Have You Been Framed and Can You Prove It?

D.O. Lawal[*], D.W. Gresty*, D.E. Gan[*], and L. Hewitt[**]
[*] University of Greenwich/School of Computing and Mathematical Sciences, London, United Kingdom
[**] University of Greenwich/School of Law and Criminology, London, United Kingdom
D.O.Lawal@greenwich.ac.uk
D.W.Gresty@greenwich.ac.uk
D.Gan@greenwich.ac.uk
Louise.Hewitt@greenwich.ac.uk

***Abstract* - This work addresses the potential for a frameup attack through the use of a programmable USB e.g., a 'Rubber Ducky' to plant false evidence on someone else's computer. The aim is to determine who performed these actions, the human or the Rubber Ducky. Experiments were undertaken where a human interacted with a computer and a Rubber Ducky performed the same actions using identical computers, with identical baseline configurations, to detect differences in the artifacts left behind in each case. Forensics images generated from each experiment were analysed using forensics tools. Our findings pose the question can a programmable USB device be used to masquerade as a human, and can the forensic analyst or legal counsel make informed decisions about the provenance of any artifacts identified, as the expert may not be able to differentiate between the actions of the human user or the programmable USB, which could lead to a miscarriage of justice. This work alerts investigators and experts to the potential presence of a programmable USB device, and presents some artifacts that show that a programmable USB could have carried out these actions, which might prevent an innocent individual being wrongfully convicted of a crime they did not commit.**

Keywords - Rubber Ducky, Programmable USB, Hardware Hacking, Miscarriage of Justice, Framing attack, Digital Evidence

I. Introduction

The aim of this work is to test the feasibility of using a programable USB, such as the Rubber Ducky to perform a frameup attack to incriminate another user by planting false evidence of indecent photographs and a matching web history on the victim's machine while they are logged in or their session is active. We determine if it is possible during a forensic investigation, to differentiate between the user or the programmable USB performing the same set of actions or if any evidence of what has occurred are found that can indicate the device did the download and not the user. This has the potential to lead to a miscarriage of justice where the innocent user is charged with the crime of "possession of indecent photographs or child pornography" if found guilty.

A programable USB is not necessarily a malicious USB, as they can be used in task automation by system administrators, penetration testers, etc. These devices have a processor that is able to process information or carry out instructions. A programable USB could be pre-programmed to do almost anything a person can do on a computer, and it will do these actions faster, more accurately and most importantly, more subtly unlike when there is a person operating the computer. This is one of the reasons why it might be possible to use devices like this to plant false evidence or implicate someone else. This could ultimately lead to a miscarriage of justice, which can occur when an innocent person is prosecuted for a crime they did not commit or when a guilty person is acquitted of a crime they committed.

A disadvantage of these devices is their ability to carry out USB delivered exploits [1], steal passwords, and cause some other form of disruption. Furthermore, they are often detected as a Human Interface Device (HID) which makes input from them trusted by the computer (as it believes they are keystrokes entered by the user), and able to avoid detection by anti-virus software.

Framing attacks are not new threats. For example, it was quite easy to fool facial recognition systems only by presenting a 2D picture instead of the actual face, or by cosmetic surgery [2]. Now, it is more difficult, requiring more effort but still not impossible as discussed and demonstrated in [3]. Framing can be described as presenting or planting false evidence to make it appear as if someone else is guilty of a crime. Knowledge and awareness of forensic techniques has made it easier and at the same time more difficult to ascertain that a crime was indeed committed by the person whose attributes (fingerprint or DNA for example) match those found at a crime scene. This is because it is possible to mislead some procedures used in forensic investigations.

There are several reasons why a target might be framed (i.e., implicated). It could be in a corporate environment by a colleague, or by a company to implicate a competitor and take them out of business. For example, Schneier [2] mentioned how bots were built to click on the AdSense ads of competitors so Google's anti-fraud system could detect them as trying to inflate their own revenue and shut them down as a result. This is a subtle form of framing attack that leveraged the security mechanism in place; making it look like the competitors were the ones clicking for their own profit. A frameup could be done by a Company to have a reasonable cause to fire an unwanted employee etc.

