FLINDERS UNIVERSITY

HONOURS THESIS

Combating Increasing Security Issues through Reduction in Complexity

Author:
Jayden GRIGG

Supervisor:
Dr. Paul GARDNER-STEPHEN

A thesis submitted in fulfillment of the requirements for the degree of Bachelour of Engineering(Electronics)(Honours)

in the

August 22, 2018

Declaration of Authorship

I, Jayden GRIGG, declare that this thesis titled, "Combating Increasing Security Issues through Reduction in Complexity" and the work presented in it are my own. I confirm that:

- This work was done wholly while in candidature for a degree of Bachelour of Engineering(Electronics)(Honours).
- This document is in accordance with the plagiarism policy of Flinders University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Strange how paranoia can link up with reality now and then."

Phillip K. Dick

FLINDERS UNIVERSITY

Abstract

Faculty Name
College of Science and Engineering

Bachelour of Engineering(Electronics)(Honours)

Combating Increasing Security Issues through Reduction in Complexity

by Jayden GRIGG

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

I would like to than Paul Gardner-Stephen for providing me witht eh unique opportunity to work on the MEGA65 project as well as for his support and advice on many of the tasks that I was required to perform. I would also like to thank Benjamin Gerblich for his assistance in understanding the MEGA65 and for being so accommodating during my time working with him. In addition, I would like the thank my fellow students LAchlan McDonald and Tanguy Nodet, who helped me and worked along side me on the MEGA65.

Contents

List of Figures

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Motivation

In the current digital age, many people are becomming increasingly anxious about the security of communication devices.

1.2 Scope

1.2.1 Problem Statement

Everyone has personal data that they wish to keep from prying eyes. Whether the secrecy of this information is important at a national or economic level, such as banking details, protecting ones location from abusive ex-partners, or simple secrets, such as planning a suprise party, it doesnt matter. This information should be able to be stored on a computer, or in a computer-containing mobile device, without fear of its disclosure. In the current world this is not the case; more and more exploits in security are being found every day and more and more patches are being released to count them. The result is systems that are always vulnerable. Worse, the complexity of modern computer systems means that there is no way of verifying that data has not been compromised, or that data is truely safe from compromise. So instead of continuing to add more security features, that may bring with them undesireable side-effects, and are unlikely to be completely effective in any case, focus on the creation of simple systems, simple enough that a determined use can verify their own security. This would allow for greater security that is unable to be provided by any current device.

1.2.2 Objectives

Within the overall objective of creating a simple, secure smart-phone like device, it is hope that the following will be accomplished in this project:

- Secure inter-process communication is established.
- The sub-systems of the phone are evaluted for their readiness.
- The unready sub-systems are completed.
- The architecture as a whole is evaluated at a high-level for security and completeness, ready for being handed over to the next phase of development.

1.2.3 Research Questions

What are the missing or non-functional sub-systems that the MEGA65 requires to be implemented?
Which will be answered in the form of a survey of the current sub-systems of the MEGA65, and a survey of the sub-systems required to create a functional prototype.
How can these sub-systems be implemented?
Which will be answered by the creation of plans for the implementing of the missing or incomplete sub-systems.
How can the simplicity, understandability and security of these sub-systems be maximized?
Which will be answered by considering each sub-system, qualitatively appraising its simplicity, understandability and security, and where appropriate, making recommendations for refining those components to improve one or more of these axes, and time permitting, acting on those recommendations.
How can the complete MEGA65 architecture be physically prototyped on the bench?
Which will be answered by examination of the current partial bench prototype and comparing it with the sub-systems identified through the other research questions, and designing and realising a complete bench prototype. This will occur in coordination with Mr. Lachlan McDonald, who is designing the PCB for the MEGA65 smart-phone device.
How can the secure compartmentlisation architecture planned for the MEGA65 be realised?
Which will be answered by considering this architecture and current state of the MEGA65 system to create and follow a plan to implement the secure compartmentalisation architecture.
Overall the succes of the project will be measured against the creation of a functioning bench prototype device that implements the secure compartmentlisation architecture to the point of demonstrabilirty.

