Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

DFRWS USA 2016 — Proceedings of the 16th Annual USA Digital Forensics Research Conference

Deleting collected digital evidence by exploiting a widely adopted hardware write blocker

Christopher S. Meffert[*], Ibrahim Baggili, Frank Breitinger

Cyber Forensics Research & Education Group, Tagliatela College of Engineering, ECECS, University of New Haven, 300 Boston Post Rd., West Haven, CT, 06516, United States

Keywords:
Digital forensics
Digital forensic tool testing
Hardware write blocker
Root access
Anti-Forensics
Vulnerability
Frameworks
Gismo
NIST
TD3

A B S T R A C T

In this primary work we call for the importance of integrating security testing into the process of testing digital forensic tools. We postulate that digital forensic tools are increasing in features (such as network imaging), becoming networkable, and are being proposed as forensic cloud services. This raises the need for testing the security of these tools, especially since digital evidence integrity is of paramount importance. At the time of conducting this work, little to no published anti-forensic research had focused on attacks against the forensic tools/process. We used the TD3, a popular, validated, touch screen disk duplicator and hardware write blocker with networking capabilities and designed an attack that corrupted the integrity of the destination drive (drive with the duplicated evidence) without the user's knowledge. By also modifying and repackaging the firmware update, we illustrated that a potential adversary is capable of leveraging a phishing attack scenario in order to fake digital forensic practitioners into updating the device with a malicious operating system. The same attack scenario may also be practiced by a disgruntled insider. The results also raise the question of whether security standards should be drafted and adopted by digital forensic tool makers.

© 2016 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Introduction

There is an ever growing need for collecting digital evidence from media, especially from hard drives. As of 2004, claims have been made that eighty to ninety percent of cases in the United States involve some sort of digital evidence (Rogers, 2006b). Since 2004, no doubt, computing devices have increased in ubiquity and decreased in size. A logical assumption can be made that this percentage may continue to increase, thus, upholding the notion for the necessity of digital evidence collection in an accurate and efficient manner.

Digital forensic investigation is defined by Ieong (2006) as “a process to determine and relate extracted information and digital evidence to establish factual information for judicial review”. If data on a disk drive can be considered evidence then one may argue that the whole disk should be considered evidence; both physically and digitally. If this is to be the case then it becomes critical that the integrity of the data is not compromised especially for the admissibility of evidence into the court of law (Argy and Mason, 2007; Accorsi, 2009; Givens, 2003). Landwehr (2001) defines integrity conceptually as “assuring that digital information is not modified (either intentionally or accidentally) without proper authorization”.

Methods, procedures and tools exist to ensure that evidence maintains its integrity throughout the digital forensics process. The two prominent tools in use today are software and hardware write blockers, with hardware write blockers being the preferred tool of choice.

[*] Corresponding author.
E-mail addresses: cmeff1@unh.newhaven.edu (C.S. Meffert), IBaggili@newhaven.edu (I. Baggili), FBreitinger@newhaven.edu (F. Breitinger).
URL: http://www.unhcfreg.com/, http://www.FBreitinger.de/

http://dx.doi.org/10.1016/j.diin.2016.04.004
1742-2876/© 2016 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

A software write blocker is a tool that handles write blocking at the software level via the mounting process. It ensures that the Operating System (OS) mounts the hardware with write blocking flags set to on. Software write blockers are easier to design and implement, but unless the write blocking settings are handled at the lowest levels possible (BIOS as an example), and the OS is secure, they tend to be easier to subvert (Lyle and Black, 2005).

A hardware write blocker is a device that attaches a host device (like a hard disk) typically to a forensic workstation with the purpose of preventing any possible modifications to the evidence drive before, during, and after the acquisition process. The name hardware write blocker comes from how the device prevents the write function from executing as it uses techniques for blocking writes to the media.

A hardware write blocker typically operates by breaking the bus that connects the hard drive to the host machine into two segments; a bus segment between the host and blocking device and another bus segment from the blocking device to the evidence drive. The two bus segments may consist of different protocols. One can be Small Computer System Interface (SCSI) and the other Advanced Technology Attachment (ATA). Once the devices are connected and the blocking device is powered on, all commands are intercepted by the blocking device. Once intercepted, the blocking device will filter any write commands from passing (Lyle, 2006). The Tableau TD3 used in this research is an example of a hardware unit that includes a hardware write blocker.

Initially, hardware write blockers were devices that simply blocked writes to disks after being connected to forensic workstations when digital media was either acquired or mounted for triage (Rogers et al., 2006). As products in this space continued to advance, devices became smarter, more efficient and packed with features. Devices such as disk duplicators with built-in hardware write blockers were developed to allow for use in forensic labs as well as on the field. As systems increased in size and storage, the need to accomplish network forensic imaging emerged. To tackle this challenge, these devices adopted networking features.

With this advancement came many benefits such as remote access via a user interface and the ability to remotely image a drive on a disk of interest. Tableau's TD3 model is one of these devices, and allows for browsing drives that are attached directly to the write blocker via the Internet Small Computer System Interface (iSCSI) protocol.[1] iSCSI works on top of the Transport Control Protocol (TCP) enabling the SCSI command to be delivered end-to-end over Local Area Networks (LANs), Wide Area Networks (WANs) or the Internet. The Ditto Forensic FieldStation from WIEBETECH[2] is another hardware write blocker and disk duplicator that allows for remote cloning and duplication of drives via iSCSI. Both devices allow for creating and modifying users and the settings associated with them.