Many articles have addressed how much damage could be done via USB devices and based on this awareness, some organisations have either limited or totally prohibited the use of USB devices (flash or USB storage drives) because of their capability to deliver malware, data theft or some other malicious actions. Moreover, industry standards such as the ISO 27002 [4] have also advised against enabling USB drives except where they are necessary for business use. Moreover, many individuals are now more careful of links they click or sites they visit to avoid malware or the likelihood of being hacked. These are all good security practices but what about situations where the users do not have to click anything or do anything, or when the device is not seen as a regular USB? It might be interesting to know that this kind of device also works on smart phones and not just computers or servers; thus, making it possible to frame mobile users as well.

In this study, the possibility of a framing attack via a programmable USB (or Bad USB) is investigated, using forensic software. Could this lead to a miscarriage of justice where an innocent person is prosecuted? The rest of this paper is structured as follows, Literature review, Experimental Setup, Tools and Technologies used, Experimental Scenarios, Experiment results and Conclusions.

II. RELATED WORK

There are a number of related works reporting framing attacks, how they can be used for malicious purposes and on the potency of programmable USBs. Gelernter, Grinstein and Herzberg [5] identified the threat of cross-site framing attacks. Cannols and Ghafarian [6] discussed the use of a Rubber Ducky to obtain user passwords from a Windows machine. Harianto and Gunawan, [7] also used the Rubber Ducky to steal Wi-Fi passwords from computers in a matter of minutes. The work by [5] demonstrated how easy it is to frame someone for an attack and were able to plant false evidence that outsmarted forensic software. The framing attack was possible because it took advantage of the vulnerabilities of browsers and weaknesses in the forensic investigation software [5]. A cross-site request forgery (CSRF) attack was used to plant false evidence on the victim computer without requiring access to it and to also plant evidence in the logs of some popular websites. The attack left no discernible traces on the victim machine and there was also no need to engage client-side malware (i.e., something delivered to the target machine which might need to be triggered by user action or a service). The presence of malware on the client machine would have given the investigator more tangible evidence which could be reverse engineered, e.g. to analyse its behaviour, its components and what services or programs it impacts. This could be a lead to other deductions or inferences by an analyst. The outcome of the framing attack was validated with the help of forensics experts from the Israeli Law, Information and Technology Authority (ILITA) and the Israeli Police. CSRF attacks usually leverage social engineering techniques such as sending malicious links over email or using a public forum to trick the target into sending a forged request to a server. This is also sometimes referred to as session riding. Basically, it is an attack that coerces the user into taking actions they never intended [8]. The success of planting the fake evidence in [5] was considered an indication that forensic experts could be misled. Needless, a misled forensic expert equals misleading testimony, which equals a potential justice miscarriage. Miscarriages of justice can occur to someone who is guilty of a crime, as well as someone who is not guilty of a crime.

The sophistication of attacks, delivery methods and the malicious tools that are widely available is increasing. An example is the emergence of tools like reprogrammable USB devices or Bad USBs. Some of these USB devices can register as HIDs such as a mouse, joystick or keyboard. The ones that act like keyboards for example have the ability to send keystrokes to the device they are connected to. Examples of damage that could be done through this include but are not limited to launching a computer terminal or a command prompt, stealing data, creating a backdoor. Devices like this are called schizophrenic devices [1]. Beitler [1] also talks about the possibility of the Rubber Ducky being used to perform USB delivered exploits. Using the Rubber Ducky, it is possible to load a “trusted” root certificate into a victim's machine, launch a Man-in-the-middle (MitM) attack, and decrypt Secure Socket Layer (SSL) traffic. A Rubber Ducky was used in [6] for a hacking experiment. One of their aims was a proof of concept of how much harm could be done in a matter of seconds with such a device. The Rubber Ducky was used to obtain clear text password credentials from a Windows machine. This programmable USB (or Bad USB) takes advantage of its ability to masquerade as a human interface device (HID) device and the ability of HID devices to be automatically detected and accepted by many of the operating systems in use today [6]. They can execute malicious instructions on the target device covertly and in a sophisticated manner.

There has been more focus on providing solutions to counter software attacks. This could be attributed to the higher occurrence rate of software-based attacks over hardware-based [9]. Paying less attention to this area has led to a deficiency in security defences against hardware attacks. This, according to [9] has in turn led to a rise in the threat of hardware-based attacks.

