Analysis of the Wide-Spread Impact by the NSO Group's Pegasus Exploit

KIET HA

CPSC 385: COMPUTER SECURITY

PROF. EWA SYTA

DECEMBER 13, 2022

Contents

1 Introduction **2**

2 Background **2**
2.1 The NSO Group 2
2.2 Apple's iOS System 3
2.2.1 WebKit 3
2.2.2 Kernel System 3

3 Exploitation **5**
3.1 Payload 5
3.1.1 Achieving phone access 5
3.1.2 XNU Exploitation 7
3.1.3 Jailbreak 8

4 Impact **9**

5 Solution **10**
5.1 Defensive mechanisms 10
5.2 Aftermath 10

6 Conclusion **11**

Abstract

The Pegasus spyware is a highly advanced piece of surveillance software that is capable of infiltrating and monitoring smartphones. Pegasus was first reported in 2016 using spear-phishing-text messages or emails, developed by the NSO Group, a Israeli-based cyber-arms firm, and has since advanced its capabilities to allow for "zero-click" attacks that don't require user input to activate. The zero-click exploit is able to bypass the security measures on smartphones and gain access to a wide range of sensitive data, including text messages, emails, location data, and even live audio and video. The exploit has been used to target over 50,000 individuals including journalists, human rights activists, and political dissidents, and has been criticized for its potential to be used for citizen surveillance. This paper will dive deep into Apple's security measures to present the technical analysis of the exploits, the damaging impact of the spyware and what defensive measures that took place to ensure this would not reoccur.

1 Introduction

Pegasus was first founded in 2016, in a failed attempt to install on a human rights activist, Ahmed Mansoor. He received messages on his phone promising details about the state of the prisoners in the United Arab Emirates [1]. Although he did not fall for this and avoided the situation of being compromised, this led the exploit to burst into global prominence. Pegasus did not stop there but continued to progress; once first was a spear-fishing attack, has become a zero-click attack. Once infected, whatever the phone compromised is all in the control of the attacker. The exploit has been used for espionage and theft of sensitive information. Spyware has been used by police and intelligence services to catch terrorists, but Pegasus has became much larger. While it is unclear how much data Pegasus has recorded, it breached privacy worldwide as it was used on many citizens. Although the NSO Group has denied any wrongdoings of their technology, it is clear that Pegasus originated from them [2]. Due to the severity and complexity of the exploit, it is necessary to gain insight into Pegasus to prevent future attacks from happening like this.

There are several vital questions that the paper will explore:

1. Who is the NSO Group and why are they using surveillance spyware on users?
2. What is the attack methodology Pegasus used to get to the user's device?
3. What privileges were given to the attacker once compromised?
4. Who was affected by the attack?
5. What was done to stop the attack?
6. What are potential precautions users can do to prevent from attacks like this?

2 Background

2.1 The NSO Group

The NSO Group is an Israeli cyberintelligence company that develops and sells technology products and services for governments and law enforcement agencies. According to the company's website, their products are designed to help these organizations prevent and investigate terrorism, crime, and missing people. They also assist rescue teams to provide a safe environment for citizens to live in [3]. The company offers a range of products and services, including spyware, mobile network exploitation tools, cyber intelligence platforms, and digital forensics tools. The NSO Group often exploits zero-day vulnerabilities - security vulnerabilities that consist of loopholes and bugs in the software that are unknown to the manufacturer. These vulnerabilities can be sold to the government or third-party companies which can be used for mischievous activities. The group outlines that they take accountability such as following ethical standards such as Human Rights Policy very seriously. However, the company has been criticized for its products being used by governments to engage in human rights abuses, including the 2019 WhatsApp spyware and the most recent Pegasus spyware, which has been used to target journalists and human rights activists [4].

2.2 Apple's iOS System

To understand the payload of the vulnerability better, some information about Apple's WebKit system in correlation to Safari and Apple's Kernel System needs to be explored.