Since most devices are proprietary and costly, an open source hardware write blocker and forensic imager alternative was developed by the Digital Forensics Investigation Research laboratory (DigitalFIRE) at University College Dublin (UCD). Their project aimed at providing law enforcement in underdeveloped countries with a cheap yet effective substitute to expensive hardware write blockers. The open source hardware write blocker and imager encourages practitioners to purchase the necessary parts, download an open source application, and assemble a device titled FIREBrick. The cost for its parts is approximately \$200[3] (Tobin and Gladyshev, 2015).

Nevertheless, provided that evidence integrity is of paramount importance in digital forensics, we argue that it is important to test the security of these devices given their wide adoption by government and industry – especially due to their increased features and network connectivity. In this work the following contributions were accomplished:

- We present a primary study focused on the security of these hardware imaging and write blocking devices (In specific we tested the most widely adopted one – the Tableau TD3).
- We illustrate how one may gain root access to such equipment.
- We construct and share the results of a preliminary proof of concept attack against the integrity of the imaging process when using the Tableau TD3.
- We raise the much needed awareness within the digital forensics community for integrating security testing as part of the digital forensic tool testing process since digital forensic tool testing focuses on the accuracy and correctness of the tools without accounting for plausible security weakness.

The rest of the paper is organized as follows. In Sec. Related work, a review of the related literature is shared, setting the motivation for this work. In Sec. Tableau TD3, the widely adopted device used in our study – the Tableau TD3 – is presented. Sec. Methodology delineates the approach of gaining root access to the TD3, the constructed integrity attack scripts, and the testing approach used to validate the integrity attack. In Sec. Results the results are presented, followed by the limitation of our work in Sec. Limitations. The work presented is then discussed in Sec. Discussion, and concluded in Sec. Conclusion. Lastly, we open the door for future research in Sec. Future work.

Related work

The following sections review works related to digital evidence integrity. These works underpin the motivation for exploring the security of the TD3 device.

[1] https://www2.guidancesoftware.com/products/Pages/tableau/products/forensic-duplicators/td3.aspx (last accessed April 11, 2016).

[2] https://www.cru-inc.com/products/wiebetech/ditto_forensic_fieldstation/ (last accessed April 11, 2016).

[3] http://digitalfire.ucd.ie/?page_id=1011 (last accessed April 11, 2016).

Digital evidence integrity

Maintaining evidence integrity in all investigative domains both physical and digital is of chief significance. In the digital domain, this means evidence remains unchanged while it is being acquired, authenticated and analyzed. Although, one may argue that only minimal changes should be made especially in the areas of Small Scale Digital Device (SSDD) and memory forensics. Nonetheless, in traditional computer forensics, hardware write blockers have become the golden standard for maintaining the integrity of evidence during the acquisition and authentication processes. Authenticating the copy of the digital evidence is usually achieved via hashing functions. Other forms of verification may be utilized as well during the course of an investigation such as visual verification, digital signatures, written documentation, Cyclic Redundancy Checks (CRCs), encryption and other proprietary methods (Cosic and Baca, 2010).

Anti-forensics

Anti-forensics relates to impeding the investigative process through various means. There are examples of research being conducted with regards to anti-forensics. One example is the iSEC Partners fuzzing of Encase's software suite, which yielded several different bugs as a result of software issues (Homewood, 2012). With that said, anti-forensics work is often overlooked by digital forensic researchers as Baggili et al. (2012) pointed out that only 2% out of their ($n = 500$) analyzed scientific articles pertained to anti-forensics. As a result of this lack of attention to anti-forensics, research regarding hardware write blockers is nearly non existent.

To the best of our knowledge, only one security issue has been reported regarding hardware write blockers in the media.[4] The specific tool tested was a Ditto Forensic Field Station.[5] The attack vector was through the remote web interface. Several Cross Site Scripting (XSS) attacks were successful against the device.

As there are a number of anti-forensic tools and techniques available, Rogers (2006a) proposed the following high-level anti-forensic taxonomy:

- Data hiding
  - Encryption
  - Steganography
  - Other forms of data hiding
- Artifact wiping
  - Disk cleaning utilities
  - File Wiping
  - Disk degaussing/destruction techniques
- Trail obfuscation
- Attacks against computer forensic tools and processes

A complete discussion regarding the aforementioned taxonomy is beyond the scope of this article, however, it is of note that should one gain root access to a forensic write blocking/duplicating device, then many of the anti-forensic taxonomy elements may be abused by a potential adversary. Our work is most synonymous to attacks against computer forensic tools and processes as it aimed at testing the security of a write blocking/disk duplicating device while consecutively hindering the forensic process by tainting the integrity of the collected evidence.

Dependency on tools is a weakness that is easy to fall into. The issue with depending on tools is that while they are helpful in expediting the forensic process, they are not immune to attacks. One thing to consider is how tools commonly used in digital forensics may be exploited using anti-forensics, which is why tool testing is of paramount importance to the digital forensics community.

Digital forensic tool testing

Currently, the National Institute of Standards and Technology (NIST) is the most prevalent organization that leads an extensive Computer Forensics Tool Testing (CFTT) program. They built their testing efforts around the following overarching themes[6]: (1) Disk imaging (2) Forensic media preparation (3) Write block (Software) (4) Write block (Hardware) (5) Deleted file recovery (6) Mobile devices (7) Forensic file carving and (8) String search. Others have also conducted research on digital forensic tool testing.

Baggili et al. (2007) suggested a mechanism for testing mobile phone forensic tools. They proposed a systematic database driven approach for testing mobile forensic tools. By employing a database driven method, it allows researchers and academics to study error rates and ensures cataloging of the testing process.

Guo et al. (2009) devised a method of mapping fundamental functions required in computer forensic processes. Through this effort they were able to develop a functionally orientated validation and verification framework for available digital forensic tools.

