INL/CON-17-41034
PREPRINT

Vulnerabilities Under the Surface

2017 Cybersecurity Symposium

Todd M. Keller, Jacob S. Benjamin,
Virginia L. Wright, Bryan H. Gold

April 2017

Vulnerabilities Under the Surface

Tracking potential vulnerabilities by voiding warranties

ABSTRACT

This paper will describe a practical methodology for understanding the cyber risk of a digital asset. This research attempts to gain a greater understanding of the cyber risk posed by a hardware-based computer asset by considering it as a sum of its hardware and software based sub-components.

KEYWORDS

2017 Security Symposium, Hardware Security, Software Security

ACM Reference format:
. 2017. Vulnerabilities Under the Surface. In *Proceedings of 2017 Cybersecurity Symposium, Coeur d'Alene, Idaho USA, April 17–19, 2017,* 3 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the proliferation of computers and embedded systems in all personal, commercial, and industrial settings, identifying the level of cyber risk associated with those assets is increasingly important. However, in most cases, documentation provided to the asset owner describing the software and hardware comprising an asset is incomplete at best. To properly understand and identify an asset's cyber risk, a thorough investigation of its hardware and software components is essential.

Many system vulnerabilities cannot be identified through superficial hardware inspection or a list of software installed in an asset because many vulnerabilities are contained at lower levels of hardware or software abstraction. It is necessary to first establish an asset's component makeup by performing an in-depth investigation of its hardware and software components to discover, understand, and address any potential vulnerabilities. A comprehensive security assessment investigation must examine the details of the hardware components and the software components installed in an asset. During this investigation we discovered vulnerabilities that would have remained undisclosed with currently used analysis methods. However, a proper security assessment examines all of the details associated with the systems. To achieve a comprehensive security assessment, a detailed asset inventory should be created by disassembling and cataloging hardware and software. This catalog can then be used to extrapolate the necessary information for a thorough vulnerability analysis of an asset and its components.

2 APPROACH

2.1 Hardware Approach

Comprehensively cataloging hardware components requires at least some disassembly and reverse engineering. However, for the sake of brevity, and since physical disassembly of a device is a relatively easy task, that aspect is not covered in this paper. When disassembling an asset, the focus should be to catalog any logic containing Integrated Circuits (IC) and potential storage components for further research or firmware analysis. However, IC identification is not an exact science and may require some level of reverse engineering to gather enough information to positively identify the IC in question.

One method that can be used for hardware component cataloging is logo matching. Identifying the manufacturer can make positive identification easier by greatly narrowing the possible matching hardware components. A second method, which can be used in conjunction with the previous method, involves performing searches using all or some of the strings of characters printed on the IC stamp. This can be combined with the IC package type and pin counts to further narrow results. Searches should be done both as general web searches on sites like Google and Bing, as well as component distributor sites such as DigiKey, Newark, and Mouser.

If these methods do not provide positive identification of an IC, more-intrusive methods for identification can be performed. This includes checking for the power and ground connection pins and attempting to match these or other potential identifiers to suspected data sheets of possible matches, which will narrow the options for positive identification. In some cases it may be necessary to check the voltage levels with a multimeter and the logic pins with an oscilloscope prior to using logic analyzers, if more simplified efforts fail.

Once the ICs have been identified, the compiled information should be placed into a form that is readily usable for searching, sorting, and filtering. Microsoft Excel was used in this research effort, but it is envisioned to store the results in a more user-friendly database to document and characterize any relationships that can be found.

2.2 Software Approach

Understanding the software installed in an asset is required to build a complete picture of its cyber risk or security posture. While not a completely new territory, we attempt to cover a proposed practical approach. For the purpose of this paper, software includes any drivers and/or client applications that interact with the asset's hardware. However, opposed to hardware component analysis, software component analysis is exponentially more difficult due to the complexity of software, its use of proprietary code, the time-consuming nature of code reverse engineering, and the lack of tools for analyzing binary-distributed software for risk analysis purposes.

While performing software component analysis, it is extremely important to focus on not only the software's functionality but any shared libraries or unanticipated software components contained within the programs. Software and drivers may utilize shared libraries and utilities that may be unexpected by administrators and security practitioners. In addition, shared code and utilities across multiple programs can negatively impact asset security. In a 2015 study, security researchers discovered that it took a median of 11 days and up to 118 days for two applications that shared common code to patch their respective programs.[2] For example, in the past, products that used OpenSSL had trouble with the HeartBleed vulnerability while others suffered from the ImageTragick vulnerability. These underlying components within the software are often overlooked by administrators when creating a digital asset inventory used to track vulnerabilities but are as important or more important due to the danger of shared vulnerabilities across software. The 2015 study also found a threat in multiple installation scenarios where even after patching, some shared code vulnerabilities may remain vulnerable due to incomplete patching mechanisms or inactive applications.[2]

The first step to understand what software is used by an asset is to capture and catalog all software components that are transferred during asset installation. Ideally, this can be performed with tools such as CaptureBAT or RegShot, which are typically used in forensic analysis of malware. Unfortunately, this method is costly and time-intensive and asset owners may be unwilling to spend the time or money to implement such an in-depth procedure. However, because many installation programs are comprised of relatively uncomplicated self-extracting compressed archive, a less-complex approach is to analyze the installation files. These files can then be uncompressed by conventional tools and compared with known files. A limited database of known files is available as part of National Institute of Standards and Technology's Software Reference Library (NSRL). The NSRL contains file names and hash signatures for some software and provides some limited utilities for interacting with software data. This is potentially the best-known freely available source of current data for trusted software information.

