When Firmware Modifications Attack: A Case Study of Embedded Exploitation

Ang Cui, Michael Costello and Salvatore J. Stolfo
Department of Computer Science
Columbia University
New York, US
{ang, costello, sal}@cs.columbia.edu

***Abstract*—The ability to update firmware is a feature that is found in nearly all modern embedded systems. We demonstrate how this feature can be exploited to allow attackers to inject malicious firmware modifications into vulnerable embedded devices. We discuss techniques for exploiting such vulnerable functionality and the implementation of a proof of concept printer malware capable of network reconnaissance, data exfiltration and propagation to general purpose computers and other embedded device types. We present a case study of the HP-RFU (Remote Firmware Update) LaserJet printer firmware modification vulnerability, which allows arbitrary injection of malware into the printer's firmware via standard printed documents. We show vulnerable population data gathered by continuously tracking all publicly accessible printers discovered through an exhaustive scan of IPv4 space. To show that firmware update signing is not the panacea of embedded defense, we present an analysis of known vulnerabilities found in third-party libraries in 373 LaserJet firmware images. Prior research has shown that the design flaws and vulnerabilities presented in this paper are found in other modern embedded systems. Thus, the exploitation techniques presented in this paper can be generalized to compromise other embedded systems.**

***Keywords*-Embedded system exploitation; Firmware modification attack; Embedded system rootkit; HP-RFU vulnerability.**

I. Introduction

Modern embedded devices exist in large numbers within our global IT environments and critical communication infrastructures. Embedded systems like routers, switches and firewalls constitute the majority of our global network substrate. Special purpose appliances like printers, wireless access points and IP phones are now commonplace in the modern home and office. These appliances are typically built with general purpose, real-time operating systems using stock components. They are capable of interacting with general purpose computers as general purpose computers themselves.

The diverse and proprietary nature of embedded device hardware and firmware is thought to create a deterrent against effective wide-spread exploitation. While such claims of embedded security fundamentally reduce to security through obscurity, it is nonetheless claimed by embedded device vendors to provide security for their products [1].

To demonstrate that such claims of embedded security are overly optimistic and that emerging embedded exploitation techniques and embedded system malware pose a threat to the security of our existing networks, we present the following four contributions:

General firmware modification attack description: We present firmware modification attacks, a general strategy that is well-suited to the exploitation of embedded devices. This strategy aims to make arbitrary, persistent changes to victim devices' firmware by leveraging design flaws commonly found within embedded software. Firmware modification attacks can affect entire families of devices adhering to the same system design flaw, transcending operating system versions and instruction set architectures. The HP-RFU vulnerability presented in this paper affects MIPS- and ARM-based printers alike, regardless of their underlying software implementation. We discuss the general preconditions for and the process of leveraging firmware modification attacks against modern embedded devices.

HP LaserJet firmware modification case study: We use a firmware modification vulnerability recently discovered by the authors in nearly all HP LaserJet printers [2] to present a real-world case study of the development cycle of such attacks against common embedded devices. We present the threat model characterization, vulnerability analysis and threat assessment of HP-RFU and show a full exploit against the vulnerability. The entire process, from discovery to the implementation of the final attack and malware package, took approximately two months, was carried out using public vendor information readily available on the Internet and required a hardware budget of under $2,000. This attack is effective against the majority of LaserJet printers currently in production and affects a large number of installed devices. While it is difficult to divine the actual size of the vulnerable device population, HP shipped 11.9 million such units in a single quarter of 2010 alone [3].

The design flaws identified in the HP remote firmware update functionality can be seen in other modern embedded systems. Thus, the attack strategy we present can be generalized and applied to other vulnerable embedded device types. We discuss the offensive potential of our proof of concept printer malware and its impact on the efficacy of traditional network defense doctrine.

Vulnerable population / patch propagation analysis: The severity of the HP-RFU attack is further increased due to the

ubiquitous nature of the vulnerable population. While firmware fixes have been released by the vendor, mitigation of the vulnerability discussed in this paper ultimately depends on end-users diligently updating firmware. Applying firmware updates on mission-critical embedded systems can be cumbersome and daunting [4]. It is not surprising that we have found that this diligence is lacking, which favors the attacker.

We present the results of exhaustive scans of IPv4 to show the distribution of all publicly accessible, vulnerable LaserJet printers on the Internet. We have identified over 90,000 unique vulnerable printers inside numerous government organizations, educational institutions and other sensitive environments. We periodically fingerprint the specific firmware version of each tracked device in order to analyze the rate and pattern of firmware patching throughout the world. We believe this data will shed light on the inefficacy of the patch cycle for large populations of embedded devices as compared to patch propagation patterns within general purpose computer populations. Firmware patch propagation data for the first two months following the official release of firmware updates for 53 printer models [5] is presented in this paper. Initial data indicates a global patch level of approximately 1.08%. Furthermore, 24.8% of all patched printers still had open telnet interfaces with no root password configured (a default setting).

Vulnerable third-party library analysis: Mandatory firmware update signature verification was introduced by the vendor on some vulnerable LaserJet printer models following the disclosure of the HP-RFU vulnerability. This mitigates the specific vulnerability discovered by the authors. However, mandatory firmware signature verification allows known vulnerable code to be signed and verified. It does not remove the actual vulnerabilities within the signed firmware, nor will it detect or mitigate the exploitation of the actual vulnerability.

We present the results of automated analysis of a large collection of LaserJet printer firmwares released over the last decade, including the latest firmwares released in response to the HP-RFU disclosure. We analyzed all publicly available firmware images for 63 models of HP LaserJet printers. By cross-referencing the specific version numbers of third-party libraries like OpenSSL and zlib found within firmware updates with known vulnerabilities for those specific library versions, we conclude that a large number of vendor-issued firmwares are released with multiple known vulnerabilities. In some cases, we identified recently released firmware updates containing vulnerabilities in third-party libraries that have been known for over eight years. We identified third-party libraries with known vulnerabilities in 80.4% of all firmware images analyzed.

The remainder of this paper is organized as follows: Section II describes the general firmware modification attack strategy and surveys such existing attacks against embedded devices. Section III discusses the discovery of the HP-RFU vulnerability and the subsequent proof of concept attack and malware development. Section IV discuss the real-world offensive potential of our proof of concept attack. The distribution of publicly accessible vulnerable LaserJet printers and initial firmware patch propagation telemetry is presented in Section V. Vulnerable third-party library analysis of 373 vendor-issued firmware updates is presented in Section VI. We survey related works and ongoing work in the area of host-based embedded defense and vulnerability analysis in Section VII. Lastly, we propose recommendations for hardening embedded devices against attacks described in this paper in Section VIII and present our concluding remarks in Section IX.

II. Firmware Modification Attack

Firmware modification attacks aim to inject malware into the target embedded device. Predictions of firmware modification attacks against printers are almost a decade old [6]. Firmware modification attacks can be carried out either as standalone attacks or as secondary attacks following initial exploitation using traditional attack vectors.

Standalone firmware modification attacks manipulate firmware update features instead of exploiting flaws in the victim software. For example, the firmware modification case study presented in Section III utilizes the remote firmware update feature within HP LaserJet printers. This attack vector is not unique to the vulnerable devices discussed in this paper. Other ubiquitous embedded systems like ATM machines, smart battery controllers, keyboards, enterprise routers and PBX equipment are also vulnerable to such attacks. Similar standalone firmware modification attacks [7]–[12] have recently been reported.

The standalone firmware modification strategy is well-suited to embedded exploitation in general for the following reasons:

Feasibility: Firmware update is an ubiquitous feature found in modern embedded devices. Previous work [7], [13], [14] shows that a large number of embedded devices have firmware update features that are not sufficiently protected by proper user authentication. Many devices that require authentication before allowing firmware updates are vulnerable to trivial administrative interface bypass attacks [15]. Furthermore, net-booted embedded devices that use insecure protocols like TFTP to retrieve their configurations and firmware are vulnerable to standard OSI Layer 2 attacks.