Other testing literature pertained to collecting images of volatile memory to verify the accuracy of the tool being used. In one unique approach, researchers adopted a visualization method involving dot plots (typically used in bioinformatics) to evaluate the accuracy and amount of interference of a forensic memory imager (Inoue et al., 2011).

Bootable forensic environments were also explored for their robustness and ability to provide reliable evidence for a court of law such as the work by Mohamed et al. (2014). In specific, they tested Knoppix v7.0, Helix 3 Pro 2009R3, and Kali Linux v1.0. It is possible for the bootable media to modify the hard disk. However, if modifications are predictable, and can be documented, it is acceptable to use the tool. A differential forensic analysis (a term coined by

[4] http://www.heise.de/ix/meldung/Kritische-Sicherheitsluecken-in-Write-Blocker-entdeckt-2071582.html (last accessed April 11, 2016).

[5] http://www.cru-inc.com/products/wiebetech/ditto_forensic_fieldstation/ (last accessed April 11, 2016).

[6] http://www.cftt.nist.gov/Methodology_Overview.htm (last accessed April 11, 2016).

Garfinkel et al. (2012)) was performed to compare pre and post use of bootable forensic environments. The results indicated that Helix 3 Pro 2009R3 was the most robust system in terms of its forensic soundness.

New frameworks have also been proposed to test various forensic tools. Work by Anobah et al. (2014) proposed a testing framework known as Maxwell, Oliver and Shahzad (MOS) for mobile device forensics. The MOS framework built on and extended the test plan from NIST. It accomplished this by including assertions and new test actions that cover anti-forensics as they pointed out a weakness in NIST's framework is not accounting for anti-forensics.

Work by Knüfer (2014) presents another possible method for testing forensic tools. Their work attempted to structure digital forensic tool testing using a schema based approach with an effort of identifying vulnerabilities. In their work they showed that their approach was well suited to find flaws in digital forensic tools.

Given the wide adoption of TD3s, the Department of Homeland Security (DHS) released a report about tests conducted on the Tableau TD3. The report conveyed results of primarily testing the device's forensic imaging and indicated that the TD3 acquired all visible and hidden sectors completely and accurately (DHS, 2014). Even though these tests focused on the forensic accuracy of the device's imaging function, the work did not account for the fact that the TD3 is a networkable device that could potentially be exploited, possibly tainting the forensic soundness of the acquired evidence. In the upcoming Sec. Tableau TD3, we elaborate on the TD3 device that was used in this research.

Tableau TD3

The line of Tableau hardware write blockers from Guidance Software provides a selection of portable forensic imaging devices. These devices are stand alone allowing for portability. The latest models typically have a user interface allowing the user to control and interact with the device. The TD3 model shown in Fig. 1 is of interest due to its modular design and Graphical User Interface (GUI). Additionally, the device offers a web interface where users have the ability to access it using a username and password. This web based interface works in a standard browser as well as a mobile browser.

The device is also capable of collecting forensic data from Serial Advanced Technology Attachment (SATA), Universal Serial Bus (USB) 1.1/2.0/3.0, Serial Attached SCSI (SAS) and FireWire (1394A/B) drives.

Currently, the TD3 has two read/write ports, specifically SATA and USB connections. It has read only ports for all other connections described above. The TD3 also comes with an integrated Ethernet port. This allows for the ability to accomplish remote triage and network based imaging and write blocking. The device also contains a Secure Digital (SD) card slot. This SD card contains the OS and without it the device does not boot. The storage size of the SD card is 513.3 MB split into four partitions: three Linux partitions a Linux swap partition and a Windows FAT 32 partition.

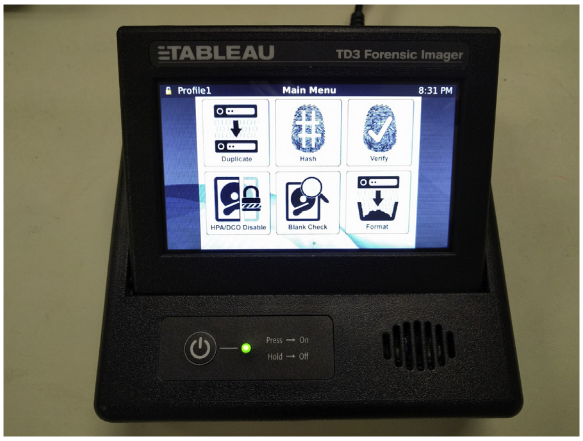

Fig. 1. Tableau TD3 hardware write blocker and disk duplicator.

While the TD3's ability to block write commands to the evidence drive has been shown to be robust (DHS, 2014), there remains other possible avenues to corrupt data on the collection (destination) drive. The process of finding ways to corrupt, hide, or destroy data falls into the realm of anti-forensics (see Sec. Anti-forensics).

Methodology

This research adopted a methodology which embodied the following three major steps:

Gaining root access: First, root access was required to exploit the device. Therefore, we performed reconnaissance on the device (details in Sec. Reconnaissance) and then replaced the firmware with a modified version (details are presented in Sec. Firmware update process).

Integrity attack scripts construction: Next, the Gismo process was identified. This process manages the calling of the different applications, duplication, hashing, etc. A script was developed to monitor Gismo and corrupt data when it stops. This is discussed in Sec. Data integrity compromise. The script can be downloaded from our website.[7]

Testing: Lastly, the script was tested and verified to guarantee that it corrupted the integrity of the data on the destination drive (see Sec. Testing).

Reconnaissance

In this stage of the methodology the TD3 was explored to recognize the OS, open ports and filesystem structure. When the TD3 was powered on, a boot up screen appeared which did not provide information about the OS and only indicated that a filesystem was loading. Once the loading process completed, the GUI appeared.

