**Security Report**

How to reduce software vulnerabilities

Luke Bruni

2018

**Table of Contents**

[**1. Introduction** 2](#_Toc511729179)

[**1.1** **Software Development** 2](#_Toc511729180)

[**1.1.1** **Software Development Process** 2](#_Toc511729181)

[**1.2** **Vulnerabilities** 4](#_Toc511729182)

[**1.2.1** **Types of Vulnerabilities [7]** 4](#_Toc511729183)

[**2. Case Studies** 5](#_Toc511729184)

[**2.1 Yahoo 2013-2014 Data Breach** 5](#_Toc511729185)

[**2.2 NHS Cyber-Attack** 5](#_Toc511729186)

[**3.** **Mitigating Software Vulnerabilities** 6](#_Toc511729187)

[**4. Primary Research** 6](#_Toc511729188)

[**4.1 Introduction** 6](#_Toc511729189)

[**4.2 How well do people know software vulnerabilities and vulnerabilities in general?** 7](#_Toc511729190)

[**4.3 What can a programmer do to prevent and reduce software vulnerabilities?** 8](#_Toc511729191)

[**4.4 How do people apply security in their personal lives?** 9](#_Toc511729192)

[**5.** **Conclusion** 13](#_Toc511729193)

[**5.1 Primary Research Conclusion** 13](#_Toc511729194)

[**5.1.1 Recommendations** 14](#_Toc511729195)

[**5.2 Secondary Research Conclusion** 14](#_Toc511729196)

[References 15](#_Toc511729197)

# **1. Introduction**

## **Software Development**

Software development is the process of creating applications or software which is made using a specific programming language. This is an iterative process used to create these programs, to fulfil either a business or personal goal, process or objective. The process is usually made up of various steps within development to create the operational software in question.

The method of creating software is commonly referred to as a Software Development Life Cycle (SDLC) and the process is normally carried out through the software programmer, with initial research, data and process flow design, flow charts, technical documentation, testing and debugging. The process is commonly iterative, but there are other methodologies to adopt depending on the nature of the program [1].

### **Software Development Process**

All Software Development Life Cycle models have phases which are in the order that they are executed in. Each phase marks a point for the deliverables which are required by the next phase. Every Lifecycle model has these six development phases:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance [2]

#### Requirement Gathering and Analysis

This phase is where the requirements are collected and understood and is the main focus of the project managers and stake holders. Meetings are held in order to determine the requirements, with questions considered in relation to the software’s purpose, such as who is this system built for? Who will use the system? How will they use it? Etc. where then these requirements are analysed for the purpose of validation and use within development.

It is relatively important to gain these requirements because it sets the development of the project in stone. Each member of the team has a solid understanding of the product in their minds, so they can build of what they are given, supporting the next phases of development. To perfectly understand what the requirements are, it should be important to keep them clear and concise, that way a good understanding is used and the requirements can support the development [3].

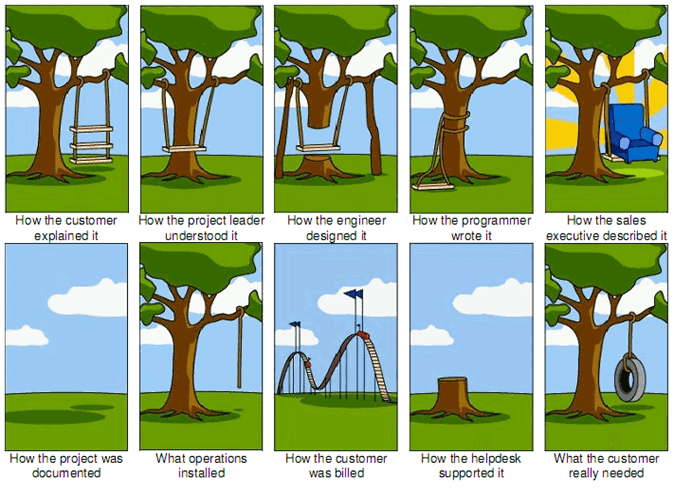


Figure : Tree Swing Project Management Comic [4]

The requirements to a project are heavily based on how they are perceived, as mentioned previously. Should the team not understand the requirements, then they won’t prove beneficial to the development and would more often result in the project being designed without the requirements in mind, confusion amidst each team member and ultimately leading to a product that the client didn’t ask for. The figure above showcases a simplified view of mishandled or misunderstood requirements.