Fail-Safe: Firmware update mechanisms usually mandate integrity and model verification prior to execution of the actual firmware modification. Malicious firmware update packages sent to incompatible embedded devices are rejected and ignored. This relaxes the reconnaissance and accuracy requirements for the attacker and reduces the penalty of a misdirected attack. For example, the final malicious binary described in Section III contains a single RFU image targeting a precise printer model. However, if the exact model of the victim printer is unknown, multiple malicious RFU commands covering all potential printer models can be sent sequentially without damaging the printer. Furthermore, each RFU command need not contain a full printer OS image, which is at least several megabytes in size. A bare-bones OS boot loader can be sent instead. Such a loader will be at most be several

hundred kilobytes in size (the development of this offensive technique is outside the scope of this paper).

Platform Independence: Attacks that manipulate firmware update features within the vulnerable device do not need to depend on specific software vulnerabilities in the victim and will generally work across many models of the same device, even across different machine architectures. For example, the HP-RFU vulnerability manipulates a feature of the LaserJet firmware, which is supported across nearly all printer models and is common among MIPS- and ARM-based devices.

While mandatory firmware signature verification can mitigate standalone firmware modification attacks, this countermeasure is not the panacea of embedded security. Firmware modification attacks can be carried out as a secondary payload following the successful exploitation of the embedded device via traditional vectors like memory modification attacks. Firmware content is typically stored in rewritable, nonvolatile memory like flash. Embedded operating systems generally lack the fine-grain privilege separation and execution isolation found in modern operating systems; even when available in later builds, vendors oftentimes choose to not utilize these memory isolation features. Furthermore, for embedded operating systems with process and memory isolation, vulnerabilities within the kernel or privileged processes can still allow an attacker to make persistent changes to the device. For example, prior research has demonstrated that it is possible to make persistent modifications to the boot ROM portion of enterprise routers using only software operations [16]. Thus, countermeasures like authentication and firmware signature verification cannot fully prevent firmware modification attacks on embedded systems with vulnerable attack surfaces.

Section III illustrates the development cycle of a typical firmware modification attack and embedded malware. Section VI presents vulnerable third-party library analysis for a large corpus of HP LaserJet firmware images.

III. Case Study: HP LaserJet Exploitation

The HP-RFU firmware modification vulnerability [2] was discovered unintentionally when the authors attempted to inject host-based defenses into network printers. The HP LaserJet family was chosen because of its popularity and commanding market share [3]. The LaserJet P2055DN model was chosen as our initial target device.

Analysis of the HP LaserJet firmware revealed a reliably exploitable design flaw that allows remote attackers to make persistent modifications to the printer's firmware by printing to it.

In order to inject host-based defenses into any target hardware, the original firmware must be unpacked and analyzed. In the case of prior work on Cisco IOS routers, this process was straightforward[1]. However, unpacking and analyzing HP LaserJet firmware images presented several challenges. Figure 8 of the Appendix shows the hex dump of a RFU file.

[1]IOS images are simple ZIP files with slightly non-standard headers.

The remote firmware update for the P2055DN printer begins with standard PJL (Printer Job Language) but enters into an undocumented language called *ACL*. Approximately 7 MB of binary data follows. Initial static analysis[2] revealed no recognizable filesystem headers and no function preambles for any known machine architecture inside the RFU binary.

Without further analysis, a key design flaw became apparent: the firmware modification mechanism is coupled with the printing subsystem, which must accept incoming requests in an unauthenticated manner as per general specification. As confirmed by vendor documentation [17], the RFU file is *printed* to the target device via the raw-print protocol over standard channels like TCP/9100, LPD and USB. Various other vendors also use the same update strategy.

When a print job is received by the printer's job-parsing subsystem, a proprietary mechanism is used to determine the presence of a valid firmware update package. If a PJL command containing a valid RFU package is present, the integrity of the RFU payload is verified and decompressed. The payload's unpacked binary data is then written to persistent storage within the target printer, thereby modifying the printer's firmware.

Once the RFU binary structure was obtained through standard hardware and software reverse engineering methods, we discovered that it was possible to pack arbitrary executable code back into a legitimate RFU package in a PJL command. This command can then be embedded into a malicious document or sent directly to the victim printer to arbitrarily and persistently modify its firmware. Such an attack does not affect the printing of the legitimate carrier document and only makes the printer unavailable for approximately 90 seconds. The printer will continue to respond to network requests throughout most of the firmware update process. Thus, the attack will likely go completely unnoticed by users and network monitoring systems.

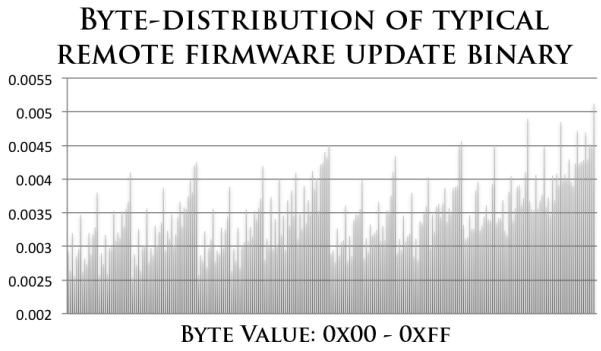

Fig. 1. Byte value distribution histogram of a typical RFU file. Distribution suggests that the data is compressed and not encrypted.

[2]We used standard industry practices of loading the image into IDA Pro, fixing the memory mapping, and so forth. A detailed discussion of reverse engineering is outside the scope of this paper.

A. Discovery Process

Initial static analysis of the original RFU binary revealed no printable strings, no known filesystem headers nor recognizable executable binaries. We concluded that the binary payload was likely either encrypted or compressed. Figure 1 shows the byte distribution histogram for a typical RFU binary payload for the P2055DN printer. The histogram suggests that the binary blob is compressed and not encrypted as common encryption algorithms typically generate high-entropy ciphertext, which was not observed.

Manual inspection of the binary revealed a simple package header structure containing a short checksum field followed by multiple entries of the same data structure, containing the compressed and uncompressed size of each firmware component as well as its target address within the printer's persistent storage address space. This header is shown in Figure 9 of the Appendix.

Fig. 2. Formatter board for LaserJet P2055DN. Dump of the onboard SPI flash revealed RFU format and integrity checking algorithm.

The printer's formatter board hardware components were desoldered and reverse engineered. Figure 2 shows the actual formatter board inside the target device. Figure 3 illustrates the main components found on the P2055DN's primary control (formatter) board. Manual inspection revealed that the system was powered by a Marvell SoC. Aside from the machine architecture (ARM), no other information was publicly available due to the proprietary nature of the chip. However, the SPI flash chip is a stock component with a publicly available datasheet.

The main SoC on the formatter board uses the Spansion flash chip as a boot device. This chip has 8 MB of storage and communicates with the main Marvell SoC via a Serial Peripheral Interface (SPI) using a simple command protocol defined in its datasheet. In order to extract the contents of the flash chip, a SPI chip dumper was implemented using an Arduino [18] to perform the actual I/O. Figure 4 shows the physical hardware setup connecting the SPI boot flash chip to the Arduino board.

Analysis of the boot loader code revealed the binary structure and compression algorithm used in the RFU format. Manual inspection of the flash chip content revealed a boot image layout shown in Figure 5.