The device was connected to the network and Network Mapper v6.49 BETA 4 (NMAP) was employed to fingerprint the OS. The results yielded a Linux OS with a possible kernel version of 2.6.32–3.10. Nevertheless, the exact

[7] Script can be downloaded from http://www.unhcfreg.com under Data & Tools.

kernel version was determined only after gaining root access (see Sec. Firmware update process) and running uname -r. In fact, the results showed that the TD3 used kernel version 3.6.11.

To identify possible vulnerabilities (e.g., open ports, services), we executed the NMAP command nmap -sS -sV -T4 -O -A -v 10.101.1.149 which is the most invasive scan one can run. The results showed that only port 3260 iSCSI was open for firmware version 1.4.1. Using the same command after updating to firmware version 1.5, the following open ports were discovered: 22 OpenSSH v5.8, 80 Lighthttpd v1.4.35, 443 Lighthttpd v1.4.35, and 3260 iSCSI. This continued to be the case for all firmware updates following version 1.5.

Per Guidance Software's website, the OS is kept on the SD card. Mounting the SD card allowed for viewing and modifying the root filesystem. The card consisted of a FAT partition, two EXT4 partitions, and one swap partition. The next logical step was to attempt an SSH login to the TD3. On the first attempt an error message was presented indicating that only key exchange methods of authentication were allowed for accessing the TD3 via SSH.

Mounting the SD card and modifying the OS

Initially, the SD card that contained the OS for the TD3 was mounted on a machine running Kali 2.0 Linux. After the SD card was mounted, as a test, to examine if the contents could be changed, the sshd_conf file was modified and password authentication was enabled. An attempt was then made to log into to the TD3 via SSH proving that the filesystem could be modified by easily mounting the SD card. Details on how the password for logging into the SSH server was cracked is presented in Sec. Firmware update process. Although this method was sufficient for gaining root access to the TD3 device, it was important to explore other avenues of attack that a potential adversary may use to trick a practitioner into modifying the TD3 without physical access to the device. Therefore, the TD3's firmware update was the target of choice.

Firmware update process

This section describes how the firmware update installer was dissected and modified. *It is worth noting now that the firmware is digitally signed and while the modifications made were able to circumvent a hash match error, the modifications did cause the digital signature to be corrupted.* More details on this are listed below. It necessary to share all the tools used so the process could be replicated for further investigation. Throughout this process, the following tools were utilized (except for the Windows OS all other tools were freely available):

- Microsoft Windows 7
- Kali Linux 2.0
- Tableau Firmware Update v7.13.msi[8]
- LessMsi v1.4[9]
- Advanced Installer v12.6.1[10]
- Zip v3.0
- GZip v1.6

The firmware update file was available via Guidance Software's website as a Microsoft Installer .msi file. This file is an executable but can be expanded to view and extract its contents using Lessmsi v1.4. The extracted files revealed another compressed file; td3-update-1.x.x. After decompressing that file, three files; ext_datafs.gz, fat_datafs.gz and rootfs.gz were found. Extracting and mounting the rootfs.gz yielded a set of files that could be replaced on the device once the update was completed by modifying the root filesystem.

Although a variety of attack vectors could be implemented in the firmware update file such as modifying the TD3's web interface code to steal credentials, we chose the low hanging fruit – to modify the SSH server configuration file as an example to test our firmware update exploit. The server configuration file was located in the /etc directory and was configured to not allow password authentication but key exchange only. After adjusting this setting, it allowed for shell access with the correct username and password.

At this point it was necessary to re-compress the rootfs file and re-compress all of the files back into a .msi file using Advanced Installer v12.6.1. We then ran the installer in an attempt to move the changes to the existing filesystem on the TD3. This pushed the updated firmware v 1.5.0 to the TD3 but with the SSH server accepting password logins.

During and after the installation process there were only two indications that something was modified. The first indication that anything had been modified was the digital signature of the installer. The unmodified installer has Guidance Software, Inc. listed as the publisher. Once modified, the publisher was listed as unknown and Windows displayed a warning message indicating that the file does not have a valid signature. Work by Sharek et al. (2008) showed that of a sample size of 42 people 63% simply clicked "OK" on pop up boxes, simply just trying to get them to close. If the person tasked with installing the update was in a rush, or was not properly trained, there is a high chance they will simply click what is necessary to move forward.

The second was a single box warning that the installed update did not match the hash of the original downloaded firmware hash as shown in Fig. 2. To make the attack less obvious to a potential victim, we modified the error message by manually changing the executable using a Hex Editor to convince the user that the update was successful as shown in Fig. 3.

One item remained for gaining SSH access to the TD3 device, which is the root password. John the Ripper password cracker v1.8.0.6-jumbo-1-bleeding-omp was used to crack the root password. This was executed using the

[8] https://www2.guidancesoftware.com/products/Pages/tableau/products/forensic-duplicators/td3.aspx (last accessed April 11, 2016).

[9] http://lessmsi.activescott.com (last accessed April 11, 2016).

[10] http://www.advancedinstaller.com (last accessed April 11, 2016).

