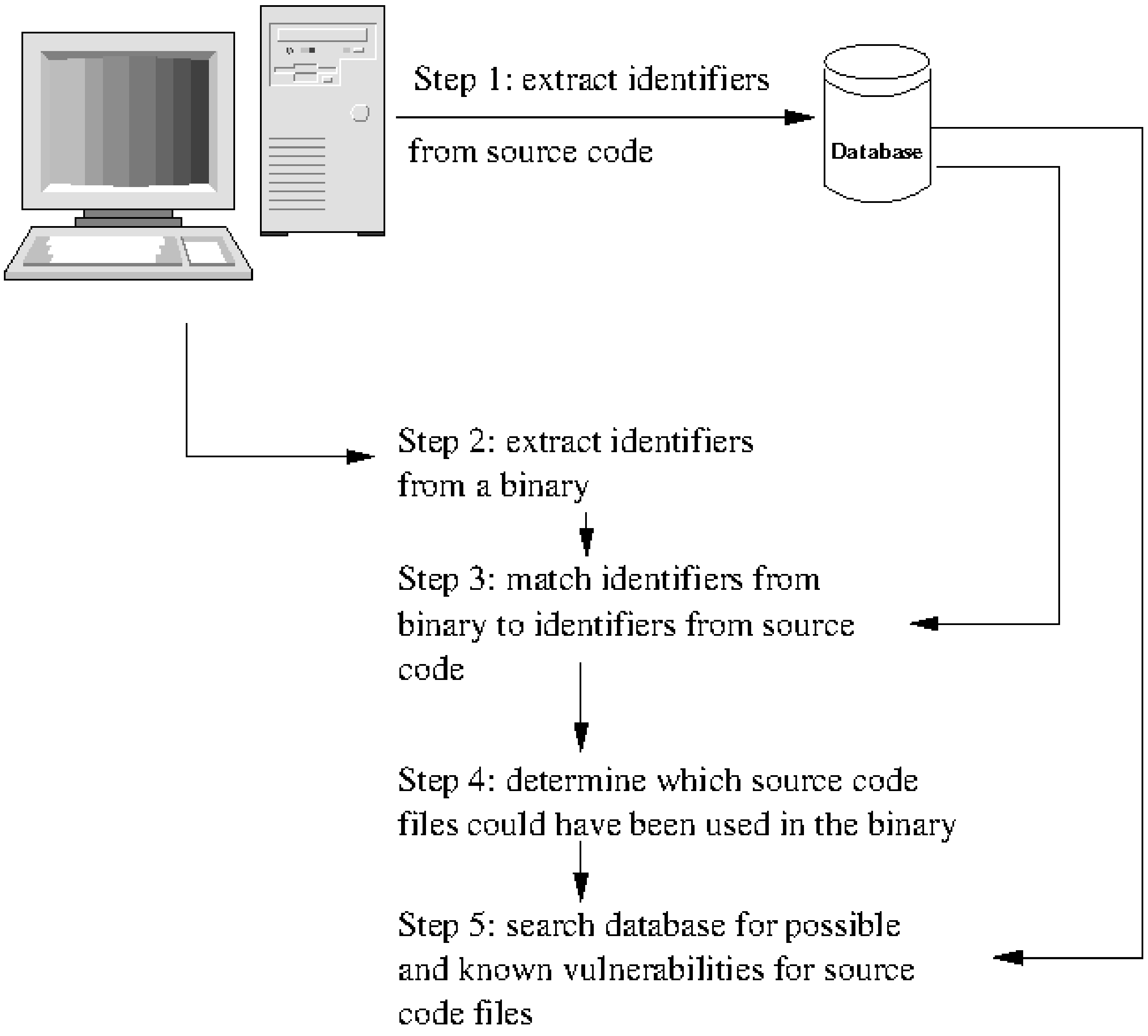
The identifiers (string constants, variable names, function names, method names and so on) are used to find out what software is inside the binary file and which specific source code files could have been used. This has been described at the Mining Software Repositories 2011 conference in the paper “Finding Software License Violations Through Binary Code Clone Detection” [2] and has been implemented in various tools like for example the Binary Analysis Tool and documented in other publications[3][4]. The method described in this document takes the result of this process and combines it with security information.

After detecting which files could have been used the database is queried to find out if there are any known and possible security bugs in those files. If so, these are reported.

Steps

1. create a database of information about software source code files, including identifiers (string constants, function names, variable names, and so on), file hash, package name, version name
2. analyse source code files to search for known and possible security defects and store these with the information from step 1, or in a separate table per file hash
3. analyse a binary file to find out which files could have been used
4. query the database to find if there are any known or possible security defects in the files that could have been used
5. report results from step 4.

Diagram


Step 1: extract identifiers
from source code
Database
Step 2: extract identifiers from a binary
Step 3: match identifiers from binary to identifiers from source code
Step 4: determine which source code files could have been used in the binary
Step 5: search database for possible and known vulnerabilities for source code files

References

[1] CERT secure coding standards - http://securecoding.cert.org/

[2] Finding software license violations through binary code clone detection. In Proceedings of the 8th Working Conference on Mining Software Repositories, MSR '11, pages 63–72, New York, NY, USA, 2011. ACM

[3] http://ip.com/IPCOM/000214472

[4] http://www.defensivepublications.org/publications/finding-software-license-violations-through-binary-code-clone-detection