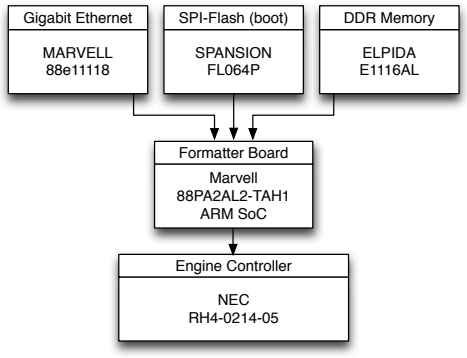

Fig. 3. Logical block diagram of the major components used on the LaserJet P2055DN formatter board. The Spansion boot flash was key to our reverse engineering effort.

A factory reset RFU image was found inside the boot flash. This image is immediately preceded by a boot loader containing the code that validates and parses RFU images. IDA Pro [19] disassembled the boot loader binary. The resulting assembly code revealed that the RFU image is validated using a trivial checksum function and compressed using a common algorithm.

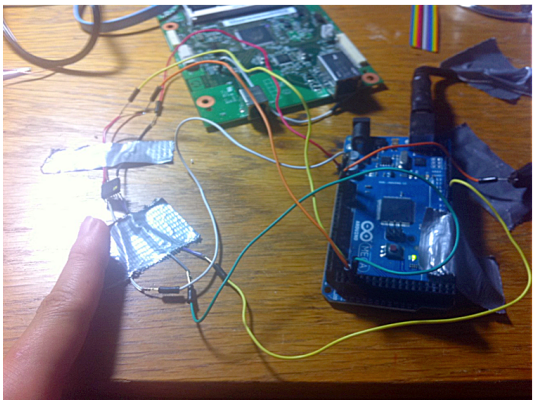

Fig. 4. The SPI flash chip was physically removed then connected to an Arduino for boot code extraction.

Furthermore, the specific version of the compression library used to process RFU images appear to have several known arbitrary code execution vulnerabilities [20]–[22]. Section VI presents an analysis of vulnerable third-party libraries found in a large number of firmware images released by the vendor.

B. Proof of Concept Printer Malware

Static analysis of the extracted boot flash code revealed the precise RFU binary structure, checksum and compression algorithms used. This information allowed the authors to write HPacker, a tool that takes an uncompressed ARM ELF image as input and returns a valid compressed PJL update command as output.

The malicious PJL command can be printed directly to the target printer or embedded within various document formats (an example is included in Figure 10 of the Appendix). Either

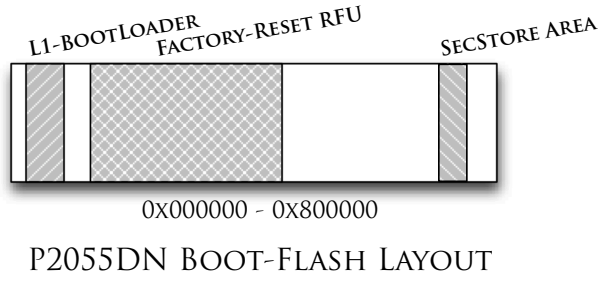

Fig. 5. Boot image layout on the SPI flash chip. The level-1 boot loader contains code that validates, unpacks and decompresses the factory reset RFU allowing us to reverse engineer the binary RFU format and compression algorithm.

way, once the PJL command is sent to the victim printer, it will recognize the print job as containing a valid firmware update and allow the attacker to make arbitrary modifications to the victim's firmware storage area.

The unpacked RFU package for the P2055DN contains over a dozen files. The main file of interest is the binary OS image, a single 14 MB ELF image containing the VxWorks operating system and various other vendor-specific additions.

The creation of the proof of concept malware essentially reduced to creating a VxWorks rootkit capable of:

- Command and control via covert channel
- Print job snooping and exfiltration
- Autonomous and remote-controlled reconnaissance
- Multiple device type infection and propagation to the Windows operating system and other embedded devices
- Reverse IP tunnel to penetrate perimeter firewalls
- Self-destruction

A video discussing the technical mechanics of this rootkit and a demonstration of its capabilities is publicly available [23].

The VxWorks OS image found within the RFU binary contains a complete socket library [24] and direct access to the underlying network transceiver hardware. The creation of the proof of concept code was mainly an exercise in identifying and intercepting the proper pieces of binary within the VxWorks image.

No host-based security mechanism exists within the firmware image. Thus, the attacker is free to make arbitrary changes to the victim device. As long as the functionality and general performance of the device is not altered, detection of firmware modification is not possible without careful removal and inspection of the hardware inside the printer.

Several challenges arose during the construction of the proof of concept code. The VxWorks image extracted from the RFU package contained no symbol information. Locating the appropriate socketlib, print job processing and raw network I/O binary interfaces within the binary proved non-trivial.

We developed a set of IDA-Python scripts to perform standard control-flow analysis of the target binary around code that we manually identified as network-facing. This effort was expedited by a patch made to the VxWorks kernel, which redirected debug messages destined for the UART to a TCP connection. Using these two mechanisms, a dynamic analysis environment was created to probe network-facing code, which eventually yielded a small set of functions likely to be libraries used by multiple pieces of unrelated code. Function prototype data was taken from available VxWorks documentation and used as a final check to positively identify each library function.

Typically, the malware would be optimized, compressed, packed and broken up to fit within gaps inside the original firmware or placed within dynamically allocated memory. However, since the attacker controls the firmware storage area absolutely, we added a new section within the ELF header marked with *rwx* privileges. This gave us more than sufficient space to implement all the previously mentioned malware functionality. In total, 2,800 lines of assembly were written to create the proof of concept malware.

IV. Threat Model and Assessment

We present the threat model and assessment analysis for the HP-RFU vulnerability presented in Section III.

A. Threat Model Characterization

The HP-RFU vulnerability exploits a design flaw in the firmware update mechanism found in nearly all LaserJet printers. In order to achieve persistent firmware modification on the victim device, the attacker must deliver a malicious PJL command to the raw-printing processing subsystem of the target. This can be done by using the following attack types:

Active Attacks require the attacker to directly trigger the firmware update process by actively connecting to the printer and sending it the malicious PJL command over the printer's raw-printing port.

Reflexive Attacks are akin to reflexive cross-site scripting attacks where malicious firmware update commands are embedded in passive data that is passed along to the user of the victim device. For example, the final binary package of the HP-RFU attack can be embedded inside innocuous-looking documents and sent to unwitting users, perhaps in the form of an academic paper or resume. In this reflexive attack scenario, the actual attack is launched when the malicious document is *printed*.

B. Threat Assessment

Figure 6 illustrates an advanced persistent attack scenario where a compromised printer is used as a reconnaissance tool and offensive asset. Once the malware package is delivered to the victim printer, it can be used to carry out firmware modification attacks against other embedded devices like other printers, IP phones and video conferencing units. Compromised embedded devices can be used to establish reverse IP tunnels back out to the Internet, giving the attacker direct access to the secured internal network. These devices can also be used to carry out standard network attacks like ARP cache poisoning and act as offensive assets to further compromise

general purpose computers and other embedded devices behind the victim's perimeter defenses.

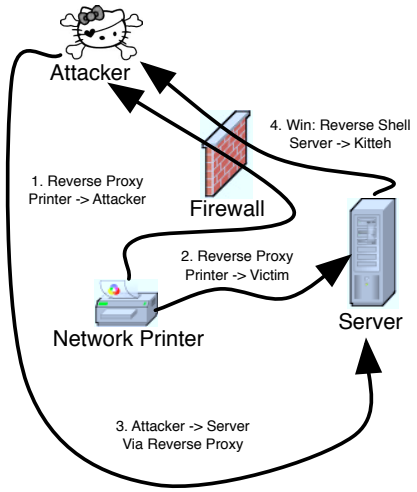

Fig. 6. Typical advanced persistent threat attack scenario involving compromised printers.