Chapter 2

Literature Review

Since the introduction of general-purpose mobile operating systems such as Symbian, Android, Windows 10 and iOS, and especially throughout the last decade, mobile phones have evolved dramatically.[1] The introduction of feature-rich and complex operating system has not only brought the benefits of a computer, but also the risks of one too.[17] Since phones have become computerized people have become more trusting of what data they can store on their phone, such as location, bank details, etc.[4] With private data increasingly being stored on phones, it becomes necessary for mobile security to receive a larger amount of attention. This literature review explores this issue, as follows: Section 2.1 explores the interrelationship between complexity and security, Section 2.2 documents a wide variety of the security threats facing modern computers, with a particular emphasis on mobile devices, Section 2.3 relates the application of isolation and compartmentalisation to improve security, and Section 2.4 documents the history and status of the MEGA65 project, including the historical Commodore 64 platform on which it is based, as well as the recent work towards introducing compartmentalisation and related security features to the platform.

2.1 Complexity and Security

One of the major issues with the current state of security is that the complexity involved is far beyond what any one person can comprehend. Intel embodies the complexity issue perfectly when examining the number of employees vs the number of transistors on one of their CPUs.
From 1972 to 2000 Intel's workforce grew by 86 times, comparing this growth with the transistor growth of 1200 times over the same period, a huge disparity can be seen. If every employee at Intel was dedicated to testing transistors, in 1972 each employee needed to test 3.5 transistors, compare this to the 487 they need to test now. This clearly demonstrates the impossibility for, not only one person, but for all of Intel to successfully verify that their CPU is working as intended. With modern computers unable to be understood by a single person, they are inherently not to be trusted.[6] Not only are they not to be trusted, but they can never be trusted, because while they are not understood by a single person there is always the possibility of interactions in the CPU that remain undiscovered.[7]
As systems become more complex, they too see more issues arise within them. A key property of this is the amount of failure modes that are present in the system. Larger, more complex, systems naturally have more security vulnerabilities as they have more interactions, or opportunities for failure.[21] These larger systems also require more time and effort to test. More worryingly is that for the end user, this complexity aids malicious attempts.[10] For the same reasons that testing is difficult, so too is finding indicators of a security breach, or even malicious activity after a

breach has been detected.[23]
To combat this increase in security flaws there are a few methods that can be used. One such method is to patch the security flaws with work arounds. As these work arounds were not originally designed as part of the system, there is a possibility that they can make the entire system less secure overall.
This has been beautifully demonstrated by the recent exploits found in speculative execution processors. This exploit, named Meltdown, was a massive cause of panic as it allowed privileged information to be read at megabytes a second. The Windows 7 January 2018 patch for the exploit resulted in permissions for the user being erroneously set. This allowed the user to read memory that was normally only accessible by the processor.[5]
The other method of reducing security flaws is to reduce the complexity of the system. Less interactions allows more time dedicated to ensuring each one works correctly. In addition to this, simpler systems allow for users to better understand them, and thus are more easily verified as secure. This was listed as one of the key aspects of a trusted computer system by the Department of Defence as it is only then that understandable and maintainable protection is achieved.[3]

2.2 Mobile Security Threats

When analysing current mobile (smart) phones there are multiple avenues under which data can be maliciously obtained, or normal service can be disrupted. These avenues can be categorised under the following labels:

- Mobile Hardware
- Mobile Software
- Mobile Networks
- Physical Access
- Mobile Enterprise

2.2.1 Mobile Hardware

Mobile hardware refers to the various sensors and physical devices built into the phone. By accessing the hardware directly on the phone, most software security provided by the operating system is bypassed, allowing malicious users to access data provided by cameras, microphones, etc. This form of attack is particularly difficult to protect against in some cases due to the advent of rooting or jail-breaking.[13]
Jail-breaking/rooting is where the user deliberately exploits vulnerabilities to gain more control over the phone's systems.
Currently, the best way to defend against mobile hardware exploitation is to ensure that the operating system is up to date and that patches for publicly known exploits are installed. This does not prevent mobile hardware exploits but rather makes them nontrivial to perform.[9]
Even with all the updates and patches installed, there still exists improvements that could be made to further reduce hardware exploits. As most hardware on a phone does not have an indicator for when it is active, it is difficult to determine its status; implementing this in hardware would allow a user to tell decisively whether hardware was on or off.[11]