#### Design

The system design is then prepared based on the requirements in the previous phase. The design helps in specifying the hardware and defining the overall system architecture, setting in place the next stage of development.

It is also worth noting that testers involved come up with a test strategy, detailing what to test and how to test it.

#### Implementation or Coding

When the design documents are completed, the work is then split up into units and modules and the actual coding starts. This phase is the main focus as it is where the features are implemented and the code is produced and it is also worth noting that this is the longest phase out of the entire development lifecycle.

#### Testing

After the code is developed, the initial product is tested against the requirements to make sure that the product meets the client’s needs and serves its purpose as stated within the first phase. All types of testing are carried out, such as unit testing, integration testing, system testing and acceptance testing as well as non-functional.

#### Deployment

This phase comes after successful testing, the product is then handed over to the product to see if said product fulfils the goals that were asked. As soon as it is deployed, the first beta testing will commence, where any changes requested are noted and bugs are discovered. The outcome of the beta testing will serve as a basis for the final release.

#### Maintenance

When the customer uses the system, problems will arise from time to time and would need to be solved. Care is taken for the developed system to ensure it still operates perfectly without any faults.

## **Vulnerabilities**

Regarding computing security, vulnerabilities are weaknesses within software, being a flaw within code or design that creates an opening for an attacker to breach security and either run code or access a system’s memory. Because these vulnerabilities are discovered, they are then exploited, allowing to be conducted through hacking scripts, applications or even free hand coding [5]. Once the attacker has breached access, they have the opportunity to gain access to information and can exploit the vulnerability to hide their actions [6].

### **Types of Vulnerabilities [7]**

This section describes each vulnerability that could occur in software.

#### Buffer Overflow

A buffer overflow is where an application attempts to write data past the end of a buffer. This can cause the program to crash, compromise data and provide an attack vector to compromise the system.

Applications that take input from the user, from a file or a network has to store that input. This storage is temporary, but there are two exceptions to where it can be stored, in either the stack or the heap. Buffer Overflow attacks occur by compromising either one storage or both.

#### 1.2.1.2 Unvalidated Input

This is unwanted input that can occur by having the attacker interfere and pass in abnormal data. Once the program crashes, then the attacker looks for exploits in the system and subsequently take control of the system, steal data, corrupt disk, etc.

Any form of input from an untrusted source is a more than likely target for an attack, such as:

* Text input fields
* Commands passed through a URL used to launch the program
* Files provided by users or other processes and read by the program
* Command line input
* Any data read from an untrusted server over a network
* Any untrusted data read from a trusted server over a network.

#### 1.2.1.3 Race Conditions

Race conditions are made when changes in events cause a behavioural change, to which the attacker can take advantage of the situation to either insert their code, change the name of a file or just interfere with the program’s operation. However, this may not always be the case, as if the correct order of execution is required for the operation of the program, then the change is a bug.

#### 1.2.1.4 Interprocess Communication

This is a mechanism that allows for data exchange between processes [8]. The methods involved include shared memory or a messaging protocol such as sockets. These messaging protocols are vulnerable as the end communication channel could be hostile, so programmers have to always assume that they are hostile.

#### 1.2.1.5 Insecure File Operations

An attacker can change the permissions of a file after creation, if not checked properly. This is the result creating insecure temporary files and can result in tampered files that the owner wouldn’t notice [9].

# **2. Case Studies**

## **2.1 Yahoo 2013-2014 Data Breach**

Recently, Yahoo had said that all 3 billion of its accounts were hacked in a data theft that had occurred in a disclosure during the company being sold to Verizon. It had been the biggest breach in history, tripling its earlier estimate [10].