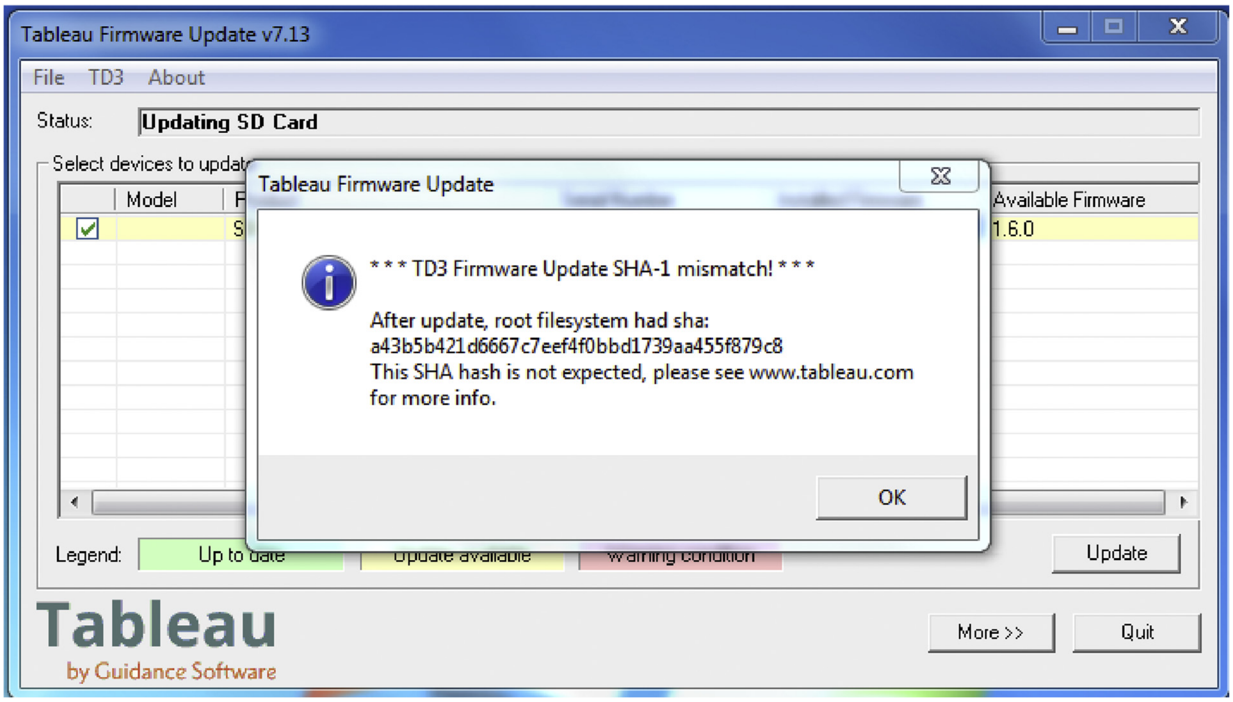
Tableau Firmware Update v7.13
File TD3 About
Select devices to update
Model
Available Firmware
Legend: Up to date Update available Warning condition
Update
More >>
Quit
Tableau by Guidance Software

Status: Updating SD Card

1.6.0

Tableau Firmware Update

* * * TD3 Firmware Update SHA-1 mismatch! * * *

After update, root filesystem had sha:
a43b5b421d6667c7eef4f0bbd1739aa455f879c8
This SHA hash is not expected, please see www.tableau.com
for more info.

OK

Fig. 2. Warning message after firmware update.

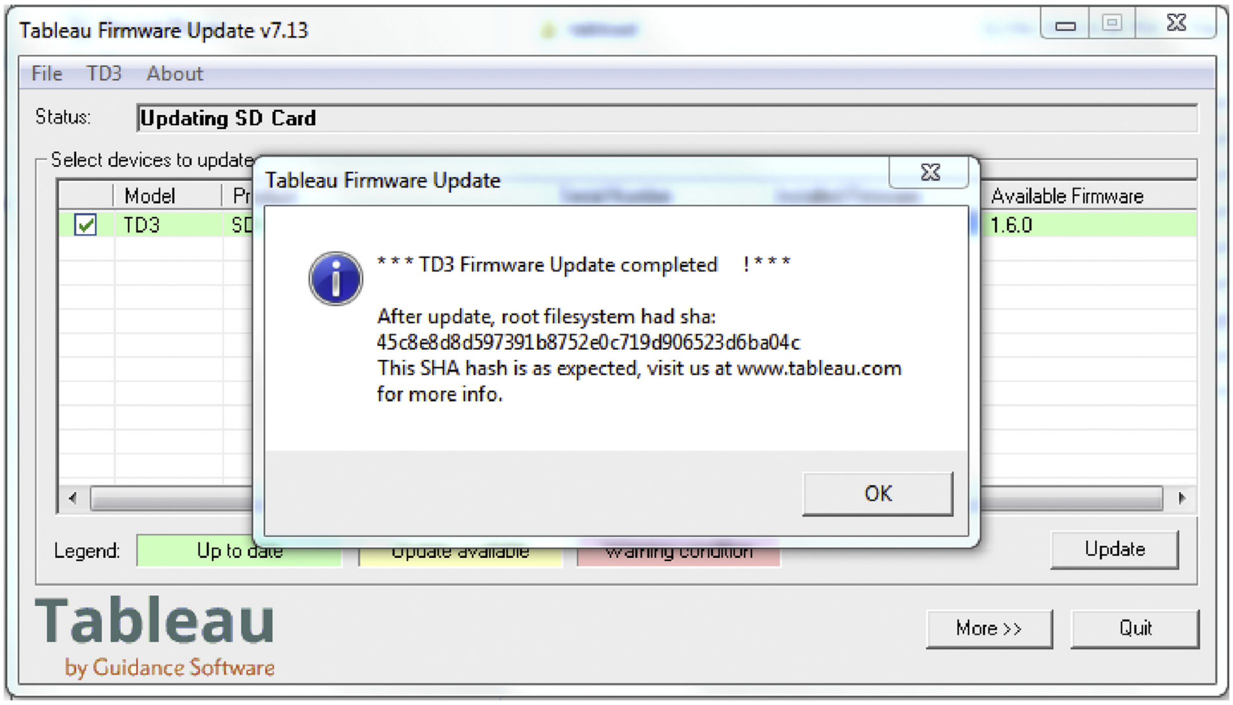
Tableau Firmware Update v7.13
File TD3 About
Select devices to update
Model
Available Firmware
Legend: Up to date Update available Warning condition
Update
More >>
Quit
Tableau by Guidance Software

Status: Updating SD Card

TD3 1.6.0

Tableau Firmware Update

* * * TD3 Firmware Update completed ! * * *

After update, root filesystem had sha:
45c8e8d8d597391b8752e0c719d906523d6ba04c
This SHA hash is as expected, visit us at www.tableau.com
for more info.