2.2.2 Mobile Software

Mobile software refers to the applications running on the phone. While these applications give the user more control over their phone, they are also a potential source for coding errors.
Through these errors such as, files being stored in an unprotected location or files being stored in an insecure format, it is possible to leak sensitive data.[2] Not only is it possible to read stored data, but through an insecure network or a vulnerable third-party software library, it is possible to read and alter all data being sent to and from the phone.
To protect against vulnerable software there are limited options available. Keeping the operating system up to date is the best solution as more security monitoring helps catch exploits.[9] The process of application analysis involves two different methods, static analysis and dynamic analysis. In static analysis the source code for an application is evaluated without running it to detect possible exploits. This method relies on the source code being available, which isn't always the case, and, if it is computer assisted, can provide false positives.[24] Static analysis can be performed to a lesser extent on compiled code, but it is limited to program interface, compiler optimisations and software library checking. Dynamic analysis is like source code static analysis with the exception that it is done with the code running. This allows behaviour that is not apparent from source code examination to be identified.[11]
In addition to coding errors allowing third parties to exploit security vulnerabilities, some applications are developed specifically for malicious activity. These applications often exploit underlying vulnerabilities in the operating system or trick the user into giving the application privileges that exceed those needed for proper function.[26]
Malicious applications can breach phone security in many ways to extort money from the user or gain access to protected data. Ransomware is the name given to an application that encrypts all the user's data and demands payment to decrypt it. This form of extortion poses extreme risk when dealing with cooperate data due to the loss in productivity. The unauthorised access of user data can be done through many methods such as jail-breaking the phone so that hardware can be exploited, subverting login details, sharing data with other applications (such as Dropbox) and logging private data.[11]
Malicious applications are difficult to protect against. In almost all situations they require some form of active security that is user reliant to effectively protect against them. In addition to understanding the threats present and keeping all software up to date, users can:

- Use dynamic analysis in a safe environment
- Isolate the application from sensitive data
- Use authentication protocols

These methods would allow for adequate protection against malicious applications or applications used for a malicious purpose, but there is still more that can be done. Better application development practices and regulations would prevent many of the issues with software exploitation. This is unrealistic though as most applications are so complex there will always be a missed bug. More transparent applications are the solution to this as it would allow users to not only verify the application but improve upon it.[11]

2.2.3 Mobile Networks

A mobile network is the interconnection between all phones that allows for phone calls, messages and internet access to the mobile device. These connections are facilitated by the numerous radios and modem. The mobile network can be broken down into three major sections, the radio access network, the core network and the external services network.
The radio access network facilitates the connection between the mobile device and the service provider for the phone. This network is used to connect the mobile device to the telephone tower which then leads to the core network. The core network is responsible for tracking billing, signal routing and connecting the mobile device to the external services network, such as the internet or mobile devices from a different carrier.[11]
Radio access networks are susceptible to three basic types of exploits, denial of service attacks, eavesdropping and device tracking. Denial of service attacks is used as a large umbrella term for all the attacks that can block or hinder normal mobile phone operation. These attacks can be conducted by filling up all available radio access slots on a telephone tower, impersonating a telephone tower to intercept emergency calls or flooding the area with radio waves to obscure legitimate calls.[20]
Eavesdropping, as the name suggests, involves gaining access to data being transmitted between the telephone tower and mobile device. This is possible because it is not required that data between the mobile device and the telephone tower be encrypted, in the case that end to end encryption is not used, all the user's data is free to access by those that can.[11] For superior data transmission a shorter radio access network is desired, this bring forth a need for mobile device locational services, so the closest radio tower can be used in data transfer. This can be exploited to track a mobile device and by extension, a user's location.[20]
The core network, once subverted, offers attackers many avenues of attack. Most mobile networks use some sort of management system that controls the entire network. This provides the ability to intercept or block phone calls and texts, as well as denying service to users were it subverted.[11]
External networks are how a phone connects to the internet among other networks. This brings all the benefits, but also all the dangers of an internet connection.
To defend against data interception and theft, end to end encryption must be used. This negates any chance of the radio access network and core network leaking data, as the leaked data would still be encrypted. Denial of service attacks can be mitigated by alternative methods of data transmission. Though it is highly unlikely that an entire carrier will have it's services denied, using an alternative method would negate any and all effects of such an attack. Unfortunately, mobile device tracking cannot be defended against due to it being so intertwined with the mobile network. Even bouncing a connection across a privately-owned network would not result in much, as it can be traced by a skilled attacker.[11]