An investigation was carried out in 2013 saying that 1 billion accounts were compromised, but went unnoticed. Yahoo did blame the breach on an “Unauthorized third party”, though most details weren’t provided until law enforcement had reported that the files were originally from yahoo but a third party had managed to obtain the files. The company had then verified that the stolen data was legitimate shortly after.

The data that was affected was mainly user data; names, emails, phone numbers, dates of birth and hashed passwords were obtained, mainly because they were protected with outdated encryption which was considered easy-to-crack and security questions along with backup email addresses, to which it was easy to break into the other accounts that were held by the users. However, the investigation had specified that the stolen information did not include payment card data or bank account information [11].

Another investigation was carried out, reporting another massive breach which affected 500 million in 2014 but botched its response. The security team at Yahoo had knew about what Yahoo report as a ‘state-sponsored hacker’ had stolen backup copies of backup files, containing personal details on users. The reason it was finally brought to light, two years after the breach, was because of a stolen database purportedly went up sale on the black market [12].

It wasn’t until much recently that additional information had been obtained by Yahoo, where it showed that all user accounts had been affected, the 3 billion figure including accounts that were opened but never or briefly used. In response, Yahoo emailed notifications to additional users that were affected.

The following breaches had affected a deal with Verizon where following the newer information about the first breach, lowering it by $350 million from Verizon’s original offer to buy the company and brought about 43 class action lawsuits against Yahoo. The original deal had closed in July 2017.

## **2.2 NHS Cyber-Attack**

NHS trusts were hit by a huge ransomware attack in May 2017, where more than a third of the trusts throughout England and Scotland were disrupted by the ransomware. As a result of this, at least 6,900 NHS appointments were cancelled but no data relating to patients was compromised. There was no evidence that any NHS organisation paid the ransom but the cost of the incident remained unknown [13].

The Ransomware in question was called ‘WannaCry’. A file encryption software, that encrypts the PCs files, that prevents user access to the PC and demands a payment to be made, in bitcoin, to decrypt them, usually demanding $300 in bitcoin. The Vulnerability that WannaCry exploits is in the Windows implementation of the Server Message Block (SMB) Protocol, which helps nodes on a network communicate [14].

The issue was that out of 236 trusts, 88 had failed the required cyber-security standards and a lack of action on critical alerts from NHS digital. Computers at the NHS were running the old Windows XP operating system, which were vulnerable due to the stopped support on 8 April 2014 [15]. A patch had been brought out but it was also discovered that also the Windows 7 OS was vulnerable, but that operating system was still supported. A plan was developed to help counteract such attacks, it wasn’t seen through properly, with poor management, lack of testing and the fact that it was not communicated properly.

# **Mitigating Software Vulnerabilities**

It is always beneficial to tackle Software Vulnerabilities, as often, the vulnerabilities mentioned earlier can have an adverse impact on the security of the app or system. An attacker can exploit the system in many ways, which in turn leads to different purposes, such as data stealing or to damage a mainframe.

Modelling out the vulnerabilities is a good first approach to gain an understanding of what vulnerabilities can come out of development. By using the model as reference, it helps towards more careful development and can give the developer an idea on how to counteract said vulnerabilities. However, it would be considered necessary to count on methods to prevent the risks related to vulnerabilities.

Software Inspection is a method of vulnerability mitigation, in which its process is to read or visually inspect the program code or documents in order to find any defects and correct them early in the development process. This can help with the development as if they are found later, it becomes costlier to fix. Good inspection depends on the ability and expertise of the inspect, along with the defects in what he’s looking for [16].

Well embedded and upheld user access controls will restrict the applications, privileges and data that users can access, making for tighter controls to prevent sudden unauthorised access. Secure configuration can remove unnecessary software and default user accounts, making sure that passwords are changed and automatic features that could activate any malware to be turned off.

Monitoring software activity and analysing it to identify malicious or unusual activity, is a good call to help recognise a breach [17].