Comparing the software and hardware keyloggers, there are more advantages to using hardware-based attacks for the perpetrator. Such advantages include flexibility and stealth as they can operate without computer resources, they are platform independent and antivirus programs cannot detect them [9]. Threats from the inside are more dangerous than those that originate from the outside [10], and in most cases, these hardware attacks leverage some form of social engineering or compromised insider (e.g., a disgruntled employee) for a successful launch. This mode of attack has become more attractive to attackers especially recently [11].

Malicious USB attacks are usually done to perform mischief such as obtaining passwords, installing malware, or running some dangerous scripts.

A frameup is usually achieved simply by planting false evidence or lying in any way that implicates an innocent person. Another cause of miscarriages of justice is "tunnel vision" [12]. Tunnel vision occurs when there is evidence that proves (or could possibly prove) that an individual or party is not guilty as charged but this evidence is being ignored or suppressed (because of bias and pressure) and the case is built on other evidence presented that already implicates the suspect [12]. After a suspect has been identified during a police investigation, a case is usually then built against this "suspect" instead of gathering more information on the crime [13]. This initiation of suspicion, according to [13] is the first step toward tunnel vision.

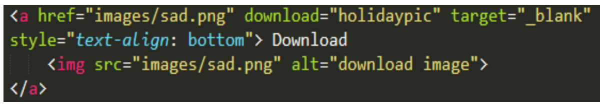
In cases of indecent photographs or child pornography for example, there is a principle that if you cannot show how the indecent photographs of children made their way onto the device, there is the presumption that it was made by a Trojan Horse. This is called the Trojan Horse Defence (THD). The THD is a more recent type of the SODDI (Some Other Dude Did It) defence [14]. The SODDI defence is when a crime happened but the defence tries to push the blame or responsibility on another party or actor, whereas in the THD, the act of crime is blamed on malicious software [14]. A Trojan Horse is a malicious program that presents itself as legitimate or useful in order to gain access to a machine. It is often activated without the user's knowledge and could be used to gain remote access to a machine, log keystrokes, steal data or perform some other actions (depending on what it has been programmed to do). The THD is raised by the accused denying they are responsible for all or part of the evidence presented [15]. The THD claim is not always valid but the burden lies with law enforcement agents to ensure the digital evidence presented has no trojan or malicious codes or any other offenders who could be responsible in any way for the offence [15]. Indeed, a trojan could exist on a machine without the user's knowledge and it has become common to blame a trojan for crimes related to a computer [14].

However, we can show a mechanism for an event happening, these traces cannot be totally or always trusted because of the advent of programable USBs which could be considered the next generation trojan horses. In this work, the possibility of using devices like the programable USB for a framing attack are is considered, to determine how easy it is to plant evidence to frame a user using such a device. Would a digital forensics analyst detect this, what are the obvious signs of devices like this, and can this lead to a miscarriage of justice?

III. Experimental Setup

A website (AirSlide) has been developed using HTML5 and CSS3 and is hosted on a Linux web server running Apache2 which represents an illegal child exploitation site. On this site, are some images representing child pornography. Fig 2 gives a logical view of the testbed and what is running on each (software/programs). The Victim machine is the target machine to be exploited. All are connected via a wireless access point.

To get to the user area where the pictures are, the user logs into the site, proceeds to the login page, enters their credentials and hits the login button which gives them access to the user area where the pictures are located. To download any of the images, the user would need to click on an image of choice to begin the download. This leverages the auto-download feature available in HTML5. HTML5 also allows renaming of a file during download so the code written here also automatically renames the image as “holidaypic.png” on the user’s machine (see Fig. 1).



```
<a href="images/sad.png" download="holidaypic" target="_blank"
style="text-align: bottom"> Download
    <img src="images/sad.png" alt="download image">
</a>
```

Figure 1 Auto-download HTML5 code

There are two scenarios. The first is where a human uses the system to perform these actions and then opens the downloaded image to view it. The second scenario is where the Rubber Ducky performs identical actions, including viewing the downloaded image and then closing it. This exploit is used against the Windows 10 machine with security (firewall/windows defender) in place and active. The Rubber Ducky is plugged into the target machine to execute this process. Having completed this, we forensically analysed both computers to look for any clues left that could differentiate between the Rubber Ducky's actions and the human user's actions. This experiment was repeated five times in order to gather enough data to study and compare the pattern of operation between the Rubber Ducky and the human user.

