Faculdade de Engenharia da Universidade Do Porto

Mestrado Integrado em Engenharia Informática e Computação

Engenharia de Software

T1 – History of Software Engineering

Alunos:

João Gama Amaral

João Nuno Ferreira

Pedro Galvão

12 de Outubro, 2018

Índice:

Software Failure Cases

WannaCry Ransomware Cyber attack (2017)

What happened?

On Friday May 12th, the world awoke to news of a new ransomware that was attacking computers throughout some countries, quickly making its way globally. The program, named WannaCry, **has infected hundreds of thousands of computers, making it one of the most destructive cyberattacks in history**. It involved more than 700 different malware samples with a view to encrypting files with different extensions. The ransomware attack has struck hospitals, communication, and other types of companies and government offices around the world, seizing control of affected computers until the victims pay a ransom.

What is WannaCry?

WannaCry is a ransomware program that was used to attack unprotected computers throughout the world. The malware has been described by some as an SMB (Server Message Block) worm. It arrives on the infected computer in the form of a dropper, a self-contained program that extracts the other application components embedded within it. Those components include:

- An application that encrypts and decrypts data;
- Files containing encryption keys;

This malware variant incorporates a code to exploit vulnerability published by Microsoft on March 14th, known as **EternalBlue**. EternalBlue is a leaked NSA exploit of the SMB protocol in Microsoft Windows that is used to propagate the malware in affected systems.

The program code was relatively easy for security pros to analyse. Once launched, WannaCry tries to access a hard-coded URL (the so-called ***kill switch***). If it can't, it proceeds to search for encrypted files in a slew of important formats, ranging from Microsoft Office files to MP3s and MKVs, leaving them inaccessible to the user. Then it displays a ransom notice from the attacker, demanding payment of a ransom within 3 days using bitcoin to decrypt the files. Anyone who refused to pay would eventually lose access to their files and information stored in them.

What was the impact?

More than **300000 computers were attacked in over 100 countries;** the worst hit was Britain's National Health Service, affecting 36 hospitals across the country. Globally, companies that were affected include Nissan Motors, FedEx, China National Petroleum, Renault SA, Deutsche Bahn, Hitachi, Sberbank of Russia, Yancheng police department in China, and the Russian Interior Ministry. Companies suffered monetary setbacks on account of business losses. Cyber risk modelling firm Cyence estimates the potential costs from the hack at 4 billion dollars. This also includes loss of productivity, cost of restoration of data and cost of investigation.

What caused it? How did it spread?

Wellsmore and other cybersecurity experts say the identity of the perpetrators is still unknown. The hackers were using tools stolen from the U.S National Security Agency and released on the internet. As said, WannaCry incorporates a code known as EternalBlue. Eternal Blue is an SMB exploit affecting various Windows operating systems from XP to Windows 7 and various flavours of Windows Server 2003 & 2008. The exploit is known as HeapSpraying and is used to inject shellcode into vulnerable systems allowing for the exploitation of the system. The code is capable of targeting vulnerable machine by IP address and attempting exploitation via SMB por 445.

Expected behaviour of the software:

- Sends an SMB Echo request to the targeted machine;
- Sets up the exploit for the target architecture;
- Performs SMB fingerprinting;
- Attempts exploit;
- If successful exploitation occurs, WIN;
- Pings the backdoor to get an SMB reply;
- If the backdoor is not installed, it's game on;

The ability of this code to beacon out the other potential SMB targets allows for propagation of the malicious code to other vulnerable machines on connected networks. This is what made the WannaCry ransomware so dangerous. The ability to spread and self-propagate causes widespread infection without any user interaction.

How to avoid it?

The ability of this code to beacon out the other potential SMB targets allows for propagation of the malicious code to other vulnerable machines on connected networks. This is what made the WannaCry ransomware so dangerous. The ability to spread and self-propagate causes widespread infection without any user interaction.

References

https://acp.us.com/the-wannacry-attack-what-happened/

https://www.cnet.com/news/facebook-unveils-new-portal-video-chat-devices-for-the-smart-home/

https://blog.malwarebytes.com/cybercrime/2017/05/how-did-wannacry-ransomworm-spread/

https://techspective.net/2017/09/26/wannacry-ransomware-detailed-analysis-attack/

https://www.businesstoday.in/magazine/technology/wannacry-ransomwarecomputershackersattackransomwarebitcoinscryptocurrencyvirtual-currencymonero/story/258734.html

https://www.csoonline.com/article/3227906/ransomware/what-is-wannacry-ransomware-how-does-it-infect-and-who-was-responsible.html

https://www.pandasecurity.com/mediacenter/src/uploads/2017/05/1705-Informe_WannaCry-v160-en.pdf

Patriot Missile (1991)

What happened?