# **4. Primary Research**

## **4.1 Introduction**

In the wake of security breaches, it is revealed that likelihood of an attack is very high and the effects can be devastating to that of a company. To aid in the report, I have devised a survey to gather information on the awareness of vulnerabilities, knowledge and prevention of vulnerabilities and personal security, mainly to understand people’s stance and knowledge to security; how important it is, if they do understand the implications of the lack of security and how they can reduce the vulnerabilities that rise the chance of being attacked.

The survey was divided into three parts: Awareness, Software Security and Personal Security. This was mainly done to facilitate the survey and the question structure for this report. The first part is more of a mixture between quantitative and qualitative questions, mainly to back up what the user has answered in the first part of the question to actually see if their knowledge actually stands, helping out results further to the report. The second part of the report is more qualitative, requesting the user give more detailed answers to the questions so it helps understand what they know about software security, where just using quantitative questions wouldn’t gather as much in this regard. The third part of the survey is more along the lines of quantitative questioning as it helps give truly honest answers from users and doesn’t look like the questions are trying to expose the user. Qualitative questioning in this part would mean either untrue answers or answers that would be hard to draw a conclusion to.

A selection of 20 people was chosen to answer all selected at random from different parts of the IT department of the West Herts college. The reason behind this method of sampling is that the answers should vary from person to person and should result in some interesting results. The people who answered the surveys will not have their names recorded and all physical copies will be disposed of indefinitely after the results are written up to excel.

## **4.2 How well do people know software vulnerabilities and vulnerabilities in general?**

For the first Question, it asked for the knowledge of software vulnerabilities. As the chart states, a majority of people were more along the lines of ‘average’ knowledge. Second best being ‘good’ knowledge, the main answer that should be expected of students working in the IT department, especially in regard to software development. The results were also taken from different departments, such as a business or others that wondered by, such as tutors, so their knowledge would be ‘poor’, however, what surprised me was the fact that no one answered ‘very poor’, meaning there was at least a low-level knowledge of software vulnerabilities.

Onto part b of the first question. This part was mainly used as a backup towards the first question, in which the reason behind was to verify if the person does indeed know what vulnerabilities are. Almost all answers were backed up clearly, but it’s interesting to note that the people that two people that answered ‘average’ and ‘good’ did not back up their question and instead either wrote ‘nothing’ in the box or didn’t write anything at all. This could mean many things, commonly though, it could mean that they actually have no idea or simply forgot.

The second question has the same format but asks on the types of vulnerabilities that they know off, rather than vulnerabilities in software. The expected result would be that at least a low-level amount of knowledge of viruses regardless, where certain participants did answer with a ‘good’ despite their answer in the first question of this part. However, it also happened vice versa (answered ‘poor’ or ‘very poor’) with a couple of participants, which is surprising considering that their results in the last question have been answered with ‘good’. This could mean anything.

Part b of this question is similar to the results in the first question, where answers backed up their knowledge. It’s important to note that those who answered with ‘Excellent’ gave more in-depth answers with examples to showcase their knowledge. This was the same as in the first question of this section, where those who answered ‘Excellent’ were sure to give answers that were different to those who answered ‘Good’ or ‘Average’ in the first part.

## **4.3 What can a programmer do to prevent and reduce software vulnerabilities?**

The next part of the survey was more of written answers and is the perfect basis for this question.

The first question was in software vulnerabilities, ‘How do you reduce vulnerabilities?’. All had answered with what would be commonly used to reduce vulnerabilities so at least a level of understanding is present with how they can reduce vulnerabilities. The quality in answers varies between departments, with those in business or other departments answering with ‘Install Anti-virus software’ or obvious answers along the lines. Those in software development departments answered more in depth, with ‘checking code’ or ‘software analysis’.

The second question revolves around areas of mitigation, ‘What sort of methods would make for good mitigation?’. This question revolves around what sort of methods would the developer take in case an attack should happen. Unfortunately, a majority of answers show that some people didn’t understand the question whatsoever, answering with nothing; common from those in departments other than software development, or answering with the same from the first question; common here from development departments. Only a couple of participants answered with proper mitigation techniques, so if the wording of the question being hard to understand may be up for debate.