2.2.1 WebKit

WebKit is a free open-source web browser engine that is used by many web browsers, including Apple's Safari. It is a layout engine that is designed to render web pages and provide a native-like experience for web applications. WebKit is used by Apple on its macOS, iOS, and iPadOS operating systems. WebKit is also an open-source and cross-platform web browser engine meaning it can be used by third-party developers and even competitors in their own projects. This allows for a wide range of applications and allows for greater collaboration and innovation in the development of web-based technologies. However, since it is open source, anyone can use and modify but also find vulnerabilities that could be dangerous if taken advantage of.

Apple has implemented several security measures in WebKit to protect against malicious web pages and protect users' privacy. WebKit supports the use of Content Security Policy (CSP), which allows website owners to specify which sources are allowed to load content on their pages, helping to prevent malicious code injection [5]. There is also the use of Transport Layer Security (TLS) and Secure Sockets Layer (SSL) encryption to secure communication between the browser and web servers, protecting sensitive information such as login credentials from being intercepted by attackers. WebKit includes built-in protection against cross-site scripting (XSS) attacks, which are a common method used by attackers to inject malicious code into web pages [6]. WebKit uses a sandboxing technology to isolate different web pages and prevent them from accessing sensitive information or compromising the security of the system. Pegasus expliots a memory corruption vulnerability exists in Safari WebKit that allows an attacker to bypass sandbox to execute arbitrary code. More details about sandboxing will be discussed later.

2.2.2 Kernel System

Apple's kernel system, XNU, is the core of the operating system that powers its devices [7]. The kernel is the first part of the operating system to load into memory during device starting up, and it remains there for the entire duration of the device session due to its services are required continuously. The kernel is responsible for managing the resources of the device and system calls, such as its memory, processor, and storage, and providing the necessary interfaces for applications to access these resources. The kernel code is loaded into a protected area of memory so it does not get overwritten by an other programs by the operating system. To protect XNU and prevent unauthorized access, there are many security mechanisms that are built into the kernel. The main components that the paper is going to discussing is Sandboxing, Kernel Address Space Layout Randomization (kASLR) and Code Signing.

iOS sandbox is a security feature in the iOS operating system that limits the access an app has to the device's resources and gathering or modifying information from other apps. This is intended to prevent malicious apps from gaining unauthorized access to sensitive data or causing harm to the device. The sandbox enforces strict rules on which actions and data an app is allowed to access, and it acts as a barrier to prevent unauthorized access [8]. Once sandboxed, each app has a randomly assigned unique home directory for its files when installed. Sandboxing protects the security and privacy of the user's data, system files, resources, as well as the stability of the system. (Figure 1)

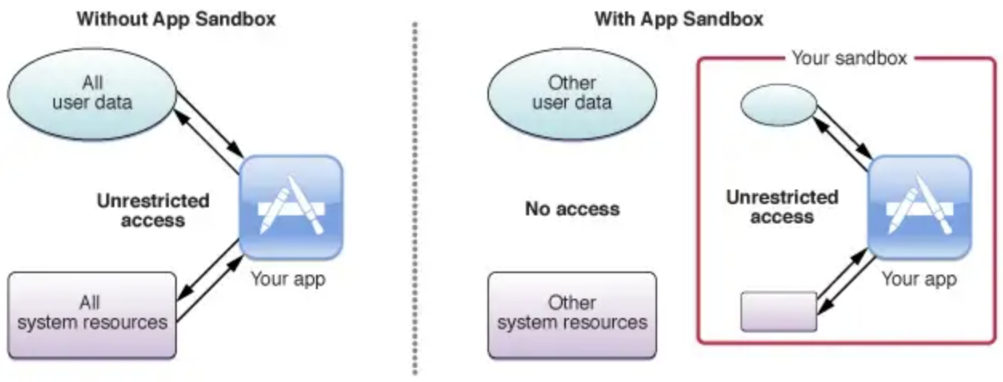


Figure 1: iOS Sandbox Protection [8]

To protect the kernel's base, XNU adds Kernel Address Space Layout Randomization (kASLR). kASLR randomizes the memory addresses used by the operating system kernel, making it more difficult for attackers to predict where certain data or code is located in memory [9]. Attackers will find a much harder time to exploit vulnerabilities or perform other types of malicious activity in memory. Before attacking the kernel, Pegasus has to find where the kernel is located. Once attackers has gained access to a kernel address, they could potentially use it to modify or delete sensitive information, or to gain access to other parts of the system.