No host-based security mechanisms exist on the compromised printer. Thus, the presence of malware on this device will most likely go undetected if the functionality of the device is not affected. The compromised printer is an ideally situated stealthy asset that can be used as a fail-safe device allowing the attacker re-entrance into the victim network even if all compromised general purpose machines are neutralized. Contrary to the sensationalized media coverage regarding the HP-RFU vulnerability, it would be unwise for the attacker to destroy a compromised printer physically[3].

The HP-RFU vulnerability disclosure is described in CVE-2011-4161 [25]. As Section V shows, there are currently over 90,000 vulnerable LaserJet printers publicly accessible over the IPv4 Internet.

C. Compounding Factors

The following factors compound the severity of the HP-RFU vulnerability:

No authentication prior to firmware update: The PJL/RFU mechanism is coupled with the raw-printing protocol, a clear-text protocol that does not support authentication. Any party who is allowed to use the victim printer can carry out a firmware modification attack against the printer. Therefore, the attacker does not need to have direct IP connectivity to the victim printer even in the active attack scenario because the malicious payload can be relayed by intermediate print servers.

RFU feature enabled by default: The majority of the firmwares we analyzed enable the remote RFU update feature by default. Network-printing typically requires the printer to be reachable via TCP/9100. Since arbitrary binary traffic is allowed in the raw-printing protocol by specification, it is difficult to detect and stop malicious PJL commands at the network layer. Recent research suggests that it may be possible to use languages like PostScript to compute a valid, malicious PJL command on the victim printer when the malicious document is processed [11]. If so, this will significantly increase the difficulty of detection of this type of attack on the network level or within print servers.

Poor and incomplete configuration interface: The configuration interface of many "advanced" security features does not exist on the printer's HTTP or Telnet administrative interfaces. For example, disabling the remote RFU feature and setting PJL passwords can only be done through a separate enterprise printing management tool called HP Web Jetadmin (WJA) [26]. This is a 315 MB program that requires the installation of Windows-based web and SQL servers and is generally not practical for average users without enterprise IT support.

RFU feature cannot be disabled: Several LaserJet models, including the P2055DN used in our initial experimentation, do not support any way to disable the remote RFU feature, even through Web Jetadmin. As far as the authors are aware, prior to the release of the second version of the security bulletin [27], no combination of available configurable settings could disable the vulnerable feature on these printers. Furthermore, these models were not included in the first release of the security bulletin [28], since security bulletins released by the vendor must contain an acceptable mitigation method. Since no firmware fix was available, the devices most affected by the HP-RFU vulnerability were not listed in the initial vendor disclosure document.

Potential for irreversible, permanent malware injection: The SPI boot flash chip used on the P2055DN formatter board supports a One-Time-Programmable (OTP) feature [29] that allows areas of memory within the chip to be programmed and locked *permanently*. This is an irreversible operation that is typical for similar flash components. If the malicious malware package injected into the boot flash chip of the printer took advantage of this feature, removal of the malware would be impossible without physical removal of the compromised component.

V. Vulnerable Device Population Analysis

Vendors of general purpose operating systems and popular applications have deployed large-scale distribution networks to automatically update host software with little to no user interaction. However, no such widely deployed distribution exists to push patches and firmware updates to embedded systems.

The results presented in this section indicate that approximately 1.08% of vulnerable HP LaserJet printers have been patched worldwide, despite the public announcement of the HP-RFU vulnerability and the rapid release of firmware updates by the vendor (see Table I).

[3]While it has been demonstrated and stated in the initial reports that using the printer's fuser as an ignition source to create fire is not possible, physical destruction of the printer is possible via multiple methods.

This highlights the ineffectiveness of simple public release of firmware updates for vulnerable embedded devices. Empirical evidence suggests that vulnerable embedded devices will persist for a long period of time as compared to vulnerable general purpose computers. The threat will persist unless proactive firmware update mechanisms are developed for legacy embedded systems. However, a more proactive firmware update mechanism may also be exploited in firmware modification attacks.

A. Methodology

In order to quantify the number of printers that are vulnerable to the HP-RFU attack, we scanned the IPv4 Internet for publicly accessible HP printer web, telnet, SNMP and raw-print server sockets. Model numbers and firmware datecodes were gathered by employing the following methods:

- "@PJL INFO ID" command over TCP/9100
- "@PJL INFO CONFIG" command over TCP/9100
- "@PJL INFO PRODINFO" command over TCP/9100
- SNMP GET using "public" as the community string
- Model-specific banner scraping over TCP/23,80

B. Findings

In the two months following the official release of firmware updates for the HP-RFU vulnerability, we identified 90,847 unique HP printers that are publicly accessible over the IPv4 Internet. Firmware version data is collected periodically for each device. Table I shows our findings.

Potentially vulnerable printers	90,847
Printers with identifiable firmware datecode	74,770
Number of patched printers	808
Overall patch rate	1.08%

TABLE I
OBSERVED POPULATION OF PRINTERS VULNERABLE TO THE HP-RFU ATTACK ON IPV4.

Patching vulnerable printers to the latest firmware does not necessarily secure the printer. We probed each printer for other well-known vulnerabilities and common misconfigurations that can result in unrestricted root-level access to the printer. Table II lists the vulnerabilities, including a ChaiVM vulnerability FX exploited in 2003 [30] (this talk also discussed the potential for firmware modification).

Vulnerable printers are grouped into five general organizational types: *educational*, *private enterprise*, *military*, *civilian government* and *Internet service providers*. Tables III and IV show the distributions of the average age of the firmware images currently installed across different organization types and continents, respectively. The firmware age is taken from the datecode in the response from the devices' administrative interfaces. Organizational and geographic data were gathered though the DNS, Internet Routing Registry (IRR) whois or commercial geolocation databases.

Unrestricted Telnet	50,500
Unrestricted ChaiVM[4]	27,570
Vulnerable Virata EmWeb[5]	2,740

TABLE II
OBSERVED POPULATION OF PRINTERS VULNERABLE TO ATTACKS OTHER THAN HP-RFU ON IPV4.

	Count	Avg Age (years)	Oldest Firmware
Education	48,626	4.13	1993-08-20
ISP	4,650	3.70	1994-10-12
Enterprise	2,842	4.02	1992-12-16
Military	201	4.63	1999-10-30
Government	126	4.33	1996-12-20

TABLE III
ORGANIZATIONAL DISTRIBUTION OF VULNERABLE PRINTERS.

The above data is a lower bound on the number of vulnerable LaserJet printers on the Internet since it does not include devices behind firewalls or NATs or in other private networks..

In the months following the HP-RFU vulnerability disclosure, we observed 808 unique vulnerable printers that have been updated to firmware versions that mitigate the problem. We also observed 211 printers that did not require updated firmware to be invulnerable to the HP-RFU. However, out of these 1,019 devices, 24.8% (253) of them still have open telnet interfaces with no root passwords configured.

Approximately 64% of all vulnerable printers were located in North America. Over 65% of all vulnerable printers were found within the networks of educational institutions worldwide.

We also identified the following populations of vulnerable printers within two notable organizations:

- United States Department of Defense: *201 printers*
- Hewlett-Packard: *6 printers*

VI. VULNERABLE THIRD-PARTY LIBRARIES

Mandatory firmware signature verification was introduced by the vendor [5] in response to the disclosure of the HP-RFU vulnerability. While this effectively mitigates the specific attack presented in Section III, we believe this response is inadequate for at least two reasons:

Signed firmware $\neq$ secure firmware: Firmware signature verification guarantees that the binary data to be processed at firmware update time originated from a trusted source within the vendor's organization. Vulnerable code that is signed by the vendor remains vulnerable to exploitation. This mechanism does not prevent firmware or memory modification attacks in general and thus contributes little to the overall security of the embedded device.

Signed firmware prevents independent third-party defense development: Mandatory signature verification that only accepts firmware updates signed by the vendor will categorically

[4]The ChaiVM EZLoader allows unsigned .jar files to be installed [31].
[5]A remote crash vulnerability exists in Virata EmWeb R6.0.1 [32].