2.2.4 Physical Access

While mobile phones are almost always on person all the time, there are certain situations in which contact must be broken. These instances occur during certain security checks, like police or airport searches, or during charging and communication sensitive operations. During these times it would be possible to access data from the phone or perform other malicious activity.
One avenue of attack is the combined charging/data port of the phone. By plugging

the phone into a PC, it is possible to abuse vulnerabilities to gain access to data. This is not limited to PCs however, by modifying a charger with a small PC board it is possible to execute complex attacks on the device whenever the phone is charging. If done right, this attack could also work in the reverse, infecting a host computer that the phone is connected to.[11]
Another avenue is the phone itself, NowSecure released a report in 2016 that stated 43% of mobile users do not use a pass-code, PIN, or pattern lock on their device.[18] This, coupled with cached memory and browser cookies could result in banking details, among other things being stolen.
The best defence against physical access attacks is to use screen locks and use personal charging equipment. These negate the threat of physical access attacks and data port attacks.
Despite the ease of ensuring physical security, more needs to be done to encourage users to use this security. Currently, most phones use authentication methods that don't fully capitalise on all the mobile sensors available. While using sensors like fingerprint scanners and facial recognition is emerging in mobile phones, there exists sensors like the capacitive sensor, gyroscope and accelerometer that are still unused in lock screen technology.[[11]

2.2.5 Mobile Enterprise

When mobile devices are integrated into business there comes a need for servers, processes and systems that manage these devices. This enterprise, such as a company application, can serve as an infection source, should the enterprise itself become infected. To counteract these flaws mobile device managers for enterprise purposes have been developed. These managers enforce security policies, remote access, remote wiping, etc. for the device.
Because these device managers have a higher level of privilege than the user, exploiting the device manager to gain access to the entire enterprise is a serious threat, as doing so would allow infection of all devices connected to the enterprise. Once compromised, it is possible for all data within the enterprise to be stolen, manipulated or for all services to be blocked completely.
These attacks can be mitigated using comprehensive authentication protocols to prevent impersonation, protected execution environments and network monitoring. While authentication prevents most forms of attack, it is still subject to authentication information being discovered, which is why the execution environment is used to contain malicious activity and network monitoring is used to look for such activity.[11]

2.3 Security Through Isolation

One approach to securing a computer has been to have sections of it "air gapped", such that infection in one area cannot affect other areas.[15]
One example of an operating system built on this principle is the Qubes OS, which has gone about this through compartmentalisation. By isolating things, such as untrusted websites, in their own qube it is possible stop any attempt to compromise the whole system as they are contained in that qube. These qubes are not limited to software, they work with hardware too, meaning that USB controllers and network cards are secured in their own qubes. This reduces the effectiveness of using hardware as a security exploit.[19]
Isolative computing can be emulated by hosting multiple virtual machines on one

host machine, thus recreating the Qubes OS concept. This comes one major disadvantage over using Qubes OS however, once the host OS is subverted all the virtual machines are subverted too.[8] As Qubes OS uses a hyper-visor to manage all the qubes it is inherently more secure than virtual machine recreation. This is due to the increased difficulty of subverting the hyper-visor as compared to subverting an operating system.
Another method for air gapping a computer is to have it physically isolated from external connections. By removing the computer from external networks almost all attack vectors are removed too, thus ensuring the computer is safe. This form of protection is mainly used in critical systems, as by isolating a computer a lot of the functionality of it, such as web browsing, is removed.[27]
While air gapping computers appears to make data exfiltration impossible, there are a few ways in which this can happen. The simplest method is via physical access and a USB or USB device. The best know device is the USB Rubber Ducky, this device can be used to imitate a keyboard inputs via a script loaded onto the device.[22] This allows the rubber ducky full control over the host device and can even be used to install malicious software.[22] Another method of circumventing an air gap is to use high frequency sound from a speaker to transmit and infect air gapped computers[25] "AirHopper" is another method for data exfiltration, instead of using sound, it uses radio waves generated by cables from the computer.[12] Even when Faraday shielded to prevent these sound and electromagnetic signals from escaping, it is still possible to transmit data from the computer by using the CPU. The CPU can be used to generate low frequency magnetic waves by altering the load on specific cores. These magnetic waves more readily penetrate Faraday shielding, allowing the transmission of data from the air gapped computer.[14]