Code signing is a security feature in Apple's operating systems that is used to verify the authenticity and integrity of software on a device [10]. When software is code signed, a digital signature is added to it that can be used to verify the identity of the developer and ensure that the software has not been tampered with. To use code signing, developers must first obtain a code signing certificate from Apple. This certificate includes the developer's identity and a public and private key pair. The developer can then use the private key to sign the software, and the public key can be used by the operating system to verify the signature and ensure that the software is from a trusted source. Code signing is an important security measure because it allows users to trust that the software they are installing on their device is from a trusted source and has not been modified in any way. This helps to protect against malware and other security threats that can be introduced through untrusted software.

Apple's kernel system plays a crucial role in the performance and security of its devices. It manages the device's resources and provides the necessary interfaces for applications to access them, while also implementing security measures to protect against unauthorized access and malware. By ensuring that XNU is secure and efficent, Apple is able to offer such high quality and reliable computing experience.

3 Exploitation

Pegasus is high level espionage software achieved via the Trident exploit chain, a set of three zero-day vulnerabilities in Apple's iOS operating system.

3.1 Payload

The figure below shows how the 3 vulnerabilities were the building blocks in developing the three stages to install the surveillance software - **CVE-2016-4657**: Memory Corruption in Safari WebKit, **CVE-2016-4655**: Kernel Information Leak Bypassing KASLR, and **CVE-2016-4656**: Memory Corruption in Kernel that leads to Jailbreak [11]. (Figure 2)