The third question, ‘What are the issues of a vulnerable piece of software?’, goes into what consequences may arise if the software is not secure or developed well. As usual, the software development students and those who had good knowledge based on the first part gave pretty in depth and clear answers, but what is surprising is that some participants in business or other sectors gave good and relative answers that would be answered by those in software development, meaning that it could be picked up. Others in business or other sectors, were indifferent or just didn’t answer.

## **4.4 How do people apply security in their personal lives?**

The final part of the survey, investigates how secure people are in their personal lives and if they can apply it to their lives.

The first question asks if the participants go on social media. The chart above shows that all but one participant answered with ‘yes’, showing that they remain social online. The reason can vary for the one person who doesn’t go on social media, but in relation to the survey could be information sharing.

The next question asks how much personal information is shared online. The pie chart above shows that the majority of answers are ‘only a few’ and ‘Some of it’, these answers being that the only information given out is personal information to that of banks or login information. Only one answered ‘not at all’ so chances are that no personal information was given and no one said ‘all of it’, which went to what I expected.

The next question goes into if one or multiple emails are used for all uses, this being personal or business, etc. The majority of people said no, indicating that more than one email is used for various purposes. Some people did say ‘yes’, meaning they might be unaware of the risks associated to having only one email for all purposes.

The fourth question goes into the number of passwords used when logging in to various sites. The reason for this question is because of the risks accompanying having only one password and to understand if the participants know this risk. The chart shows that half of participants had voted to having ‘4+’ passwords, meaning that they understand the risks given to having minimal. What is important to note is that all participants have more than one password.

The fifth question revolves around which password is harder to crack. Unlike the rest of the survey, this is the only question that has a correct answer. Both passwords were written out and tested on its amount of time to crack by using the site <https://howsecureismypassword.net>, a site that safely checks the password and based on the computer, returns the amount of time it would take to crack it. The password ‘SomeInfuriatingOverworkedCars’ takes substantially more time to crack than ‘In54N3b0sS’ because of how the structure of the password.

The majority of people come to the conclusion, based on the bar chart, that ‘In54N3b0sS’ takes longer to crack than ‘SomeInfuriatingOverworkedCars’ because of the assumption of the numbers and the randomly cased letter would make for a reinforced password.

The next question is ‘Do you regularly install updates?’. The importance here being that installing updates makes for an up to date and secure version of the software. All 20 of the participants had answered ‘yes’. This cancels out the reason for a chart.

The final question tests the participant on their knowledge of phishing emails; emails that coax the user to giving valuable information to scammers. This question is set the same way as the questions in the first part of the survey with a second question that serves to back up the participant on their knowledge.

The chart above shows that the majority voted yes and in the second part of the question, gave sufficient factors of phishing links such as links in the email or poorly worded email addresses. Only one participant who answered ‘yes’ didn’t respond in the second part of the question.

# **Conclusion**

## **5.1 Primary Research Conclusion**

The results of the survey confirm numerous things; the answers to the main questions and how well people understand vulnerabilities. It is revealed that the results confirm that there are a majority of people that understand software vulnerabilities, security and what makes for secure software, but the questions have just barely been answered due to the fact that some of the questions in the survey were answered barely.

The first question reveals that people have an average understanding of software vulnerabilities and vulnerabilities in general. From the results of the first part of the survey reveal that the majority have answered both parts and most does back up their knowledge. Those who claim to have answered well and didn’t back up their answer show that they don’t actually understand vulnerabilities in software or in general.

The second main question shows that they only understand how to ensure that software security and what the risks associated with unsecure software can bring, rather than actually understand prevention methods to do in case of any attack. The results show that only a few can show what methods of prevention that a developer would carry out.

The third question answers the question of how they can apply security in their lives. The survey shows that the clear majority of participants take measures to ensure security and understand what sort of malicious stuff can arise, showing us that they are more than capable of applying security safely.