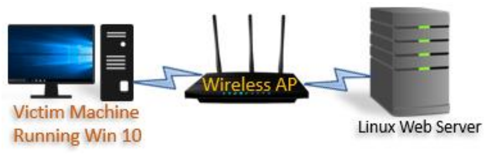


Figure 2 Experiment setup

IV. TOOLS AND TECHNOLOGIES

Web Server: a Linux web Server running Apache2 was used. This web server hosts the web site of interest (Airslide) which also contains the picture of interest.

There is a need for consistency of files and configuration of the machines used for each experiment scenario as this is paramount in achieving the aim of this experiment. Advantages of hosting Airslide on a private local server over existing/popular Internet web pages include:

1- There is control of the server configuration.

2- There is control of the web content.

3- There is knowledge of the code being run.

4- If any of the experiment scripts malfunction, it would affect only the private server on the LAN; thereby not breaching the Computer Misuse Act.

5- Knowledge of any changes to server or file can be easily detected as it is important to ensure consistency across all the experiments.

Bootstrap 3: Opensource front-end CSS framework for developing responsive websites.

Victim machine: this is the computer on which the exploit would be carried out. The Rubber Ducky is plugged in and executes the malicious instructions. A VM Ware virtual machine (VM) is used for this. A VM is used because it is paramount to ensure the same files, settings, configurations, exist on the machines for each scenario. Using a VM makes this much more consistent as it is easier to flash back to the original state before each experiment.

USB Rubber Ducky: Hardware device by Hak5 [16] made for penetration testing. This device works like a USB keyboard, in that it can be readily plugged into a computer, has the capability to execute commands or run scripts, and the system sees it as a HID. This device is cross-platform and can be used to attack any system that supports a USB Keyboard. The rubber ducky can inject keystrokes at an extremely fast rate; however, the speed could also be reduced/controlled, and keystrokes spaced as desired to mimic a human user [17].

Duck Encoder: ducky scripts need to be encoded before they can run, and this is where the Duck Encoder comes in. The Duck Encoder is a program that would be used to convert the ducky script into a cross-platform payload. Every payload needs to be encoded and stored as an “inject.bin” file on the ducky for it to execute. The writing of the script and encoding are done on a computer before being transferred to the Rubber Ducky.

V. EXPERIMENT SCENARIOS

A. General experiment information

The VMs run a Windows 10 operating system (OS) with 4GB RAM and double processor core. Hard disk size is 60GB. The machine is powered up and logged in at the start of each scenario. It is assumed that normal user activity was taking place and that the user stepped away for a few minutes (perhaps for a tea) which is enough time for the Rubber Ducky to plant the evidence without being seen. A malicious co-worker or an innocent looking janitor could have subtly plugged this in while “cleaning”, but modes of delivery is not really the focus here rather the harm that could be done.

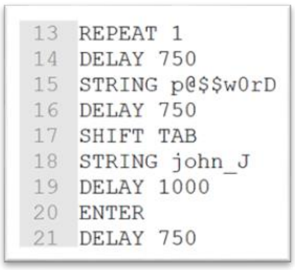
B. Scenario 1 (Ducky Interaction)

In this scenario, the Rubber Ducky attempts to perform actions to mimic a human user by visiting the site, entering credentials and logging in. The Rubber ducky also downloads a picture from the same site, closing the browser and navigating to the downloaded file location to view and closes the viewer afterwards.

Creating the Ducky Script and attack execution

The exploit script was written in a language called ducky script [16]. The script can be written using basic or common text editors such as Notepad++. The written script is then encoded using the duck encoder to make it executable by the Rubber Ducky. This process of encoding generates a binary file (inject.bin) which is transferred to the Rubber Ducky. The Rubber Ducky is then plugged into the target machine. The script has been written to perform the actions described earlier. Fig. 3 below shows a segment of the script where the Rubber Ducky enters the login credentials.

The user database part of this was not implemented because as long as correct credentials (matching username and password) are entered, the login would be successful. That is, if a user (username) exists in the database, and a matching password is entered, the authentication system accepts it as valid. The username entered by the Rubber Ducky in Fig. 3 above is “john_J” and the password entered is “p@$$w0rD”.



```
REPEAT 1
DELAY 750
STRING p@$$w0rD
DELAY 750
SHIFT TAB
STRING john_J
DELAY 1000
ENTER
DELAY 750
```

Figure 3 Ducky script section for entering login credentials

C. Scenario 2 (Human Interaction)

In this scenario, there is an actual user (human) logging on to the site, entering credentials, clicking on one of the pictures to download it to their machine and then viewing the file after download.