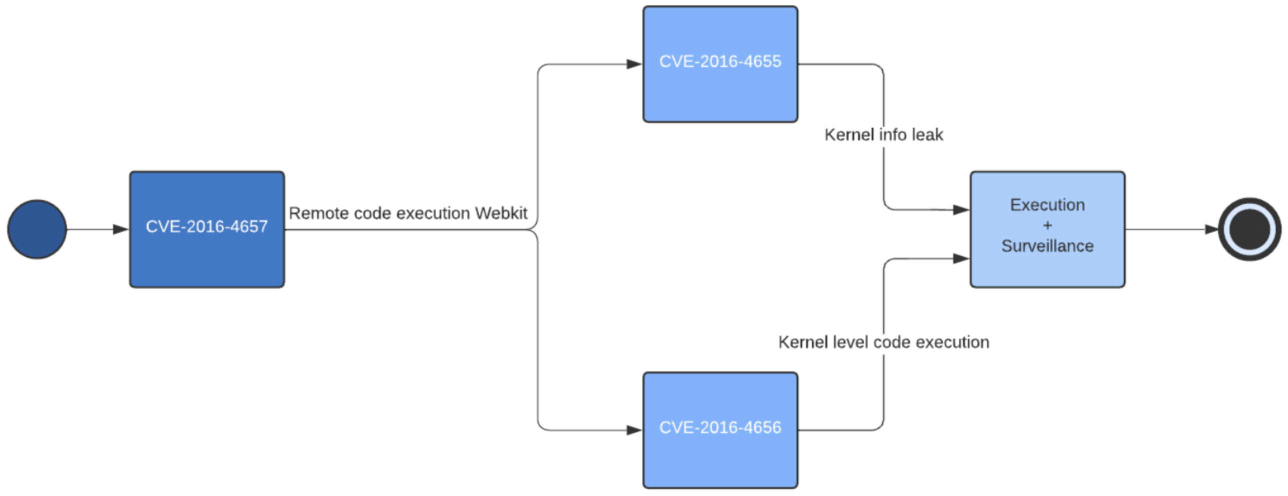


Figure 2: The 3 Stages to Exploitation of User's Phone

3.1.1 Achieving phone access

The attack sequence begins with a simple phishing scheme which starts by the attacker sending a text that will look like to benefit your life in some way. For Pegasus, it started as a spear phish URL, where the attackers have done some research on the victim and their organization in order to create a more convincing and compelling message. That is what happened to Ahmed Mansoor, a very well recognized human rights defender, received SMS text messages on his iPhone promising him new secrets about detainees tortured in United Arab Emirates jails if he clicked the link. Instead of clicking, Mansoor sent the messages to Citizen Lab researchers. Citizen Lab researchers found out that the link was composed of a vulnerability in Safari's WebKit that would allow the attacker to compromise the device. The link contained obfuscated JavaScript, remote code execution in WebKit, and URLs for stage 2 of the attack [1].

CVE-2016-4655 is a user after free vulnerability in WebKit's JavaScriptCore library. A use-after-free vulnerability is a type of computer security vulnerability that occurs when an application continues to use memory after it has been deallocated or freed. This can lead to memory corruption, which can in turn be exploited by an attacker to gain unauthorized access to a system or cause other types of harm. For this vulnerability, the bug existed in the `slowAppend()` method for `MarkedArgumentBuffer` which can be triggered by the usage of `MarkedArgumentBuffer` in the `defineProperties()` method, all of which rely on WebKit garbage collector [12].

The `defineProperties()` method defines new or modify its properties directly on object it takes. It takes a couple arguments, the object itself and the properties objects which can have a descriptors that constitute the property to be defined or modified [13]. The very first pass of the method, each property descriptor was checking for correct formatting and the object gets appended to a descriptors vector. After each property descriptor has been validated, the `defineProperties()` will associate each of the user-supplied property with the target object. A problem exists here because it's possible when `defineProperties()` is called, it's possible to call any user-defined JavaScript method causing the garbage collection cycle to be triggered. The garbage collector is responsible for deallocating an object from memory when they are no longer reference it. It works through the stack and check for reference to an object. To make sure that the reference to property descriptors don't become stale, they need to be protected from being garbage collected which is done by `MarkedArgumentBuffer`.

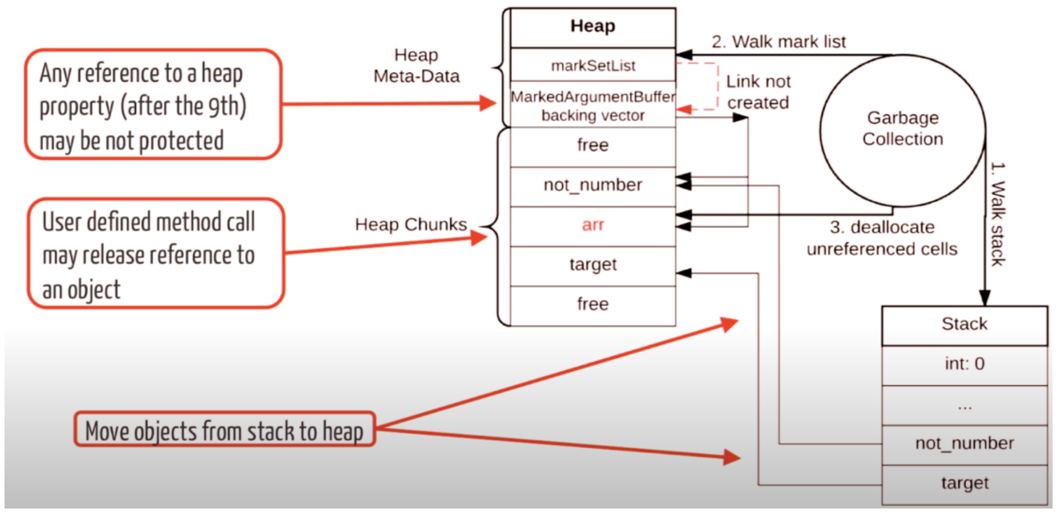

Any reference to a heap property (after the 9th) may be not protected
User defined method call may release reference to an object
Move objects from stack to heap
Heap Meta-Data
Heap Chunks
Heap
markSetList
MarkedArgumentBuffer backing vector
free
not_number
arr
target
free
2. Walk mark list
Link not created
Garbage Collection
1. Walk stack
3. deallocate unreferenced cells
Stack
int: 0
...
not_number
target