The Gulf War operations, which started at 2 August 1990 and took end on 28 February 1991, led to the accumulation of troops and defense on Saudi Arabia. In this war participated a coalition force from 35 nations commanded by the United States against Iraq, due to Iraq's invasion and annexation of Kuwait. During this was war the **US Patriot missile defense system was used in combat for the first time.

What is The Patriot?

The Patriot is an Army surface-to-air, mobile, air defense missile system. Since the mid-1960s, the system evolved to defend against aircraft and cruise missiles as well. During the Gulf War it was being developed to counter short-range ballistic missiles.

This system was designed to operate against Soviet missiles traveling at speeds up to about 2448 km/h. To avoid detection it was designed to be mobile and operate for only a few hours at one location.

The weapons control computer of the Patriot obtains target information from the system's radar. The radar sends out electronic pulses that scan the air space above it. The pulses that actually hit a target are reflected back to the radar system and interpreted. If the object that was found is recognized as a missile, the Patriot is instructed to intercept it.

What was the failure impact?

On the day 25 of February, one of the Patriot missile systems failed to intercept and destroy an enemy missile. Subsequently this one ended up hitting Army barracks, killing 28 Americans and making over 100 injuries.

Why did it happen?

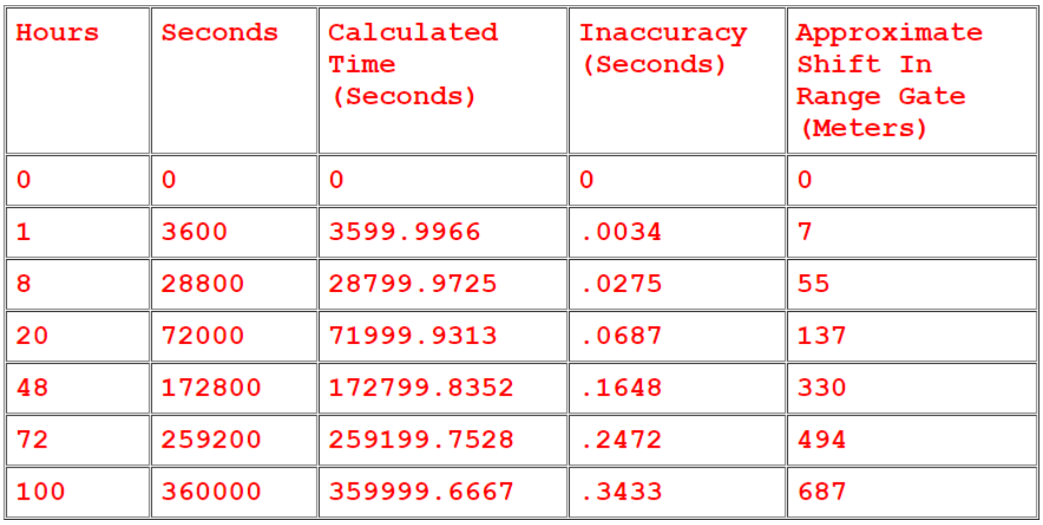
Since it was the first time the Patriot was being used to defend against Scud missiles, which were being used by the Soviet Army and fly at approximately 6035 km/h, the Army still was learning how to track and intercept them. Besides this, the Patriot system was equipped with a data recorder that would save the overall system performance information, but it was not being used.

Two weeks before the incident, the Army officials received Israeli data that notified them that, after 8 consecutive hours of operating the Patriot system, this one was showing some loss in accuracy.

The system that was active during the fail of the missile interception, was working uninterruptedly for more than 100 hours. The Patriot failed its purpose due to a software problem in the weapons control computer, which "led to an inaccurate tracking calculation that became worse the longer the system operated", according to a report about the accident.

The new and improved software system arrived to Saudi Arabia on February 26, the day after.

The whole problem on the accuracy was the fact that velocity is a real number that cannot be represented completely with decimal numbers and the fact that although the time on the system is measured in tenths of second, it is shown as an integer. Therefore, because of how the Patriot software calculated and because the registers were only 24 bits long, the conversions and calculations could not be more precise than 24 bits. When the system kept operating for so long, the error started accumulating leading to a loss of precision.



Hours	Seconds	Calculated Time (Seconds)	Inaccuracy (Seconds)	Approximate Shift In Range Gate (Meters)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0275	55
20	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100	360000	359999.6667	.3433	687

Fig. 1 – Uninterrupted Run Time Effect on Patriot system.

How to avoid it?