VI. EXPERIMENT RESULTS

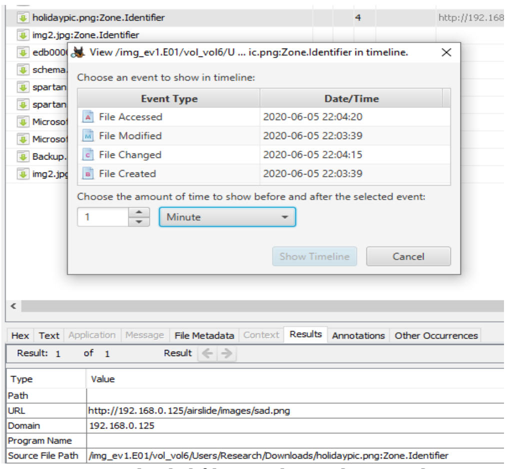
The experiments have been carried out and the results are presented here:

A. Ducky Interaction

The web history shows the user visiting the website, proceeding to the login page with a successful login which leads to member access. One of the files of interest (holidaypic.png) that represented an illegal image was also found under downloads. It is therefore plausible to connect the web history with the downloaded image which appears to have been opened and viewed by the “user” (see Fig. 4) indicating it was of interest to the “user”. The downloaded image metadata also contains the source URL, domain etc. indicating it was downloaded from the AirSlide site (see Fig. 4). The file was checked for any alteration or modification of some sort, but the hash remained same indicating no change.

B. Human Interaction

The evidence shows that the same files have been downloaded and viewed. The web history etc. with same attributes including hash, descriptions, URL, and so on is also present on the victim machine. The most noticeable difference was the timeline of actions (which is expected as they were not accessed, downloaded or viewed at the same time). The URLs, files etc. were found in the same locations in both cases. The hash of the downloaded picture also remains the same. A difference, however, is the creation and modified time.



```
holidaypic.png:Zone.Identifier                    4          http://192.168
img2.jpg:Zone.Identifier
edb000(   View /img_ev1.E01/vol_vol6/U ... ic.png:Zone.Identifier in timeline.
schema.   Choose an event to show in timeline:
spartan     Event Type          Date/Time
spartan     File Accessed       2020-06-05 22:04:20
Microsol    File Modified       2020-06-05 22:03:39
Microsol    File Changed        2020-06-05 22:04:15
Backup.     File Created        2020-06-05 22:03:39
img2.jpc  Choose the amount of time to show before and after the selected event:
          1   Minute
                                         Show Timeline   Cancel

Hex  Text  Application  Message  File Metadata  Context  Results  Annotations  Other Occurrences
Result: 1   of  1        Result

Type              Value
Path
URL               http://192.168.0.125/airslide/images/sad.png
Domain            192.168.0.125
Program Name
Source File Path  /img_ev1.E01/vol_vol6/Users/Research/Downloads/holidaypic.png:Zone.Identifier
```

Figure 4 Downloaded file metadata and accessed time

C. *Investigation overview*

Table I shows a high-level view of the investigation into web activity of interest in both scenario one and two. It can be seen that there are no differences between the actions carried out by the device and those carried out by a user. Nothing to indicate which action was carried out by the Rubber Ducky or which was carried out by the Human. Both visited the site, Airslide, both proceeded to the login page, and both got to the user area where the pictures are.

Table II shows an overview of the files of interest. It can also be seen here that based on the evidence, the same file was downloaded and viewed by the user in both scenarios. There was no change whatsoever or any form of discrepancy to arouse suspicion of the analyst or to push the analyst to dig deeper into the evidence to see if there could be anything to exonerate the user in scenario 1 or to show there is a possibility of them not being guilty as accused. The file hash remained intact indicating no alteration or modification whatsoever. The picture's metadata also contains the source of the picture which corroborates the web history from Table I above and implicates the user again. It appears like the user's actions were deliberate and they had a purpose for visiting the site, saw a picture of interest, downloaded and opened it after download. The implication of this is that the analyst will not be able to tell the difference between the actions carried out by the Rubber Ducky and the ones carried out by a user. This could lead to a faulty conclusion or a miscarriage of justice where the user is charged with "possession of child pornography or possession of indecent photographs".