Figure 3: CVE-2016-4655's WebKit memory attack [12]

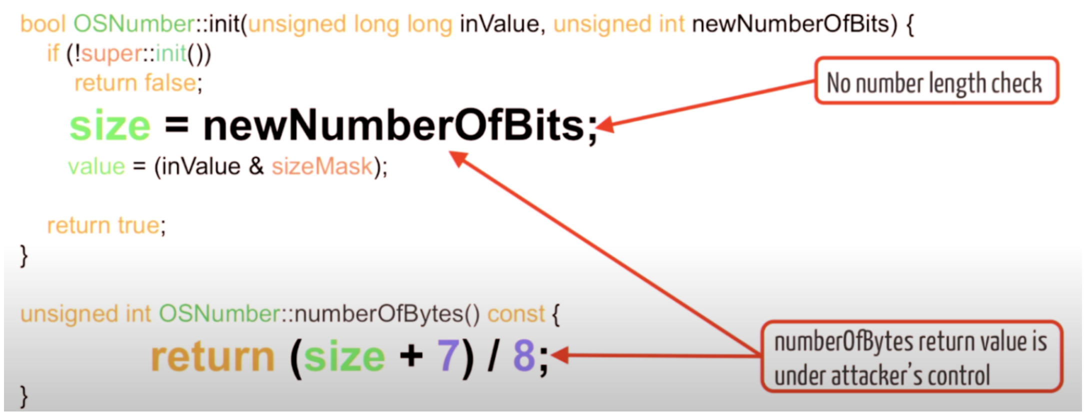
`MarkedArgumentBuffer()` initially maintains an inline stack buffer containing 8 values. However, when the 9th value is added, the capacity will be expanded. The method `slowAppend()` will then be ran to move from a stack memory to a heap memory. Once that happens, objects are not automatically protected from a garage collection so they need to be added to heap's `m_markListSet`. This ensures that the object will not be deallocated by a garbage collection cycle. However, when the heap acquires a complex object so for primitive types for booleans, integers, they are not going to be added into the `m_markListSet`. Therefore, when the buffer is moved from stack memory to heap memory and one of the properties is a simple types, they are not automatically protected by garbage collection and all the next corresponding values will not be as well. The `slowAppend()` will fail to acquire the Heap. (Figure 3)

The Pegasus exploit triggers the vulnerability with a specifically crafted sequence of properties passed in the `defineProperties()` method to trigger improper deallocation of a `JSArray` object. The exploit has code written to repeatedly attempt to trigger garbage collection at a specific time to allocate another object over the stale `JSArray`. Instead of a `JSArray`, the exploit uses an allocation of Unit32Array to achieve arbitrary read/write primitive which is done by corrupting the length. By doing so, this bypasses the Sandbox and allows the exploit to gain arbitrary native code execution, which it uses to execute its payload.

3.1.2 XNU Exploitation

Stage 2 of the Pegasus attack will set up the pieces needed to gain privileges to the user's phone and jailbreak the device. Two vulnerabilities were used to find the kernel's memory location and achieve kernel level code execution. In order to access the kernel, programs must use specific memory addresses that are reserved for the kernel. Theses addresses are protected by many security measures to prevent unauthorized access and ensure the security of the kernel.