	Count	Avg Age (years)	Oldest Firmware
N. America	47,840	4.46	1992-12-16
Europe	14,196	4.16	1993-08-20
Asia	10,353	3.77	1998-09-02
Oceania	1,081	4.79	1998-09-02
S. America	673	3.43	1999-01-27
Africa	60	4.56	2001-04-26

TABLE IV
GEOGRAPHICAL DISTRIBUTION OF VULNERABLE PRINTERS.

prevent all non-vendor issued code from running. This makes the injection of legitimate third-party host-based defenses into vulnerable firmware images impossible.

In order to show that firmware signing as the sole security mechanism is inadequate, we present the results of the automated analysis of the third-party library vulnerabilities in a set of 373 firmware update packages issued by the vendor over the last decade. The dataset includes 358 RFUs released prior to the disclosure of HP-RFU as well as 15 RFUs released as part of SSRT100692 rev.3. The printer models and firmware images analyzed are listed in Table VII of the Appendix.

A. Methodology

All RFU images were unpacked and decompressed. Embedded filesystems (LynxFS) were extracted from the decompressed data. Extracted executables and shared objects were pattern-matched against known ASCII and binary signatures to detect the presence of specific versions of two specific third-party libraries: zlib and OpenSSL.

While this process suggests the presence of specific versions of third-party libraries in the analyzed firmware updates, no analysis was performed to check whether the libraries can be invoked by the attacker, or that the known vulnerabilities are reliably exploitable on the printers' machine architectures. This is the topic of ongoing research.

We present findings for the following third-party library vulnerabilities found in 373 vendor-issued firmware updates:

zlib: *CA-2002-07, CERT-{68062, 238678}* Discovered in 2002, zlib ver. 1.1.3 and earlier have a double free bug that allows arbitrary code execution [20]. In 2005 the vendor was notified of a buffer overflow in zlib ver. 1.2.1 and 1.2.2 [21]. The vendor was notified of a DOS condition in zlib ver. 1.2.0.x and 1.2.x in 2004 [22].

OpenSSL: *CVE-{2009-3245, 2006-3738, 2006-4339}* There are over 100 known vulnerabilities in various versions of OpenSSL. We scanned for the above three critical vulnerabilities in our firmware update dataset because they involve features that are likely to be reachable via network attack. The first two vulnerabilities can lead to arbitrary code execution. The last vulnerability can bypass x.509 certificate verification.

B. Findings

Figure 7 shows the percentage of vendor released firmware images that uses versions of zlib and OpenSSL library containing known vulnerabilities for a subset of LaserJet models.

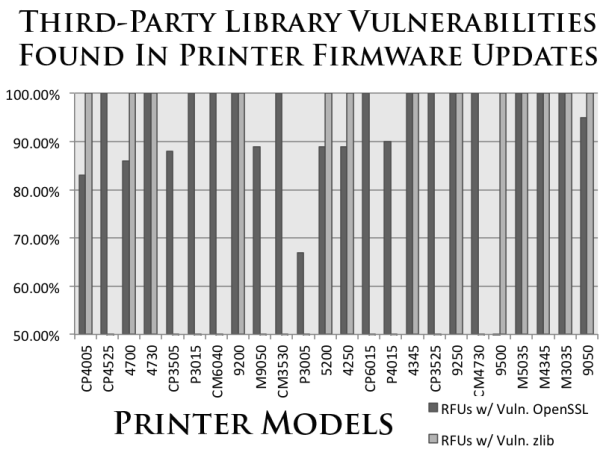

Fig. 7. Percentages of RFUs for each printer model containing known zlib and OpenSSL vulnerabilities.

Model	Lib	Earliest RFU	Latest RFU
2055	ssl	Unknown	Unknown
	zlib	2009-04-30	Present
4005	ssl	2010-02-11	Present
	zlib	2009-06-05	Present
4250	ssl	2004-09-02	Present
	zlib	2004-09-02	Present
4700	ssl	2009-09-14	Present
	zlib	2009-06-05	Present
9050	ssl	2004-06-30	Present
	zlib	2004-06-30	Present

TABLE V
LIFESPAN OF VULNERABILITIES IN THIRD-PARTY LIBRARIES USED BY LASERJET FIRMWARE.

Table V shows the duration of which known vulnerabilities have existed for in various models of LaserJet printers. Overall, we made the following observations:

Printer models analyzed	63
RFU images analyzed	373
All RFUs w/ at least 1 vulnerability	300
Latest RFUs w/ at least 1 vulnerability	41 (65.1%)
Most common zlib version	1.1.4
Most common OpenSSL version	0.9.7b

TABLE VI
THIRD-PARTY LIBRARY VULNERABILITY ANALYSIS OBSERVATIONS.

Mandatory firmware update signature verification is not an adequate defense mechanism against vulnerabilities that exist in the codebase of existing printers. Therefore, a large population of network printers is still potentially vulnerable to exploitation, despite the firmware updates released by the vendor.

VII. RELATED WORK

This section surveys recent firmware modification attacks as well as host-based defense technologies that can be applied to mitigate firmware modification attacks against embedded systems and concludes with a brief discussion of firmware analysis and its tools.

A. Recent Firmware Modification Attacks

Firmware modification attacks against the telecommunication infrastructure [10], [16], SCADA and PLC systems [33], laptop battery controllers, network interface cards, automated teller machines, medical devices and a wide range of other critical embedded systems have been demonstrated. For example, in the so-called *Athens Affair*, Ericsson AXE mobile phone base station controllers were altered to have their lawful intercept code surreptitiously activated in the Vodafone network in Greece [34].

PsycoB0t, the first publicly known router botnet, modified the firmware of approximately 85,000 DD-WRT home routers to include an IRC-based bot controller that was used briefly to carry out denial of service attacks before mysteriously disappearing [35]. Barnaby Jack controlled ATMs and drained them of their cash by replacing firmware in specific models [9]. Miller demonstrated Mac laptop battery controller firmware modification [8]. Costin demonstrated several PostScript-based attacks against Lexmark printers capable of memory inspection and possibly arbitrary modification [11]. Finally, Fu's body of work on medical device security, including realized attacks against an implantable cardioverter debrillator [36] and an automated external defibrillator [7], has shown the consequences of the exploitation of embedded devices.

B. Embedded System Defense Technologies

Numerous rootkit and malware detection and mitigation mechanisms have been proposed for general purpose computers and operating systems (virtualization-based [37], binary analysis [38], function hook monitoring [39], etc). These strategies may perform well within general purpose computers and well-known operating systems, but they have not been adapted to operate within the unique characteristics and constraints of embedded device firmware (limited storage, memory and processing; absence of memory management units; real-time operating systems; etc). Effective prevention of binary exploitation of embedded devices requires a rethinking of detection strategies and deployment vehicles.

Rinard posits that security vulnerabilities are excess, unwanted features in a software system [40]. This comes about through overly general (bloated) software, feature accretion, subsystem reuse and development errors on the part of designers and implementors and vulnerability insertion on the part of attackers. Several remedies are outlined including feature replacement or excision, input rectification and dynamic modification, and techniques for allocating memory and handling loops and typical failure conditions are discussed.

DynamoRIO [41] originated from a collaboration between HP, who created Dynamo, and MIT, who created RIO. DynamoRIO is a runtime code manipulation system that supports code transformations on any part of program. An application launched by DynamoRIO can be analyzed and manipulated through its API. DynamoRIO is designed for general purpose operating systems like Windows and Linux on the x86 architecture.