OK

Fig. 3. Modified warning message after firmware update.

command unshadow on /etc/passwd and /etc/shadow, piping this into a text file (filename.txt) and then running John on the output. If John succeeded in cracking a password it dumped it to /.john/john.pot. Issuing the command john –show filename.txt showed the username and password. In this case for user root, the password was simply *secret*.

Finally, an attempt was made to log into the SSH server with username root and the password *secret* and root access was granted. This illustrated that our approach for updating the .msi installer, and pushing the modified firmware to the TD3 was successful.

Data integrity compromise

After root access was acquired, thinking like the bad guys, an attempt at an integrity attack was designed. Initial thoughts were to redirect the commands issued when a process was selected via the GUI or to somehow modify the commands to corrupt the integrity of the collection drive. After gaining root access and logging in via SSH it was possible to monitor the device's processes. Issuing the command top[11] allowed monitoring of the processes running on the device. The goal of monitoring the processes was to attempt to identify any critical processes when an application was selected via the touch screen interface.

We were able to identify a spike in a process called Gismo, which used up to 90% Central Processing Unit (CPU) capacity during the disk duplication and hashing activities. When idle, Gismo only consumed approximately 0.0%–1.0% CPU usage.

Gismo is proprietary software that appears to be developed by Tableau to manage the applications provided on the touch screen of the device. Gismo was taken as a reference point. In other words, the %CPU use may be constantly monitored with a script and thus used as a “flag” to launch malicious code.

Script construction and pseudocode

A script was written to monitor Gismo's Process ID (PID) and when the process ramped down it issued the command dd if=/dev/zero of=/dev/sdXXX bs=1M. This command corrupted the copied image on the destination drive by overwriting the entire disk with zeros. This left the end user unaware that data was being corrupted since the process launched after the imaging process and verification stages were completed by the TD3.

When the TD3 runs the duplication process it generates a hash value at the end of its process. The script only executes after it sees the CPU usage drop below 1%. If the end user has the hash of the evidence drive and compares it to the duplication hash generated it will appear to match. The data itself meanwhile was corrupted by the aforementioned command. Two scripts were constructed to achieve this process. A bash script was created that checked the PID of Gismo automatically and then called the python script with the appropriate PID.

In Algorithm 1, we present the pseudocode for the main algorithm of the process.py script. The algorithm executes a while loop that calculates the CPU percent for the PID value passed in on execution of the script. It then proceeds to check for a change in the CPU percent, if this change has negative direction less than −10 percent and the current CPU percent is less than one percent, the DD command was executed. If both of these conditions were not meet, the previous CPU percent gets assigned to the current CPU percent and the loop executes again.

Algorithm 1 Process detection and execution of integrity attack using DD.

```
$previous \Leftarrow 0$
while $TRUE$ do                                                    ▷ Loop runs until 'break'
    $current$ = getCurrentCPUUsageOfGismo();                       ▷ Value between 0 and 100
    if (($current - previous$) $< -10$) && ($current < 1$) then   ▷ Checks CPU drop ($curr - prev$) & done ($curr < 1$)
        execProgram();                                             ▷ In our test, we run DD
        Break Loop;
    else
        sleep(1);                                                  ▷ Sleep for 1 s to not use all CPU
        $previous \Leftarrow current$;
    end if
end while
```

Testing

Testing the result of the script was necessary to validate this integrity attack. A full duplication was executed on the TD3 and a hash on the collection drive was run to verify that it was altered. The following tools were utilized in this testing process:

- Two identical 2 GB USB Flash Drives. One contained a file to be duplicated (source drive). The other was blank (destination drive).
- TD3 Forensic Imager with custom firmware.
- Network with a Dynamic Host Configuration Protocol (DHCP) server enabled.
- process.py and callprocess.sh scripts loaded on the TD3 device (in the attack verification stage).
- Computer with access to the same network the TD3 was connected to and an SSH client installed.

The test procedure was broken into three phases (I) calibration pre-firmware update (II) calibration post-firmware update and (III) attack verification.

Phase I, the calibration pre-firmware update, the source drive was first hashed. A disk to disk duplication process was completed from the source disk to the destination disk. The destination disk was hashed to verify that both the source disk and destination disk hash values matched post duplication. They matched as expected. This confirmed that the duplication was successful and the integrity of the process was validated.

[11] Top is a program that monitors the processes in Linux, similar to task manager in MS Windows.

Phase II, calibration post-firmware update, was the same as Phase I. We repeated the steps to verify that the modified firmware update did not affect the disk to disk duplication processes. Again, it was found that both the source drive and the destination drive hash values coincided.

Lastly, in Phase III, the custom script was first executed by SSHing into the TD3 device, and then the disk to disk duplication process commenced. After the disk duplication completed, the TD3 showed to the user that the destination drive hash value matched the hash values in Phases I and II. However, when the destination drive was removed and hashed, the hash value of the destination drive did not match as expected, confirming that our constructed script indeed compromised the integrity of the collected evidence.