The primary research overall was more or less a success, as it did answer the questions that was the basis for this research. All twenty people managed to answer almost all of the survey with little to no trouble and has proved to provide a good amount of information that helped with answering the questions and assisting with recommendations. With the way I conducted the research, I managed to get the most honest answers that supported this report. The research was conducted with 20 people in the IT section of West Herts College, differing in branch and what was taught, such as business or software development, with emphasis on random sampling, giving us accurate and authentic results. Had the researched been structured differently, such as changing the sampling method or the people involved, then less than satisfactory results would be produced, making it harder to come to a complete verdict or result in biased answers.

### **5.1.1 Recommendations**

In response to the first question, it is important to understand the implications of software vulnerabilities as well as vulnerabilities in general, but given the results, I would say, in my opinion, that the knowledge presented was lacking from the following groups, both those involved in software development and business, despite their lack of utilising software. I would recommend that this knowledge is more enforced in both groups, educating software developers a more in depth view of software vulnerabilities and this would help considerably when it comes actually develop software, thus effectively apply their knowledge and techniques. With the other groups, at the least should have the knowledge of viruses and general vulnerabilities. The reason I’m recommending this is because viruses are a persistent threat in both normal and work lives. By learning this, they can easily recognize viruses and the threats associated with it, such as virus links in emails, etc.

My second recommendation mainly aims towards the group who are mainly involved in software development or similar. Those part of that group should gain more knowledge of software vulnerabilities, to pick up more on the foundations of said vulnerabilities, including methods of mitigation and security in software, also to be able to apply these secure methods when developing software. This in turn should help them understand software vulnerabilities and how to counteract them.

In regards to personal security, the answers from the survey generally showcase that in most cases, they understand the risks involved and do act on it (i.e. more than one email for different uses), leading to very interesting answers. I will still recommend that this knowledge of personal security still be enforced so people will improve on browsing the internet safely and other personal measures. This will also help with those who don’t normally use these measures when doing stuff online.

## **5.2 Secondary Research Conclusion**

This conclusion ties with the case studies of both the Yahoo data breach and the NHS ransomware attack, it is revealed that the sole reason behind the attacks were down to the negligence of both organisations, where the difference lies in how the attack was brought on or handled. But in both cases on the matter to the cause of attacks, it is revealed that the attacks were caused because of outdated software, leaving an exploit open towards the attacker; where the case of Yahoo was because of outdated and easy-to-crack security tools that allowed the attacker to easily access Yahoo’ database, along with personal names, emails and hashed passwords, spanning over a devastating three billion users, and regarding the NHS case, an old unsupported operating system, Windows XP, was used as the main OS for all computers within the trust centre, lacking security controls currently implemented in more modern operating systems, having the attacker to exploit old tools and poorly embedded and maintained user controls to take control and access the files to encrypt.

Now, onto how each case was handled, both were handled poorly; the attack caused disruptions to the NHS centres and in turn had caused cancellation of appointments, leading to outrage amongst patients. Yahoo’s was more controversial, with them only issuing a disclosure of the breach three years later from the actual attack, the main issue being that the security team had the knowledge that the foundation was breached by a hacker, but failed to even try to enforce any prevention method. The NHS attack had sorted out in the end, despite the ongoing disruption to their workstations and systems, but the Yahoo breach had left all the accounts at risk and action taken was late. This dealing a massive blow to Yahoo.

Both were down to negligence and in my opinion on both matters, it seems that the issues were down to the lack of mitigation techniques. In the case of yahoo, the factors here where the outdated security tools, lack of analysis and lack of action, as it was clear that they knew the attack had been carried out, but only brought the breach to light at a later date. Had there been monitoring of user activity, finding any unusual activity would have meant that there was a breach and in turn would lead to quicker response. Updated tools would have also meant that the possibility of an exploit becoming apparent would be very unlikely.

In the case of the NHS, it would’ve been simple to impose the newest, modern operating system, containing tools that aren’t present in older operating systems. Forcing cyber security standards within every trust would also have proven beneficial as that can impose tighter security and better embedded access controls.