Much work has been done in using remote software attestation as a defense against firmware modification. SWATT: Software-Based Attestation for Embedded Devices, proposed by Seshadri *et al.* [42], and SBAP: Software-Based Attestation for Peripherals, proposed by Li *et al.* [43], involve the external validation of embedded devices through the use of a challenge-response protocol. In fact, VIPER, proposed by Li *et al.* [44], can be directly applied to mitigate a real-world firmware modification attack against keyboards [12]. While promising, such defense mechanisms are generally stop-the-world algorithms, requiring a full halt of the system while remote attestation is in progress. While perhaps adequate for printers, it would be difficult to directly apply such techniques to embedded devices like routers and firewalls, which must deliver uninterrupted availability.

Guards, originally proposed by Chang and Atallah [45], are simple pieces of code that are injected into the protected software using binary rewriting techniques. Once injected, a guard can perform tamper-resistance functionality like self-checksumming and software repair.

C. Further Firmware Analysis and Useful Tools

The vulnerable third-party library analysis presented in Section VI is likely symptomatic of a larger phenomenon. We believe that rigorous analysis of the code and data of proprietary embedded systems will yield important insights into the exploitability of such devices. The HP-RFU case study presented in this paper revealed several obstacles that impeded vulnerability analysis on legacy embedded systems. While the process of reverse engineering proprietary firmware image formats is a necessary prerequisite to useful analysis, it is a time-consuming and energy-intensive exercise that must be repeated for each new embedded device type. The following open source tools greatly streamlined our firmware format reverse engineering process.

Binwalk [46] is a pattern-matching tool designed to search for known headers and structures in arbitrary binary data. It is particularly useful for identifying known executable headers, detecting the ISA of potentially executable data by identifying known function prolog and epilog patterns and recognizing compressed data by locating headers of well-known algorithms.

FRAK [47], the Firmware Reverse Analysis Konsole, is a recently released open source framework designed to modularize and automate the firmware unpacking, analysis, modification and repacking processes. It has been particularly useful in automating large-scale analysis of firmware collections and identifying structures within firmware images of unknown formats.

VIII. Recommended Defenses

We discuss two host-based defense techniques developed by the authors to mitigate the vulnerabilities described in this paper. The vulnerable firmware update feature found in HP LaserJet printers is rarely used and should be disabled until it is needed. However, we found that disabling this feature

was not trivial and at times impossible, as was the case with the LaserJet P2055DN. We propose a technique, which we call Autotomic[7] Binary Structure Randomization (ABSR), which not only disables unnecessary features, but also removes the unused binary from the firmware image. This technique simultaneously reduces the attack surface of the embedded device as well as the amount of code and data that can be used as part of any shellcode.

Disabling unused features on the embedded device is helpful, but does not guard against exploitation via attack vectors within necessary features that cannot be removed. For example, vulnerable third-party libraries like ones identified in Section VI may be pivotal to the functionality of the embedded device. We believe techniques like ABSR should be used in conjunction with other host-based defenses to detect and mitigate the consequences of successful exploitation. Software Symbiotes have been demonstrated as a viable dynamic firmware integrity attestation technique on embedded systems such as enterprise routers.

Despite proper software and security engineering practices by vendors, firmwares will continue to be released with bugs and vulnerabilities. ABSR and Symbiotes are aimed at securing devices that run such firmware.

A. Autotomic Binary Structure Randomization (ABSR)

ABSR is a fortification technique currently being developed by the authors. This approach accepts arbitrary executables or firmware images as input and outputs a hardened, functionally equivalent variant of the original. The exploitability of the input binary is reduced by two primary operations: autotomic binary reduction and binary structure randomization. First, unused code, as determined by the particular configuration state of the target device, is autotomically removed in order to reduce the potential vulnerable attack surface of the overall system. For example, if a network printer is not configured to support LDAP authentication and UPnP, code sections corresponding to these feature sets are programmatically stripped from the resulting binary.

Furthermore, the autotomic operation can remove the binaries of features that are enabled by default but can not be disabled via configuration, which was precisely the case of the HP-RFU vulnerability. The RFU firmware update feature is rarely used but is enabled by default on all systems, some of which had no method of administratively disabling this code path; ABSR would remove the binary executables associated with the feature. Using the free space generated by the autotomic reduction phase, the binary structure randomization phase restructures the remaining executable binary blobs. We propose disabling and removing all unused features in general. However, the firmware update feature is a special case in which the code path should be disabled but not removed from the binary since this feature is necessary for future firmware updates. In this case, we propose an alternative method of enabling this code path, potentially through an ABSR configuration interface.

This approach differs from most existing techniques in that no attempt is made to remap coherent blocks of code into randomized locations in memory. Instead, ABSR decomposes all remaining basic blocks of the binary in order to transform them into a randomized, functionally equivalent program while intentionally breaking control-flow isomorphism.

Like software Guards and Symbiotes, ABSR is not a stop-the-world defense mechanism. ABSR does not halt the original functionality of the protected device while it is engaged. Like other randomization techniques such as Address Space Layout Randomization (ASLR) and Instruction-Set Randomization (ISR), ABSR is built into the architectural design of the protected device and does not require dynamic patching or binary rewriting like DynamoRIO. ABSR is an topic of ongoing research.

B. Software Symbiotes

Software Symbiotes [48] are a host-based defense mechanism that are specifically designed to inject intrusion detection functionality into the binary firmware of existing embedded devices. A Symbiote is a code structure embedded *in situ* into the firmware of an embedded system. The Symbiote tightly co-exists with its host executable in a mutually defensive arrangement, sharing computational resources with its host while simultaneously protecting the host against exploitation and unauthorized modification. The Symbiote is stealthily embedded in a randomized fashion within an arbitrary body of firmware to protect itself from removal and unauthorized deactivation. Unlike remote software attestation techniques, Guards and Symbiotes do not require the disabling of interrupts or a full system halt while the security mechanisms are engaged.

IX. Conclusion

We presented a general discussion of firmware modification attacks against embedded systems as well as a specific case study of such a vulnerability found in nearly all HP LaserJet printers. We discussed the analysis process that led to the discovery of the HP-RFU vulnerability as well as the implementation of a proof of concept printer malware. The printer malware presented in this paper can be delivered through standard PJL commands and can be embedded in innocuous document formats such as PostScript. It is capable of stealthy network reconnaissance, data exfiltration and propagation by autonomously compromising general purpose computers and other embedded device types. The HP-RFU vulnerability exploits a fundamental design flaw found not only in nearly all LaserJet printers, but in other modern embedded systems as well. Thus, the process presented in this paper can be generalized and applied to the exploitation of similarly vulnerable embedded systems.

We presented the results of exhaustive scans of IPv4 to track the size and distribution of all publicly accessible vulnerable LaserJet printers over time. Out of over 90,000 vulnerable units, only 1.08% of the vulnerable population has been

[7] Autotomy - The spontaneous casting off of parts is a (biologically) viable security mechanism.

patched since the release of firmware updates in response to the disclosure of HP-RFU. Furthermore, 24.8% of all patched printers are configured to have open telnet interfaces with no root password. In other words, we only identified 766 printers out of over 90,000 units that are simultaneously not vulnerable to the HP-RFU attack and have properly configured root passwords.

Firmware update signing can mitigate the HP-RFU vulnerability. However, it should not be used as the sole security mechanism on embedded systems. We presented the results of the analysis of all available firmwares for 63 HP LaserJet printer models that identify third-party libraries with known vulnerabilities within the signed codebase. We identified vulnerable third-party libraries in 80.4% of all firmware images analyzed. Furthermore, we identified libraries containing vulnerabilities that have been known for over eight years in several of the most recently released firmware images.

The scientific evidence, quantitative analysis and the proof of concept HP-RFU vulnerability exploitation presented in this paper demonstrate the importance of introducing effective host-based defense into vulnerable embedded devices.

ACKNOWLEDGMENTS

This work was partially supported by DARPA Contract, CRASH Program, SPARCHS, FA8750-10-2-0253. This work was supported by the United States Air Force Research Laboratory (AFRL) through Contract FA8650-10-C-7024. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA or the U.S. Government.