At a minimum, hashes of each file should be taken and included in the database. If available, the status and inclusion of exploit mitigations, such as Address Space Layout Randomization (ASLR), Data Execution Prevention (DEP), and whether Stack Canaries are enabled, can be included. Cataloging these details could provide a resource that could enable an asset owner to make purchasing decisions based on security metrics and provide an indication of the vendor's commitment to secure development.

2.3 Quick Wins and Potential Administrative Options

Administrative approaches may also provide fruitful information for component analysis. One approach is to use the vendor's license disclosure documentation of other licensed software packaged in an asset. In a discussion on finding shared libraries for network communications and packet capture on Mercedes S Class Vehicles, Peiter Zatko[4] noted that by reading the asset's license documentation, he was able to discover shared libraries of interest. On Twitter, he referenced a chart for Mercedes vehicles[1] detailing vehicle models correlated with which software licenses are included for use within the vehicle. This approach can be used to quickly gain an understanding of what might be found in an asset and any potential vulnerabilities an asset owner may want to track. This documentation can also be used to confirm and strengthen researchers' confidence in their findings stemming from the research steps discussed previously.

A second approach is to use a firmware flashing utility to expose device firmware, which can extract the firmware without requiring it to be extracted directly from the device. Once extracted, the firmware can be analyzed with standard system forensics techniques and binary code analysis.

Combining these two methods allows an asset owner to quickly obtain detailed information on hardware and software components, often at a more-detailed level than the data sheet, without incurring the expenses of reverse engineering.

2.4 Potential Gaps/Pitfalls

While relatively comprehensive, both the hardware and software approaches will likely contain gaps. Hardware will not always contain a data sheet and component identification may not be easily discerned from the vendor documentation. Software will change from updates, downloaded components, and internal update mechanisms that may not be presented to the user or administrator. Both hardware and software approaches will require either an assumption of trust or further reverse engineering to ensure an in-depth understanding of the asset's full functionality and capabilities.

2.5 Impact and Way Ahead

A vulnerability is not always obvious upon surface inspections. For example, the Genesys Logic GL3220 memory card reader controller is capable of In System Programming (ISP), and in some products it is believed to be potentially vulnerable to BadUSB. The GL3220 serves as an excellent and relevant example as it is used in many multimedia card readers. We identified products containing this chipset only by opening up the hardware for deeper inspection. A surface inspection would have not identified this chipset due to the lack of in-depth technical details in vendor-provided documentation. In addition, web searches for most card readers do not provide information on the type of chipset used, and an asset owner would be unable to determine if it was vulnerable to BadUSB. However, by dissasembiling the device and conducting searches on the GL3220 chipset itself, a single result[3] indicates it is likely vulnerable.

In some cases in our investigation, we were unable to determine if any increased cyber risk existed for some ICs. This occurred when we found an IC that lacked any public data sheet. Without a data sheet, an IC's function must be determined by other means, such as deductive reasoning or reverse engineering.

ACKNOWLEDGMENTS

This work was funded by the U.S. Department of Energy (Office of Nuclear Energy) under DOE Idaho Operations Office Contract DE-AC07-05ID14517

A APPENDIX - PROCESS SUMMARIZATION

A.1 Hardware Steps

A.1.1 Use device firmware flashing utilities to expose device firmware for binary analysis.

A.1.2 Disassemble hardware and catalog any/all logic containing integrated circuits.

A.1.3 Use logo matching to identify the manufacturer for any unidentified ICs.

A.1.4 Perform searches on strings printed on the IC stamp of unidentified ICs.

A.1.5 Combine IC package information with string searches for any unidentified ICs.

A.1.6 Check for power and ground connection pins and attempt to match to suspected data sheets or for additional search criteria if a positive match has not yet been made.

A.2 Software Steps

A.2.1 Review vendor's license disclosure for identification of other licensed software included.

A.2.2 Capture all software components that are transferred during asset installation.

A.2.3 Catalog software components including at least the file name and hash signature. Expand with security-relevant information.

A.2.4 Compare cataloged files with known files such as the NSRL. Expand with information from National Vulnerability Database and other security-relevant sources.

REFERENCES

[1] Daimler. 2013. Licence Agreement Supplement. http://moba.i.daimler.com/bai-cars/ba/foss/content/en/assets/FOSS_licences.pdf. (2013).

[2] Antonio Nappa, Richard Johnson, Leyla Bilge, Juan Caballero, and Tudor Dumitras. 2015. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. (2015). http://www.artificialstudios.org/jeppojeps/oakland15_camera_ready.pdf.

[3] SRLabs. 2015. SD Card Adapters Suspected vulnerable to Bad USB. https://opensource.srlabs.de/projects/badusb/wiki/SD_card_adapters. (2015). Research appears to be from 2015 based on that being the first date of capture on the Internet Archive Wayback Machine.

[4] Peiter Zatko twitter.com/dotmudge. 2016. Pre-hacked-car :). https://twitter.com/dotMudge/status/769588040884817920. (2016).