TABLE I. WEB ACTIVITY OF INTEREST

	Scenario 1- Ducky Interaction	***Scenario 2- Human Interaction***
Site name	AirSlide	AirSlide
Was site visited?	Yes	Yes
Home page visited?	Yes	Yes
Login page visited?	Yes	Yes
User area visited?	Yes	Yes

TABLE II. FILE OF INTEREST

	Scenario 1- Ducky Interaction	***Scenario 1- Human Interaction***
File name	holidaypic.png	holidaypic.png
Downloaded?	Yes	Yes
Viewed by user?	Yes	Yes
Original MD5 hash	5b29dca9cac3b0db218ed6a39bac27c7	5b29dca9cac3b0db218ed6a39bac27c7
MD5 hash after experiment	5b29dca9cac3b0db218ed6a39bac27c7	5b29dca9cac3b0db218ed6a39bac27c7
Any change in file/hash?	No	No
Anything looking abnormal/Unusual?	No	No
Metadata	Indicates download from Airslide	Indicates download from Airslide

The web server access logs were also checked to determine if there were any signs of USB interaction or reasons for suspicion, but no signs or traces were found on the server side. This is not strange because the Rubber Ducky interacts only with the system browser and the browser in turn interacts with the server responding to requests for pages. All the server is concerned about is that there is a GET request which it needs to respond to. This also makes it possible to plant the system information on the server as part of the site visitors without any chances of user repudiation. Example of such information include users' IP (Internet Protocol) address which is often a unique numeric address used to identify any device that is part of a network, the User Agent (usually the browser used by the client), access time etc.

VII. EVALUATION

A live acquisition of the system might make it possible to discover that this device had been plugged in by viewing the RAM content. However, the Rubber Ducky could be programmed to also shut down or restart the computer after performing the attack or at a set time leading to volatile memory being lost. The Rubber Ducky has a default Product ID (PID) and Vendor ID (VID) which is visible when the device plugged in, but these can be changed or swapped as desired by flashing it [18]. Muir [18] also put together some indicators of compromise (IoCs) of the Rubber Ducky (while it is plugged in), but some of these entries in the registry cannot be totally relied upon because they disappear as soon as the Rubber Ducky is removed. Moreover, the Rubber Ducky does not require any unique drivers for installation which could have been used to identify if it has been plugged in. It uses the same driver set as a regular keyboard which is already on the system and in the same location.

The Rubber Ducky executes many instructions via the Windows RUN command (Win + r). Files or URLs accessed via the RUN command usually create a file ending with a '.lnk' extension. During the analysis of our results, the file "http-192.168.0.125-airslide.lnk" appeared across the Ducky Interaction scenarios and was absent in all the Human Interaction scenarios. A confirmation experiment was done on a freshly installed VM to study why this has appeared only across the Ducky Interaction scenarios. Before plugging the Rubber Ducky into the new VM, we visited three web sites (bbc.com, Duckduckgo.com, YouTube.com). The first two were visited using RUN, and a random picture was downloaded from Duckduckgo.com. YouTube.com was then visited via the browser, without using RUN. The machine was then shut down and imaged for investigation. We noticed the sites and downloads that were accessed via RUN had an associated '.lnk' extension present. This includes Duckduckgo.com, bbc.com, and the files downloaded. There was no associated '.lnk' file for YouTube, most likely because YouTube was not accessed via RUN. The presence of this file does not necessarily indicate that a Rubber Ducky carried out the actions, but this could serve as a clue for investigators. A point to note however, is there are no indications on the downloaded file to prove the user did not download the picture or open it and currently no way to trace this back to this device. Comparing the timeline of events can also be unreliable as the Rubber Ducky could have been programmed to do something totally different from the actions the user was accused of which could also lead to a miscarriage of justice if a guilty person was acquitted.

VIII. Conclusion

The possibility of a programmable USB (i.e., the Rubber Ducky) being used in a framing attack was considered. It was shown that planting false evidence in such a way that forensics did not identify this is possible. This makes it difficult to prove innocence when an individual has been framed. Now that there are devices like this, are the possibilities of this also being considered during an investigation? An attacker can use a Rubber Ducky to commit a crime almost anonymously unless the police and experts are aware in the investigative process that this is a possibility.