We would like to thank Jatin Kataria for his help in creating the HP LaserJet toolkit described in Section III and Anup Kotalwar for his contributions to the collection and analysis of data presented in Section V.

REFERENCES

[1] HP, "HP Security Solutions FAQ," http://h30046.www3.hp.com/large/solutions/hp_secsolutions.pdf, 2006.

[2] ——, "SSRT100692 rev.6 - Certain HP Printers and HP Digital Senders, Remote Firmware Update Enabled by Default," http://h20000.www2.hp.com/bizsupport/TechSupport/Document.jsp?objectID=c03102449, 2012.

[3] IDC, "Worldwide Hardcopy Peripherals Market Recorded Double-Digit Year-Over-Year Growth in the Second Quarter of 2010, According to IDC," http://www.idc.com/about/viewpressrelease.jsp?containerId=prUS22476810&sectionId=null&elementId=null&pageType=SYNOPSIS, 2010.

[4] A. Bellissimo, J. Burgess, and K. Fu, "Secure Software Updates: Disappointments and New Challenges," *Proceedings of USENIX Hot Topics in Security (HotSec)*, 2006.

[5] HP, "SSRT100692 rev.3 - Certain HP Printers and HP Digital Senders, Remote Firmware Update Enabled by Default," http://seclists.org/bugtraq/2012/Jan/49, 2012.

[6] I. Arce, "The Rise of the Gadgets," *IEEE Security and Privacy*, vol. 1, no. 5, pp. 78–81, 2003.

[7] S. Hanna, R. Rolles, A. Molina-Markham, P. Poosankam, K. Fu, and D. Song, "Take Two Software Updates and See Me in the Morning: The Case for Software Security Evaluations of Medical Devices," in *Proceedings of the 2nd USENIX conference on Health security and privacy*. USENIX Association, 2011, p. 6.

[8] C. Miller, "Battery Firmware Hacking," in *Black Hat USA*, 2011.

[9] B. Jack, "Jackpotting Automated Teller Machines Redux," in *Black Hat USA*, 2010.

[10] pt, "Ooops I hacked My PBX," in *The 28th Chaos Communication Congress*, 2011.

[11] A. Costin, "Hacking MFPs," in *The 28th Chaos Communication Congress*, 2011.

[12] K. Chen, "Reversing and Exploiting an Apple Firmware Update," in *Black Hat USA*, 2009.

[13] A. J. Aviv, P. Cerný, S. Clark, E. Cronin, G. Shah, M. Sherr, and M. Blaze, "Security Evaluation of ES&S Voting Machines and Election Management System," in *EVT*, D. L. Dill and T. Kohno, Eds. USENIX Association, 2008, p. 11.

[14] A. Cui and S. J. Stolfo, "A Quantitative Analysis of the Insecurity of Embedded Network Devices: Results of a Wide-Area Scan," in *ACSAC*, C. Gates, M. Franz, and J. P. McDermott, Eds. ACM, 2010, pp. 97–106.

[15] M. Sutton, "Corporate Espionage for Dummies: The Hidden Threat of Embedded Web Servers," in *Black Hat USA*, 2011.

[16] A. Cui, J. Kataria, and S. J. Stolfo, "Killing the Myth of Cisco IOS Diversity: Recent Advances in Reliable Shellcode Design," in *Proceedings of the 5th USENIX conference on Offensive technologies*. USENIX Association, 2011, pp. 3–3.

[17] HP, "Hewlett-Packard LaserJet 4200/4300 Series Printers - Firmware Update/Download Release/Installation Notes," http://ftp.hp.com/pub/printers/software/lj4200lbreadmefw.txt.

[18] Arduino, http://arduino.cc/.

[19] IDA, "The IDA Pro Disassembler and Debugger," http://www.hex-rays.com/idapro.

[20] CERT, "CERT Advisory CA-2002-07 Double Free Bug in zlib Compression Library," http://www.cert.org/advisories/CA-2002-07.html, 2002.

[21] ——, "zlib inflate() routine vulnerable to buffer overflow," http://www.kb.cert.org/vuls/id/680620, 2005.

[22] ——, "The zlib compression library is vulnerable to a denial-of-service condition," http://www.kb.cert.org/vuls/id/238678, 2004.

[23] A. Cui, "Print Me If You Dare: Firmware Modification Attacks and the Rise of Printer Malware," in *The 28th Chaos Communication Congress*, 2011.

[24] VxWorks socklib, http://www-kryo.desy.de/documents/vxWorks/V5.5/vxworks/ref/sockLib.html.

[25] CERT, "CVE-2011-4161," http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2011-4161, 2011.

[26] HP WebJet Admin, http://tinyurl.com/ch3g72f.

[27] HP, "SSRT100692 rev.2 - Certain HP Printers and HP Digital Senders, Remote Firmware Update Enabled by Default," http://seclists.org/bugtraq/2011/Dec/175, 2011.

[28] ——, "SSRT100692 rev.1- Certain HP Printers and HP Digital Senders, Remote Firmware Update Enabled by Default," http://seclists.org/bugtraq/2011/Dec/3, 2011.

[29] Spansion, "SPANSION S25FL064P Data Sheet," http://www.spansion.com/Support/Datasheets/S25FL064P.pdf, 2011.

[30] F. Lindner, "Design Issues and Software Vulnerabilities in Embedded Systems," in *Black Hat Windows Security*, 2003.

[31] SecurityFocus, "Sec. Vulenrability in ChaiVM EZloader," http://www.securityfocus.com/advisories/4317, 2002.

[32] Offensive Security, "Virata EmWeb R6.0.1 Remote Crash Vulnerability," http://www.exploit-db.com/exploits/12095/, 2010.

[33] T. Rad and Taegue, "SCADA and PLC Vulnerabilities in Correctional Facilities," in *The 28th Chaos Communication Congress*, 2011.

[34] V. Prevelakis and D. Spinellis, "The Athens Affair," *Spectrum, IEEE*, vol. 44, no. 7, pp. 26–33, 2007.

[35] Dronebl, "Network Bluepill," http://www.dronebl.org/blog/8, 2008.

[36] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel, "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses," in *Proceedings of the 29th Annual IEEE Symposium on Security and Privacy*, May 2008, pp. 129–142.

[37] R. Riley, X. Jiang, and D. Xu, "Guest-Transparent Prevention of Kernel Rootkits with VMM-Based Memory Shadowing," in *RAID*, ser. Lecture Notes in Computer Science, R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., vol. 5230. Springer, 2008, pp. 1–20.

[38] C. Krügel, W. K. Robertson, and G. Vigna, "Detecting Kernel-Level Rootkits Through Binary Analysis," in *ACSAC*. IEEE Computer Society, 2004, pp. 91–100.

[39] Z. Wang, X. Jiang, W. Cui, and X. Wang, "Countering Persistent Kernel Rootkits Through Systematic Hook Discovery," in *RAID*, ser. Lecture Notes in Computer Science, R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., vol. 5230. Springer, 2008, pp. 21–38.

[40] M. C. Rinard, "Manipulating Program Functionality to Eliminate Security Vulnerabilities," in *Moving Target Defense*, ser. Advances in Information Security, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, Eds. Springer, 2011, vol. 54, pp. 109–115.

[41] DynamoRIO, "Dynamic Instrumentation Tool Platform," http://dynamorio.org/.

[42] A. Seshadri, A. Perrig, L. van Doorn, and P. K. Khosla, "SWATT: SoftWare-based ATTestation for Embedded Devices," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2004, pp. 272–282.