2.4 The MEGA65 Project

The MEGA65 project was born from one Paul Gardner-Stephen in 2014. The scope of the project was to create an accelerator for the commodore 64. This changed when the Museum of Electronic Games and Art sponsored the project and connected him with various other experts.[16] The project was expanded to recreating and archiving the commodore 65, a prototype computer that was never officially released. In addition to the restoration of this computer, updating the hardware was done wherever possible, so long as they didn't compromise the identity of the computer.[16] This was done so an external accelerator was not needed for the project.
The MEGA65 computer itself runs on a field programmable gate array (FPGA) dev board. This option was explored in depth due to a few factors such as, cost, onboard SD card slot, VGA output and the Artix-7 chip. The cost of boards was a large factor in the decision to use them as they would make the C65 affordable. The SD card slot on the board also helped this affordability as they are inexpensive and, conveniently, able to easily be interfaced with many computers. The VGA interface on the board was considered noteworthy as many monitors still have a port for them and there are readily available adaptors for VGA to various other ports.[16]
In addition to the MEGA65 project, another project was run along side it, the MEGAphone project. This project aimed to create a commodore 65 based smart phone with a security focus. Instead of allowing the technology gap between the MEGAphone and modern phones to be detrimental, it would be one of the key features of the phone. The vastly simplified way in which the computer worked as well as the age would allow for many advantages over modern systems.

The simplicity would allow for the entire device to be verifiable, this is not normally explored by other security development teams. By being able to verify the phone is working as intended, it becomes almost impossible to compromise. In addition to this, as the phone is security focused, malicious activity is limited by the phone design itself. Not only this, but many of the security flaws being found in systems today are just not present in the commodore 64 because it doesn't have the supporting hardware.
Through the 36 years of activity that the commodore 64 has seen, many bugs have been found and documented in the system. As such, it is monumentally unlikely that there are new exploits in the system that are unfound, making it excellent to use for security purposes.

Chapter 3

Matrix Mode Corrections

3.1 Serial Locking

While experimenting with matrix mode it was noted that entering matrix mode then leaving matrix mode resulted in the serial input being unable to supply characters to the MEGA65. This phenomenon was especially strange as the code to disable the serial input while in matrix mode was commented out, meaning there was no way for the bug to be caused by entering matrix mode.

3.1.1 Creating the Test-bench

In order to save time during the synthesising of the phone, a CPUless version of the phone was created. This version was outdated and required some additional ports to spoof the components into working correctly. Once the components were working, a tcl file was acquired from one of the other contributors to the project. This file was used to set up and execute the vivado tasks required to create and compile the CPUless version of the phone. This resulted in a bit stream that was able to be run on the Nexys4ddr boards in the lab.

3.1.2 Timing Issues

One of the theorised causes for the serial locking issue was that the timing between the handshaking protocols was misaligned and was causing a state in which the serial output was not able to be read. In order to find these issues a timing report needed to be generated during synthesis. This was achieved by altering the tcl file and enabling the timing report generation. In the timing report several video signals were found to be causing timing issues; to solve these an intermediary signal was created to host the video data. This would cause the data to be delayed by one clock cycle, this was determined to be acceptable however as the difference would not be perceptible to the human eye. Some of the signals were causing timing issues because they were in if statements that were too deep. This caused values in the outter statements to become stale before the inner logic could be resolved. In order to compensate for this the if statements were reduced in depth, this was done by increasing the arguments in the outter if statements. Unfortunetly, the resolution of the timing issues had no impact on the serial locking, but it did result in faster synthesis and a clearer image on the screen.

3.1.3 Handshaking Issues

As the timing issues were cleared, it was determined that the locking issue must be an error in the handshaking rather than a signal error. Firstly the keycode data that

was passed from the terminal emulator to the rain compositor was examined. This revealed that the correct data was being passed even after the lock had occurred. After examining some of the other various signals it was noted that the terminal emulator sent flag was not returning to zero. This was the result of the terminal emulator ready flag never being raised in the matrix rain compositor. This was due to a failure to handshake the monitor char valid between the uart monitor and rain compositor. The uart monitor was not communicating after it had sent the character that what it was sending was no longer valid. This in turn did not allow the rain compositor to prime for another character and caused it to never enter the ready status. With ready never being reached, the

3.2

Bibliography

[1] D. Adams. *The History of Mobile Operating Systems [Infographic]*. July 2011. URL: http://www.bitrebels.com/technology/the-history-of-mobile-operating-systems-infographic/. (accessed: 31.05.2018).

[2] L. Constnatin. *Attackers exploited ColdFusion vulnerability to install Microsoft IIS malware*. Dec. 2013. URL: https://www.pcworld.com/article/2080721/attackers-exploited-coldfusion-vulnerability-to-install-microsoft-iis-malware.html. (accessed: 31.05.2018).

[3] The Department of Defense. *Trusted Computer System Evaluation Criteria*. Tech. rep. The Department of Defense, 1985.

[4] B. Ellis. *Your most dangerous possession? Your smartphone*. Jan. 2011. URL: https://money.cnn.com/2011/01/12/pf/saving/smartphone_dangers/index.htm. (accessed: 31.05.2018).