Results

During the calibration phase, the results clearly verified that the TD3 operated as intended, and the hash values were:

```
CALIBRATION PHASE HASH VALUES
 Destination drive hash pre-firmware update
   MD5:1ac59e862aa1b49aeaf0b13a2b137b4e
   SHA1:7639363a0252b99108e6a7ae45d1ae3c76bfb502
 Destination drive hash post-firmware update
   MD5:1ac59e862aa1b49aeaf0b13a2b137b4e
   SHA1:7639363a0252b99108e6a7ae45d1ae3c76bfb502
```

The results also clearly validated that the integrity of the destination drive was compromised after running the integrity attack scripts on the TD3 device. The hash values from this testing phase are shown below:

```
SCRIPT RUNNING PHASE HASH VALUES
Destination drive hash shown to the user
   MD5:1ac59e862aa1b49aeaf0b13a2b137b4e
   SHA1:7639363a0252b99108e6a7ae45d1ae3c76bfb502
Actual hash value of the destination drive
   MD5:d352609773231d546b29766611ee0035
   SHA1:f8976d0b3b2f8dfbe3936e417bb03182902723e7
```

The results confirmed that the DD command was successful in its process of corrupting the destination drive. Recall, the DD command was executed only after the duplication and verification processes were completed by the TD3 as the goal was to corrupt the data whilst also making the user believe that the data on the destination drive was not compromised. To further validate this method, the script was executed, and the destination drive was removed even before the DD command completed its task. This was done to confirm that as soon as the hash verification process commenced by the TD3, and a hash value is shown to the user, the script was still capable of corrupting the destination drive.

Limitations

There are several limitations to this primary work along with other limiting factors that makes future research challenging.

From a vulnerability standpoint, the biggest challenge is the need for physical access to the device. While convincing somebody to push the modified firmware to the TD3 is not out of the question, having access to the network by SSHing into the device was the method utilized to execute the scripts. Although this is a limitation in our initial work, one may find a method to automate the script and repackage that into the firmware update.

On top of the needed physical access, with regards to the firmware update process, both a digital signature warning and a warning indicating that the firmware did not match its hash value were presented. The hash matching error was rectified by modifying the error message as discussed in Sec. Firmware update process. The original warning message is shown in Fig. 2 and the modified one is illustrated in Fig. 3.

In regards to the digital signature issue, it would take an end user that is intimate with the installation and upgrade process to possibly recognize this change.

Another limitation of this work is the constructed script. The DD command corrupted the destination drive by writing zeros to the entire disk. While this was useful in proving the ability to execute a process when desired, a more robust script would better hide its data corruption method; making it less obvious to the end user that a malicious anomaly has taken place.

Lastly, the nature of the TD3 embedded system is one that has been stripped of its tool packages, leaving only the bare-bone necessities to make the device function. This makes it challenging to add any additional tool packages. The only compiler that appears on the system is the Python 2.7 interpreter. It does contain a selection of packages to accomplish most of the basic scripting tasks.

Discussion

It is important to note the severity of our findings and articulate them in a scenario. One should understand that most, if not all digital forensic companies send product updates to their customers via e-mail. Links to updated firmware or software are usually embedded into a nicely formatted e-mail.

Now, imagine a situation where an adversary acquires the e-mails of digital forensic practitioners (easy to do by simply finding them on the web), and a phishing attack is conceived. It is also important to recognize that many Law Enforcement (LE) agency practitioners have only received training for using digital forensic tools and many do not possess a background in cybersecurity and/or computing.

The created phishing attack may target the list of e-mail recipients and ask them to download a compromised firmware update. The firmware may then be downloaded by a victim practitioner, and the installation process would be seamless. If this victim then attaches the compromised device to the network, the network the device is on may also now become compromised, plausibly allowing for reverse shell access into the TD3. This type of social engineering attack is not that far fetched.

Mitnick and Simon (2011) in their book The Art of Deception list many cases of social engineering that are far more elaborate and complex than the scheme we introduced. If the idea of socially engineering an end user to update the firmware may seem like stretch, perhaps the idea of the jaded insider is more convincing. Consider cases such as the leak of classified material by Edward Snowden. It is certainly not a stretch to believe an insider might be persuaded to install an updated firmware.

Provided the above scenario, it becomes clear that the limitation of physical access, while challenging to overcome, may be accomplished by tricking a victim into installing a compromised firmware update.

Another point that needs addressing are the ramifications of a physical device like the TD3 being compromised. The work developed in this paper was able to modify the device and plausibly deceive an end user ultimately affecting the authenticity of the acquired evidence.

As stated by Kerr (2001), before any party submits an electronic record or evidence (disk drive), it must be shown to be authentic and unaltered. Kerr (2001) also indicated that hash values are an acceptable method to validate the authenticity of a duplication. Currently, the script will allow for a hash value to be generated that convinces the end user of the authenticity of the generated hash value, while in reality, the destination disk was altered.

If the script was modified to seek pertinent incriminating files and modify or delete only those files, the consequence may appear less suspicious. It is likely that many more of the tools in digital forensics will continue to adapt features such as networking and remote access. As these features become more prominent, so will the need to test their security.

It is therefore of paramount importance for us to call for integrating security testing into the digital forensic tool testing process. We no longer live in a world where a computer forensic lab is completely isolated from a network. Network disk imaging is prevalent and devices such as the TD3 will continue to be used. We are also seeing moves towards a Forensics as a Service (FaaS) model, where testing the security of these services will also be of paramount importance. We finally note that the TD3 was used as a case study in our work due to its wide adoption by practitioners worldwide and its accepted forensic tool testing evaluation (DHS, 2014).