[43] Y. Li, J. M. McCune, and A. Perrig, "SBAP: Software-Based Attestation for Peripherals," in *Trust and Trustworthy Computing, Third International Conference, TRUST 2010, Berlin, Germany, June 21-23, 2010. Proceedings*, ser. Lecture Notes in Computer Science, A. Acquisti, S. W. Smith, and A.-R. Sadeghi, Eds., vol. 6101. Springer, 2010, pp. 16–29.

[44] ——, "VIPER: Verifying the Integrity of PERipherals' Firmware," in *ACM Conference on Computer and Communications Security*, Y. Chen, G. Danezis, and V. Shmatikov, Eds. ACM, 2011, pp. 3–16.

[45] H. Chang and M. J. Atallah, "Protecting Software Code by Guards," in *Digital Rights Management Workshop*, ser. Lecture Notes in Computer Science, T. Sander, Ed., vol. 2320. Springer, 2001, pp. 160–175.

[46] binwalk, http://code.google.com/p/binwalk.

[47] A. Cui, "Embedded Device Firmware Vulnerability Hunting Using FRAK," in *Black Hat USA*, 2012.

[48] A. Cui, J. Kataria, and S. J. Stolfo, "From Prey to Hunter: Transforming Legacy Embedded Devices into Exploitation Sensor Grids," in *ACSAC*, R. H. Zakon, J. P. McDermott, and M. E. Locasto, Eds. ACM, 2011, pp. 393–402.

[49] R. Lippmann, E. Kirda, and A. Trachtenberg, Eds., *Recent Advances in Intrusion Detection, 11th International Symposium, RAID 2008, Cambridge, MA, USA, September 15-17, 2008. Proceedings*, ser. Lecture Notes in Computer Science, vol. 5230. Springer, 2008.

APPENDIX

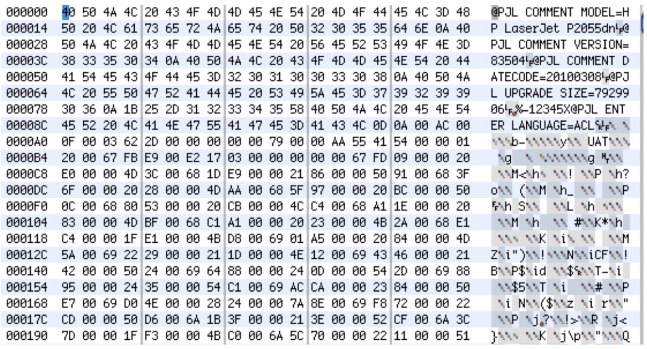

Fig. 8. Hex dump of a typical HP-RFU. For P2055DN, using the undocumented PJL/ACL language.

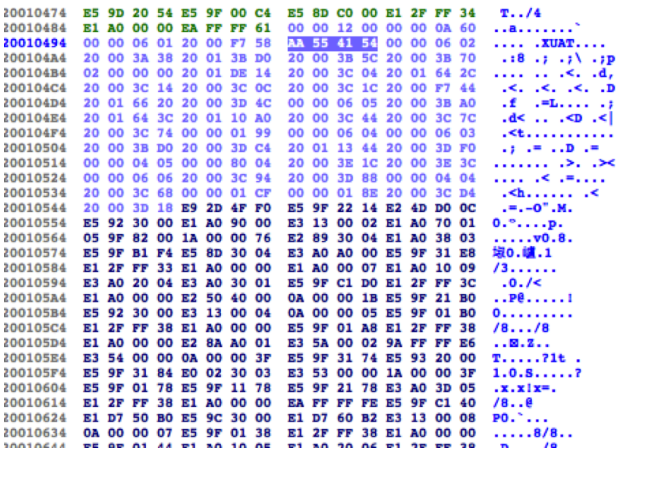

Fig. 9. "UAT" table structure. Contains a checksum value, followed by a directory manifest describing various compressed components of the binary update package.

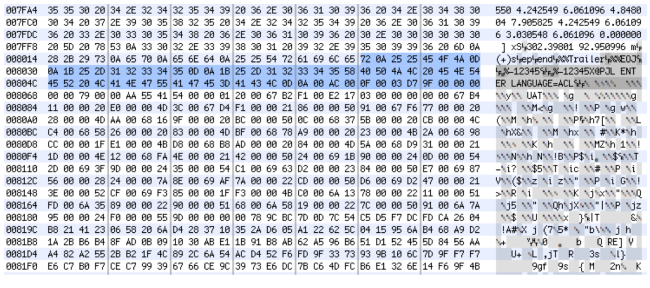

Fig. 10. RFU binary embedded inside a typical PostScript file. This illustrates the most straightforward reflexive attack.

Model	RFUs (qty.)	Earliest RFU	Latest RFU
2300	2	2004-05-12	2004-12-03
2400	4	2004-09-02	2009-06-24
3000	2	2004-01-06	2008-04-09
3500	3	2004-01-19	2007-02-20
3550	2	2004-09-22	2005-03-07
3600	2	2006-08-07	2006-08-28
3700	3	2004-03-31	2006-12-06
3800	1	2008-04-08	2008-04-08
4100	2	2004-10-08	2005-12-21
4200	2	2004-10-07	2005-06-02
4250	9	2004-09-02	2011-04-06
4300	2	2004-10-07	2005-06-02
4345	10	2005-01-25	2011-04-29
4600	2	2004-10-12	2006-10-10
4650	3	2004-08-27	2007-04-19
4700	7	2009-06-05	2011-05-11
4730	8	2009-06-04	2011-04-29
5100	1	2004-01-15	2004-01-15
5200	9	2009-06-04	2011-12-14
5500	3	2004-10-07	2005-06-02
5550	10	2004-07-29	2011-04-06
8000	1	2010-10-28	2010-10-28
8150	2	2004-01-14	2004-10-14
8500	6	2010-10-25	2011-06-29
9000	3	2004-08-09	2005-12-21
9050	21	2004-06-30	2011-12-13
9055	1	2008-02-20	2008-02-20
9065	5	2004-09-10	2008-02-20
9200	8	2005-01-25	2011-04-19
9250	10	2009-06-04	2011-12-19
9500	12	2004-10-24	2011-05-24
CM1312	5	2010-06-16	2011-12-09
CM1415	6	2010-07-21	2011-12-15
CM3530	9	2009-06-04	2011-12-13
CM4730	11	2009-06-04	2011-12-12
CM6040	8	2009-09-09	2011-12-12
CM80	5	2008-10-28	2010-08-05
CP1518	3	2010-06-16	2011-12-10
CP1525	6	2010-07-21	2011-12-15
CP2024	3	2010-05-12	2011-12-08
CP3505	8	2009-06-04	2011-04-06
CP3525	10	2008-12-04	2011-12-12
CP4005	6	2009-06-05	2011-05-11
CP4525	7	2010-01-20	2011-12-13
CP5225	5	2011-12-20	2011-12-20
CP6015	9	2009-06-04	2011-12-12
M1522	2	2011-03-19	2011-12-12
M1536	5	2010-07-21	2011-12-15
M2727	3	2010-09-02	2011-12-12
M3035	17	2009-06-05	2011-12-12
M4345	15	2009-06-05	2011-12-12
M5035	15	2009-06-05	2011-12-12
M9050	9	2009-06-05	2011-12-12
P2035	4	2011-03-30	2011-12-13
P2055	9	2009-04-30	2011-12-14
P3005	9	2009-06-15	2011-04-06
P3015	8	2009-09-10	2011-12-13
P4015	10	2009-06-04	2011-12-14
Pro 100	1	2011-10-21	2011-10-21
T1200	2	2010-08-31	2011-07-06
T2300	2	2010-08-31	2010-10-28
T7100	4	2011-09-06	2011-11-05
Z6200	1	2011-11-05	2011-11-05

TABLE VII
PRINTER MODELS AND FIRMWARE IMAGES ANALYZED FOR VULNERABLE LIBRARIES.