CVE-2016-4655 is a info-leak vulnerability, a security vulnerability that discloses kernel memory by bypassing kASLR. The vulnerability exists within the function `OSUnserializeBinary()` which converts a binary format to a basic kernel data object. The function supports different container types, sets, dictionaries, arrays, object types, strings, numbers. However, the reason this function causes issues is that there is no length check to `OSNumber` used within the method, where the Pegasus exploit will use to override the size variable [14]. Since size is used in other methods of `OSNumber`, `numofBytes` is now controlled by attacker. The `newNumberOfBytes` return value is under attacker control and it is copying memory from kernel stack to heap by using `IORegistryEntryGetProperty` function [12]. The max size of `OSNumber` is 64 bits so Pegasus takes advantage of this by making a dictionary with `OSNumber` with length of 256 bits. Since the `OSNumber` user client object is now in the kernel heap due to the excessive bits, it can be used to read the kernel pointers by it's properties and calculate where the kernel base is. Knowing the kernel addresses, Pegasus will then use CVE-2016-4656, another use-after-free exploit to gain kernel level code execution. (Figure 4)



```
bool OSNumber::init(unsigned long long inValue, unsigned int newNumberOfBits) {
    if (!super::init())
        return false;
        size = newNumberOfBits;
        value = (inValue & sizeMask);

    return true;
}

unsigned int OSNumber::numberOfBytes() const {
        return (size + 7) / 8;
}
```

Figure 4: OSNumber Missing Bound Check [12]

The exploit employs a memory corruption to disable code signing enforcement, allowing the running of unsigned binaries. CVE-2016-4656 takes freed memory that would have references and reallocate it with shell code. This is done by reallocating freed `OSString` with `OSData` buffer filled with zeros, both of which are 32 bytes. When reallocating, retain() is called since `OSUnserializeBinary` parses the reallocation. This causes the kernel to get the fake vtable pointer from the buffer Pegasus created which points at zero. Since it is zero, the kernel dereferences the pointer, and points it at the retain offset called by retain() at an offset of `0x20`. This happens at page zero in memory, which is what all null pointers reference. Now the attacker has control of RIP, the instruction pointer, pointing at `0x20` [14]. The attacker can then use stack pivoting at 0x20 to redirect the RSP, the stack pointer, which points to the top of the current stack frame, to point to the main chain and trigger the bug to escalate elevated privileges and run shell code. (Figure 5)

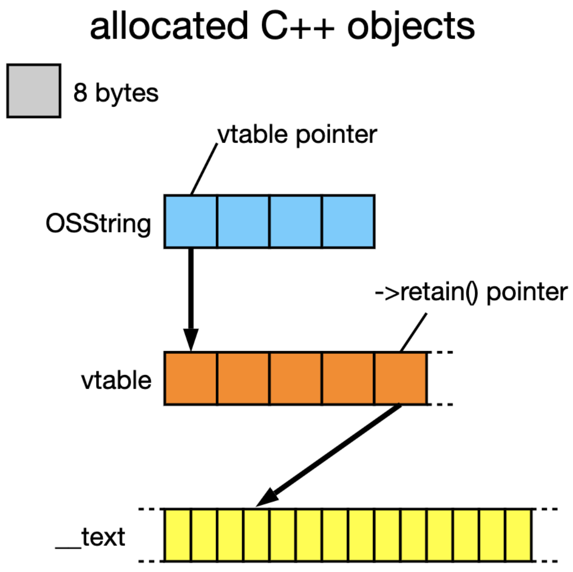


Figure 5: OSString Memory Allocation Visual [15]

3.1.3 Jailbreak

Once all three vulnerabilities have been used, there is some patches that are done to set up for the installation of the jailbreak. It starts with patching `setuid` to race KKP (Kernel Patch Protection), meaning it will run the patch to disable the protection before the KPP reaches the kernel. `Amfi_get_out_of_my_-way` and `Cs_enforcement_disable` will be patched to disable AMFI, a kernel module that enforces code-signing. Finally `Mac_mount` and `LwVM` will be patched to remount system partition to be readable and writable [12].

Now that the attacker has full access of the device and code-signing is disabled, the device can be jailbroken to install the spyware. The spyware includes "hooking", meaning the attacker can modify any class on the jailbroken device. The exploit installs Pegasus' dynamic libraries into the legitimate processes running on the device [16]. Dynamic libraries are collections of complied code that can be linked to and used by applications at run time. The exploit also hooks into the apps to be able to run surveillance and tracking software that will return back data to the attacker. Pegasus stays undetectable by blocking iOS system updates, clearing Safari's history and caches while also removing itself via self destruct mechanisms.