Conclusion

In this work, several goals were accomplished. Primarily, our goal was to test the security of digital forensic tools. We used the TD3, a widely adopted forensic duplicator and write blocker device as a case study. We were able to gain root access to the device, as well as modify the firmware update, creating the possibility of a plausible social engineering/phishing attack asking practitioners to update their devices with a compromised firmware update.

Furthermore, a method was presented by ways of using scripts that run on the device, that trick a potential practitioner into believing that the hash value of the imaged drive matched the source drive, however, as soon as that process commenced, the destination drive's integrity was corrupted. Our goal was to raise awareness for integrating security testing into the digital forensic tool testing process, as more digital forensic tools are being used in networked environments increasing the risk for adversaries to compromise the digital forensic process/tools.

Future work

This work opens the door for the security testing of forensic tools. We posit the importance of formulating a methodology for integrating security testing into the digital forensic tool testing framework and call on the community to account for this flaw. There is a body of knowledge that is untapped on how penetration testing and digital forensics intersect even though many consider them mutually exclusive domains of knowledge. Additionally, we see a need for developing concrete standards for the security of both hardware and software forensic tools so that both companies and researchers adhere to these standards when constructing digital forensic tools.

In terms of the actual TD3 device, further analysis of the processes running on the device to see if it is possible to manipulate them is viable. For example, a Lighthttpd web server is running on the TD3, so network analysis of the traffic may lead to other security shortcomings. As a preliminary test, the web interface files could also be found on the device's filesystem, and manipulating them to intercept credentials may be another avenue of attack.

Ways of also advancing the constructed integrity attack script is also possible. For example, DD may be used to reverse a file, write zeros into the middle of a file, or truncate a particular amount of bytes from the end of a file. A more elaborate script may also be designed to seek out potentially incriminating file types or files with known signatures in an attempt to modify or wipe them.

In terms of network imaging, exploring the possible corruption of network images via network layer attacks may also be tenable. Future work may also attempt to target both software layer and hardware layer attacks against the write blocked ports on the device, and not only the destination drives.

References

Accorsi R. Safe-keeping digital evidence with secure logging protocols: state of the art and challenges. In: IT security incident management and IT forensics, 2009. IMF'09. Fifth International conference on. IEEE; 2009. p. 94–110.

Anobah M, Saleem S, Popov O. Testing framework for mobile device forensics tools. J Digital Forensics Secur Law JDFSL 2014;9:221.

Argy PN, Mason S. Electronic evidence: disclosure, discovery and admissibility. LexisNexis Butterworths; 2007.

Baggili I, BaAbdallah A, Al-Safi D, Marrington A. Research trends in digital forensic science: an empirical analysis of published research. In: Digital forensics and cyber crime. Springer; 2012. p. 144–57.

Baggili IM, Mislan R, Rogers M. Mobile phone forensics tool testing: a database driven approach. Int J Digital Evid 2007;6:168–78.

Cosic J, Baca M. Do we have full control over integrity in digital evidence life cycle?. In: Information technology interfaces (ITI), 2010 32nd International conference on. IEEE; 2010. p. 429–34.

DHS. Test results for digital data acquisition tool:tableau td3 forensic imager version 1.3.0. 2014. URL: https://www.dhs.gov/sites/default/files/publications/508_Test%20Report_NIST_Tableau%20TD3%20Forensic%20Imager%201.3.0_August%202015_Final_0.pdf.

Garfinkel S, Nelson AJ, Young J. A general strategy for differential forensic analysis. Digit Investig 2012;9:S50–9.

Givens JS. Admissibility of electronic evidence at trial: courtroom admissibility standards. Cumb Law Rev 2003;34:95.

Guo Y, Slay J, Beckett J. Validation and verification of computer forensic software tools searching function. Digit Investig 2009;6:S12–22.

Homewood AJ. Anti-forensic implications of software bugs in digital forensic tools [Ph.D. thesis]. Auckland University of Technology; 2012.

Ieong RS. Forza–digital forensics investigation framework that incorporate legal issues. Digit Investig 2006;3:29–36.

Inoue H, Adelstein F, Joyce RA. Visualization in testing a volatile memory forensic tool. Digit Investig 2011;8:S42–51.

Kerr OS. Searching and seizing computers and obtaining electronic evidence in criminal investigations [US Department of Justice, Computer Crime and Intellectual Property Section]. 2001.

Knüfer P. Mitigating anti-forensics: a schema-based approach. 2014.

Landwehr CE. Computer security. Int J Inf Secur 2001;1:3–13.

Lyle JR. A strategy for testing hardware write block devices. Digit Investig 2006;3:3–9.

Lyle JR, Black PE. Testing bios interrupt 0x13 based software write blockers [Retrieved January, 28, 2007]. 2005.

Mitnick KD, Simon WL. The art of deception: controlling the human element of security. John Wiley & Sons; 2011.

Mohamed AFAL, Marrington A, Iqbal F, Baggili I. Testing the forensic soundness of forensic examination environments on bootable media. Digit Investig 2014;11:S22–9.

Rogers M. Anti-forensics: the coming wave in digital forensics [Retrieved September, 7, 2008]. 2006.

Rogers M. Dcsa: a practical approach to digital crime scene analysis. West Lafayette: Purdue University; 2006b.

Rogers MK, Goldman J, Mislan R, Wedge T, Debrota S. Computer forensics field triage process model. In: Proceedings of the conference on digital forensics, security and law; 2006. p. 27 [Association of Digital Forensics, Security and Law].

Sharek D, Swofford C, Wogalter M. Failure to recognize fake internet popup warning messages. 2008.

Tobin L, Gladyshev P. Open forensic devices. J Digital Forensics Secur Law 2015;10:97–104.