# References

1. Software Development. [online]. Techopedia. Available from: <https://www.techopedia.com/definition/16431/software-development>. [Accessed 19 February 2018].
2. Elysium Academy Private Limited (2017). What are the Software Development Life Cycle (SDLC) phases? [online]. LinkedIn. Available from: <https://www.linkedin.com/pulse/what-software-development-life-cycle-sdlc-phases-private-limited>. [Accessed 2 March 2018].
3. Duncan Haughey (2014). REQUIREMENTS GATHERING 101. [online]. Project Smart. Available from: <https://www.projectsmart.co.uk/requirements-gathering.php>. [Accessed 02 March 2018].
4. Tree Swing Project Management [online]. (n.d). Available from: <https://www.tamingdata.com/2010/07/08/the-project-management-tree-swing-cartoon-past-and-present/>. [Accessed 02 March 2018].
5. Margaret Rouse, Matthew Haughn (2014). vulnerability. [online]. WhatIs.com. Available from: <http://whatis.techtarget.com/definition/vulnerability>. [Accessed 19 February 2018].
6. SecureList [online]. (n.d). Available from: <https://securelist.com/threats/software-vulnerabilities/>. [Accessed 19 February 2018].
7. Apple Developer [online]. (2016). Available from: <https://developer.apple.com/library/content/documentation/Security/Conceptual/SecureCodingGuide/Articles/TypesSecVuln.html>. [Accessed 19 February 2018].
8. Inter Process Communication (IPC). [online]. Techopedia. Available from: <https://www.techopedia.com/definition/3818/inter-process-communication-ipc>. [Accessed 19 February 2018].
9. OWASP (2016). Insecure Temporary File [online]. Available from: <https://www.owasp.org/index.php/Insecure\_Temporary\_File>. [Accessed 19 February 2018].
10. Jonathan Stempel, Jim Finkle (2017). Yahoo says all three billion accounts hacked in 2013 data theft. [online]. Reuters. Available from: <https://www.reuters.com/article/us-yahoo-cyber/yahoo-says-all-three-billion-accounts-hacked-in-2013-data-theft-idUSKCN1C82O1>. [Accessed 20 February 2018].
11. Vindu Goel, Nichole Perlroth (2016). Yahoo Says 1 Billion User Accounts Were Hacked. [online]. The New York Times. Available from: <https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html>. [Accessed 26 February 2018].
12. Michael Kan (2017). Yahoo execs botched its response to 2014 breach, investigation finds. [online]. CSO. Available from: <https://www.csoonline.com/article/3176181/security/yahoo-execs-botched-its-response-to-2014-breach-investigation-finds.html>. [Accessed 26 February 2018].
13. NHS 'could have prevented' WannaCry ransomware attack (2017). [online]. BBC News. Available from: <http://www.bbc.co.uk/news/technology-41753022>. [Accessed 1 March 2018].
14. Josh Fruhlinger (2017). What is WannaCry ransomware, how does it infect, and who was responsible? [online]. CSO. Available from: <https://www.csoonline.com/article/3227906/ransomware/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html>. [Accessed 1 March 2018].
15. Support for Windows XP ended [online]. (2014). Available from: <https://www.microsoft.com/en-gb/windowsforbusiness/end-of-xp-support>. [Accessed 1 March 2018].
16. Willy Jimenez , Amel Mammar and Ana Cavalli (2014). Software Vulnerabilities, Prevention and Detection Methods: A Review. Evry, France: ResearchGate. Available from: <https://www.researchgate.net/publication/253704494\_Software\_Vulnerabilities\_Prevention\_and\_Detection\_Methods\_A\_Review\_1>. [Accessed 03 March 2018].
17. Common cyber attacks: reducing the impact. (2016). National Cyber Security Centre. Available from: <https://www.ncsc.gov.uk/content/files/protected\_files/guidance\_files/common\_cyber\_attacks\_ncsc.pdf>. [Accessed 03 March 2018].