[5] U. Frisk. *Total Meltdown?* Mar. 2018. URL: http://blog.frizk.net/2018/03/total-meltdown.html?m=1. (accessed: 26.05.2018).

[6] B. Gerblich. "Computers Must be Understandable to be Secure". MA thesis. Adelaide: Flinders University, 2017.

[7] D. Goodin. *Meet "badBIOS," the mysterious Mac and PC malware that jumps airgaps*. Jan. 2013. URL: https://arstechnica.com/information-technology/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/. (accessed: 05.06.2018).

[8] D. Goodin. *Virtual machine escape fetches $105000 at Pwn2Own hacking contest*. Mar. 2017. URL: https://arstechnica.com/information-technology/2017/03/hack-that-escapes-vm-by-exploiting-edge-browser-fetches-105000-at-pwn2own/. (accessed: 5.06.2018).

[9] D. Graham-Smith. *12 ways to hack-proof your smartphone*. Mar. 2017. URL: https://www.theguardian.com/technology/2017/mar/26/12-ways-to-hack-proof-your-smartphone-privacy-data-thieves. (accessed: 31.05.2018).

[10] P. Kervalishvili J. Ramsden. "Security and Complexity". In: IOS Press, 2008, pp. 249–347.

[11] Dr. R. Griffin Jr. *Study on Mobile Device Security*. Tech. rep. Washington: The Department of Homeland Security, Apr. 2017.

[12] E. Kovacs. *"AirHopper" Malware Uses Radio Signals to Steal Data from Isolated Computers*. Oct. 2014. URL: https://www.securityweek.com/airhopper-malware-uses-radio-signals-steal-data-isolated-computers. (accessed: 3.06.2018).

[13] S. Lemon. *Jailbroken IPhones Leave Users More Vulnerable*. July 2009. URL: https://www.pcworld.com/article/167760/article.html. (accessed: 31.05.2018).

[14] et al. M. Guri. *ODINI Escaping Sensitive Data from Faraday-Caged, Air-Gapped Computers via Magnetic Fields*. Tech. rep. The University of Negev, Feb. 2018.

[15] S. Mansfield-Devine. "Security Through Isolation". In: *Computer Fraud and Security* (May 2010), pp. 8–12.

[16] *MEGA65 Design Choices*. Mar. 2018. P. Gardner-Stephen, [Interviewee].

[17] J. Murphy. *Security Minister Warns of Mobile Phone Hacker Risk*. June 2009. URL: http://go.galegroup.com/ps/i.do?id=GALE%5C%7CA203154593&v=2.1&u=flinders&it=r&p=ITOF&sw=w. (accessed: 31.05.2018).

[18] NowSecure. *2016 NowSecure Mobile Security Report*. Tech. rep. NowSecure, 2016.

[19] Qubes OS. *An Introduction to Qubes OS*. Aug. 2017. URL: https://www.qubes-os.org/intro/. (accessed: 23.03.2018).

[20] et al. S. Kami Makki. *Mobile and Wireless Network Security and Privacy*. Springer, 2007.

[21] *Security and Complexity*. IT NOW, Dec. 2015. R. Anderson, [Interviewee].

[22] B. Slash. *Use the USB Rubber Ducky to Disable Antivirus Software and Install Ransomeware*. Nov. 2017. URL: https://null-byte.wonderhowto.com/how-to/use-usb-rubber-ducky-disable-antivirus-software-install-ransomware-0180418/. (accessed: 3.06.2018).

[23] McCabe Software. *More Complex Equals Less Secure*. Tech. rep. 41 Sharpe Drive, Cranston, RL 02920: McCabe Software, 2015.

[24] McCabe Software. *Software Security Analysis: Control Flow Security Analysis with McCabe IQ*. Tech. rep. 41 Sharpe Drive, Cranston, RL 02920: McCabe Software, 2015.

[25] K. Than. *Computers can be hacked using high-frequency sound*. Dec. 2013. URL: https://www.insidescience.org/news/computers-can-be-hacked-using-high-frequency-sound. (accessed: 3.06.2018).

[26] K. Thompson. "Reflections on Trusting Trust". In: *Communications of the ACM* 27.8 (Aug. 1984), pp. 761–763.

[27] K. Zetter. *Hacker Lexicon: What is an air gap?* Dec. 2014. URL: https://www.wired.com/2014/12/hacker-lexicon-air-gap/. (accessed: 2.06.2018).