The software was changed in order to compensate for the inaccurate time calculation. However, the officials never announced to the public what were the changes. Nonetheless, it can be easily seen that the biggest mistake was not using the data recorders. If these were used, the problem in the accuracy could have been discovered and solved earlier and 28 lives would have been saved.

Besides the data recorder, it can be pointed that, when developing software for something as critical as missile systems, the software engineers must be focused on not letting such problems go unsolved.

At last, the officials can be blamed as well, since they did not take in attention the recommendations that the Patriot system should be turned off and on every few hours, which would reboot the computer's clock, setting the time back to 0.

Reference:

https://www.cs.drexel.edu/~introcs/Fa10/notes/07.1_FloatingPoint/Patriot.html?CurrentSlide=10

https://embeddedgurus.com/barr-code/2014/03/lethal-software-defects-patriot-missile-failure/

https://www.nytimes.com/1991/06/06/world/us-details-flaw-in-patriot-missile.html

http://www-users.math.umn.edu/~arnold/disasters/patriot.html

http://www.dtic.mil/dtic/tr/fulltext/u2/a344865.pdf

HSBC Online Banking Glitch (2015)

In the morning of 28th august, Friday previous to a banking Holiday in UK, the HSBC suffered with a bug in its electronic payment system, causing, among other problems, failures to complete about 275.000 individual payments.

The HSBC said the problem was caused by an error in information in a file sent to Bacs, a system used in the UK to process millions of payments realized by banks. The error prevented lots of companies to complete payments to their staff. Enterprises that realized bank transactions with the HSBC were unable to pay their suppliers.

Consequently, lots of people were unable to bear costs, bills and plans made to the holiday of 29th to 31th august. Several complaints were registered in enterprises affected and in social media by individuals who were affected. For example, some people told they had to cancel travel plans, or were unable to buy a house, or had to overdrawn money to pay their bills. The event was also clearly harmful to HSBC, whose reputation is damaged under the News and client complaints in social media.

HSBC apologized for the problem and said no one should lose out as a result of the failure.

“We are taking immediate steps to ensure the payments reach beneficiaries as quickly as possible. We will work with other banks to ensure that customers do not lose out as a result of today’s problems.”

The next day the bank affirmed all the delayed payments were already processed.

Software glitches like this often cause problems of such magnitude in banking systems. The HSBC itself suffered with other bugs, for example in Brazil in September of the same year and in the UK again in January 2016. Other examples were the glitch in the TSB’s system in April 2018 and in Suntrust’s system in September 2018.

The glitch in HSBC was not explained in details, so it becomes hard to precise the causes of the problem in terms of software functioning or the work of the IT staff. However we can notice general factors that increase the occurrence of these failures in electronic banking systems.

One of this factors is the fact that nowadays these systems nowadays are updated increasingly more often, and it brings more risk of introducing errors.

Another factor is the increase in complexity of the systems. As banks try to adapt their systems to new necessities and higher volume of data, more functions are introduced upon codes written previously, creating increasingly complex systems. Software are not built from zero in each implementation because the costs to do it would be too high. They are made incrementally by different people at different times, and for this reason sometimes it is harder to notice its weak spots.

Furthermore, electronic banking systems nowadays are strongly interconnected and consequently if anything goes wrong the failure can spread over the system causing much bigger problems.

Reference:

The Guardian (2015, August 28) *HSBC system failure leaves thousands facing bank holiday without pay*. Available at: https://www.theguardian.com/money/2015/aug/28/many-hsbc-customers-facing-payday-without-pay

BBC (2015, August 29) *HSBC glitch payments 'all processed'*. Available at: https://www.bbc.com/news/uk-34095626

Itv (2015, August 29) *HSBC says all delayed payments now processed.* Available at: http://www.itv.com/news/update/2015-08-29/hsbc-all-payments-hit-by-glitch-now-processed/

Channel 4 (2015, April 28). *HSBC IT glitch leads to payday moans*. Available at: https://www.channel4.com/news/hsbc-it-glitch-leads-to-payday-moans

Cast Software (2016, March 7) *The HSBC Failure Has Many Wondering: Are Banking Providers Taking the Appropriate Measures to Ensure Code Quality and System Dependability?* Available at: https://www.castsoftware.com/blog/the-hsbc-failure-has-many-wondering-are-banking-providers-taking-the-appropriate-measures-to-ensure-code-quality-and-system-dependability

American Banker (2018, September 18) *Why those high-profile bank glitches just won't stop.* Available at: https://www.americanbanker.com/news/why-those-high-profile-bank-glitches-just-wont-stop

Folha de São Paulo (2015, September 13) *HSBC tem falha no sistema, e clientes enfrentam problemas para usar cartão.* Available at: https://www1.folha.uol.com.br/mercado/2015/09/1681200-hsbc-tem-falha-no-sistema-e-clientes-tem-problemas-para-usar-cartao.shtml