4 Impact

The Pegasus spyware has been used widely across the globe, and its impact on victims have been significant. Once the malware infects the phone through a simple text messages, it can allow attackers to access private data and communications without detection. This spyware uses advanced techniques to bypass security measures at both Apple's operating system and application level in order to access voice and audio calls. It also targets a wide range of messaging and communication apps such as Whatapps, Facebook, Viber, WeChat, and many more. It can also steal the victim's contact list, GPS locations, stored passwords for personal accounts, Wi-Fi networks and rounters [16]. (Figure 6)

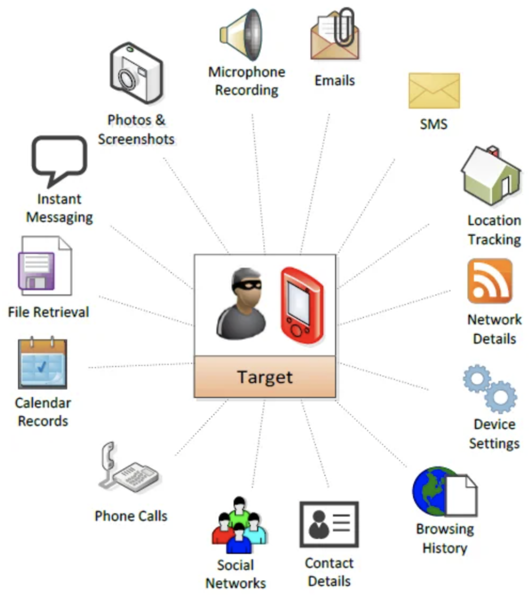


Figure 6: Visiual representation of what information can be stolen from user's device [1]

The most significant impact of the Pegasus incident was the loss of sensitive information, invasion of privacy, and decreased trust in security. The damage caused by the breach went beyond financial losses and affected people's trust in the security of their personal information. Studies have shown that more than 45 countries have been attacked and at least 10 countries are engaging in cross-border surveillance - several of which have been previously linked to human rights abuses involving spyware . According to reports, Pegasus has been purchased by Azerbaijan, Bahrain, Kazakhstan, Mexico, Morocco, Rwanda, Saudi Arabia, Hungary, India, and the United Arab Emirates for use in targeting activists, political dissidents, and journalists [17]. Pegasus is being used by countries with questionable human right records, and some of the targeting materials suggest possible political motivations for the use of this technology. This findings raise concerns about the proliferation of Pegasus and its potential impact on human rights.

Pegasus evolved into a zero-click exploit, a type of security vulnerability that allows an attacker to gain access to a device or system without requiring any interaction from the user. This means that the attacker can exploit the vulnerability without the user clicking on a malicious link, downloading a malicious file, or taking any other action that would indicate that their device has been compromised. The updated version of Pegasus used another vulnerability that impacted Apple's image rendering library. It was triggered by sending a PDF disguised as .gif iMessage attachment, and remotely triggering a heap buffer overflow in the ImageIO

JBIG2 decoder. According to Google Project Zero, it is one of the most technically and sophisticated modular pieces of surveillance, there has not been deep analysis regarding the code aspect of it [18]. Since Pegasus exploits zero-day vulnerabilites, there is no way to protect agaisnt it unless iOS releases updates to address the vulnerabilites.

5 Solution

5.1 Defensive mechanisms

Since the attack, Apple has patched all vulnerabilities that are known to public about Pegasus [19]. To prevent the vulnerabilities, both vendors and end users must take certain precautions. For example, the Pegasus spyware attack could have potentially been prevented by Apple implementing stricter checks on user accessible functions and bounds checking on JavaScript objects. From Apple's perspective, implementing strict control on objects in process memory and conducting careful code review can help prevent memory corruption exploits and other stack smashing attacks. While these fixes may seem straightforward in hindsight, it is a process that takes a lot of time and commitment.