The Rubber Ducky is a device that can be programmed to do almost anything that a human can on a computer. Are devices like this recognised by law enforcement or even by forensics professionals? Devices like this are an offensive tool used by attackers or penetration testers, not by forensics specialists or law enforcement, which makes it highly likely for evidence to be taken at face value as seen in the results obtained here. This work only considered looking for traces of the Rubber Ducky because there is knowledge of what has been done. If any traces or signs are found later by using some other software or digging deeper, it is because we are sure that we planted such a device and because we expect that there should be some traces (according to Locard's principle - every contact leaves a trace [19]). However, we could see from our results that investigators and experts may not be able to differentiate between the actions of the human user and that of the programmable USB. This leaves room for the possibility of a miscarriage of justice where an innocent user is prosecuted for a crime they did not commit or the guilty is acquitted of the crime they committed.

Research into this is still work in progress. Future investigations will include conducting similar further experiments on machines without active user sessions and using different devices such as the Bash Bunny and the O.MG Cable. Also, using other scenarios such as altering or tampering with existing files, and the theft of intellectual property (IP) to investigate the presence of programmable USB devices.

References

[1] Beitler, A., Jang, J., Kirat, D.H., Kurmus, A., Neugschwandtner, M. and Stoecklin, M.P., International Business Machines Corp, 2018. Protecting computer systems from malicious usb devices via a usb firewall. U.S. Patent Application 15/461,638.

[2] Schneier, B., 2009. Why Framing Your Enemies Is Now Virtually Child's Play | Bruce Schneier. [online] the Guardian.

[3] Owen, M., 2019. iPhone Face ID Not Fooled in Fake Head Test as Android Rivals Fail Appleinsider. [online] AppleInsider.

[4] ISO/IEC 27002:2013(E). (2017). BS EN ISO/IEC 27002:2017, p.17.

[5] Gelernter, N., Grinstein, Y. and Herzberg, A., 2015, December. Cross-site framing attacks. In Proceedings of the 31st Annual Computer Security Applications Conference (pp. 161-170).

[6] Cannoles, B. and Ghafarian, A., 2017. Hacking Experiment by Using USB Rubber Ducky Scripting. Journal of Systemics, 15, pp.66-71. http://www.iiisci.org/journal/CV$/sci/pdfs/ZA340MX17.pdf

[7] Harianto, H.E. and Gunawan, D., 2019. Wi-Fi password stealing program using USB rubber ducky. Telkomnika, 17(2).

[8] Imperva. 2020. Cross Site Request Forgery (CSRF) Attack. [online] Available at: https://www.imperva.com/learn/application-security/csrf-cross-site-request-forgery/

[9] Bojovic, P.D., Basicevic, I., Pilipovic, M., Bojovic, Z. and Bojovic, M., 2019. The rising threat of hardware attacks: USB keyboard attack case study.

[10] Sanzgiri, A. and Dasgupta, D., 2016, April. Classification of insider threat detection techniques. In Proceedings of the 11th annual cyber and information security research conference (pp. 1-4).

[11] Nissim, N., Yahalom, R. and Elovici, Y., 2017. USB-based attacks. Computers & Security, 70, pp.675-688.

[12] Reichart, B., 2016. Tunnel Vision: Causes, Effects, and Mitigation Strategies. Hofstra L. Rev., 45, p.451

[13] Lowrey-Kinberg, B., Senn, S.L., Gould, J. and Hail-Jares, K., 2017. Pathways to Suspicion: Causes and Consequences of Innocent Suspects' Origin of Implication. Cal. WL Rev., 54, p.1.

[14] Bowles, S. and Hernandez-Castro, J., 2015. The first 10 years of the Trojan Horse defence. Computer Fraud & Security, 2015(1), pp.5-13.

[15] Sepec, M., 2012. The Trojan Horse Defence-A Modern Problem of Digital Evidence. Digital Evidence & Elec. Signature L. Rev., 9, p.58.

[16] Hak5. 2021. Ducky Script - the USB Rubber Ducky language. [online] Available at: <https://docs.hak5.org/hc/en-us/articles/360010555153-Ducky-Script-the-USB-Rubber-Ducky-language>

[17] Kitchen, D., 2016. Hak5darren/USB-Rubber-Ducky. [online] GitHub.

[18] Muir, B., 2014. Ducky USB - Indicators of Compromise. [ebook] pp.6, 7

[19] Bode, J., 2019, June. Every Contact Leaves a Trace: A Literary Reality of Locard's Exchange Principle. In Outside the Box: A Multi-Lingual Forum (p. 18)