The Telegraph (2018, September 3) *TSB customers locked out of accounts in new online banking glitch*. Available at: https://www.telegraph.co.uk/personal-banking/current-accounts/tsb-customers-locked-accounts-new-online-banking-glitch/

Reuters (2016, January 5) *HSBC apologizes for online banking outage, says customers will not 'lose out'*. Available at: https://www.reuters.com/article/us-hsbc-it/hsbc-apologizes-for-online-banking-outage-says-customers-will-not-lose-out-idUSKBN0UJ0ZB20160105

No Silver Bullet – Reading notes

Today, we fear all the problems that may appear upon us just as much as werewolves are feared in tales. But our silver bullet is non-existent and will highly likely not appear in a near future.

Essential Difficulties

Complexity: Software entities are more complex than perhaps any other human construct, it differs profoundly from computers. A scaling-up of a software entity means an increase in the number of states, increasing the complexity more than linearly being the cause of several problems, such as lack of team communication, unreliability of program, hard to increase new functions without side effects and less security.

Conformity: Software must confirm because it has most recently come to the scene. In others it must conform because it is perceived as the most conformable. Much complexity comes from conformation to other interfaces, this cannot be simplified out.

Changeability: When a software product is found to be useful, people try it in new cases, beyond or at the edge of the original domain. The software product is embedded in a cultural matrix of applications, users, laws and machines vehicles. The continuous change in them all forces change upon the software product.

Invisibility: Unlike building, software cannot be seen, thus depriving the mind of some of its most powerful conceptual tools. This also means that when trying to diagram the software structure, several graphs representing relationships must be considered, which makes hierarchies difficult to visualize.

Past Breakthroughs that Solved Accidental Difficulties

High-level languages: The most powerful stroke for software productivity, reliability and simplicity has been the progressive use of high-level languages. It frees programs from much of its accidental complexity. At some point the elaboration of high-level language becomes a burden that increases the intellectual task of the user.

Time-sharing: It preserves immediacy, and hence enables us to maintain an overview of complexity. The principle effect is to shorten system response time. As it goes to zero, at some point it passes the human threshold of noticeability, about 100 milliseconds. Beyond that no benefits are to be expected.

Unified programming environments: Unix and Interlisp attack the accidental difficulties of using programs together, by providing integrated libraries, unified file formats and piles and filters.

Hopes for the silver

Object-oriented programming: The result is to remove a high-order sort of accidental difficulty and allow a high-order expression of design. An order-of-magnitude gain can be made by object oriented programming only if the unnecessary underbrush of type specification remaining today in our programming language is itself responsible for nine-tenths of the work involved in designing a program product.

Artificial intelligence: Many people expect artificial intelligence to advance in a way to provide the revolutionary breakthrough that will give order-of-magnitude gains in software productivity and quality.

Expert systems: Contains a generalized inference engine and a rule base, designed to take input data and assumptions and explore the logical consequences through the inferences derivable from the rule base, and offering to explain its results by retracing its reasoning for the user.

Automatic programming: The generation of a program for solving a problem from a statement of the problem specifications. The system assessed the parameters, chose from a library of methods of solution, and generated the programs. It is hard to see how such techniques generalize to the wider world of the ordinary software system.

Graphical programming: It has proved to be essentially useless as a design-tool programmers draw flow charts after, not before, writing the programs they describe. Software is very difficult to visualize, whether we diagram, we feel only one dimension of the intricately interlocked software elephant.

Program Verification: It is a very powerful concept, and it will be very important for such things as secure operating system kernels. Program verification does not mean error-proof programs. Whereas verification might reduce the program-testing load, it cannot eliminate it.

Promising attacks on the conceptual essence

Buy versus build: The most radical possible solution for constructing software is not to construct it at all. Another way of looking at it is that the use of n copies of a software system effectively multiplies the productivity of its developers by n.

Requirements refinement and rapid prototyping: The hardest single part of building a software system is deciding precisely what to build. One of the most promising of the current technological efforts is the development of approaches and tools for rapid prototyping of systems as part of the iterative specification of requirements.

Incremental development: Grow, not build, software. It is time to change again. The conceptual structures we construct today are too complicated to be accurately specified in advance, and too complex to be built faultlessly.

Great designers: The central question of how to improve the software art centres, as it always, on people. Good designs are achieved by following good practices instead of poor ones. A little retrospection shows that although many fine, useful software systems have been designed by committees and built by multipart projects, those software systems that have excited passionate fans are those that are the products of one or a few designing minds, great designers. Consider Unix, APL, Pascal and contrast with Cobol, PL/I, Algol...