From an end user and third party perspective, educating users on how to identify phishing texts and blocking similar texts from mobile service providers could also help prevent similar attacks. Apple was able to patch the exploit used in the attack, but some users who did not update their devices remained vulnerable. Users should be cautious when downloading and installing software from untrusted sources on the internet, as this can potentially introduce vulnerabilities into the system.

As a result of the Pegasus attacks, governments around the world should devote significant resources to detecting and preventing similar future attacks. The government should result to choosing different phones with more preventative security measures for high-ranking officials and government employees. Additionally, journalists and individuals who are reporting about certain governments should start using more secure applications to encrypt their personal information and protect privacy. Popular messaging apps should have look into adding more security to their apps as there is no doubt they are target to retrieving information about the user. This incident highlights the growing importance of cybersecurity as technology becomes increasingly integral to our daily lives.

5.2 Aftermath

Following the documented Pegasus attack, Apple took legal action against the NSO Group, alleging the company was responsible for the ongoing hacking of iOS devices [19]. Apple's lawsuit seeks to hold NSO Group accountable for these surveillance activities on researchers, journalists, activists, and government officials. Apple accuses the NSO Group's attack team of creating Apple IDs to send malicious data to victim's devices, allowing NSO Group or its clients to deliver and install Pegasus spyware without their knowledge. Apple is now seeking a permanent injunction to ban the NSO Group from any use of its software, services, or devices.

To emphasize the importance of human rights of foreign policy, the Biden's administration has since added the NSO Group into the Entity List for malicious cyber activities [20]. The entity list designation and blacklist prevents the NSO from importing any hardware or software from the United States due to the significant risk of national security and foreign policy. This could hinder NSO's ability to operate as an international company and impact its future business arrangement. Following this ban, The NSO Group's chief executive has stepped down and 100 employees are being let go as a result [21].

6 Conclusion

As mobile phones become more essential to our personal and professional lives, malicious attackers are developing sophisticated software that can run on victims' devices without their knowledge. These threats can operate without the victim being aware of the presence or intent of the attacker. Although Pegasus was first found in 2016, it is still unknown how long the vulnerability has surfaced the community before then. The NSO Group promises for secrecy and confidentiality but the increasingly amount of evidence is portraying that they are selling technology recklessly, neglecting human rights laws. While the NSO Group will continue to develop more exploits, it is important to learn from this experience so security researchers can find better ways to prevent and responsibly disclose vulnerabilities to Apple. This paper shows the importance of installing the latest software patches to keep our devices up to date and exercise vigilance with the security of our mobile devices such as being able to detect social-engineering.

References

[1] Bill Marczak and John Scott-Railton. The million dollar dissident - nso group's iphone zero-days used against a uae human rights defender. *The Citizen's Lab*, 2016.

[2] Amnesty International. Forensic methodology report: How to catch nso group's pegasus. `https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/`.

[3] NSO Group. About us. `https://www.nsogroup.com/about-us/`.

[4] Jody Serrano. Whatsapp head says new pegasus spyware investigation coincides with its findings from 2019 attack. *Gizmodo*, 2021.

[5] Daniel Bates. A refined content security policy. *Webkit*, 2016.

[6] Webkit. Tracking prevention in webkit. `https://webkit.org/tracking-prevention/`.

[7] Apple. Kernel framework. `https://developer.apple.com/documentation/kernel`.

[8] Rob Deans. Exploring ios's sandbox. *Medium*, 2017.

[9] The iPhone Wiki. Kernel aslr. `https://www.theiphonewiki.com/wiki/Kernel_ASLR`.

[10] Apple. Security. `https://developer.apple.com/documentation/security`.

[11] JD Rudie, Zach Katz, Sam Kuhbander, and Suman Bhunia. Technical analysis of the nso group's pegasus spyware. In *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 747–752, 2021.

[12] Lookout. Technical analysis of the pegasus exploits on ios. `https://info.lookout.com/rs/051-ESQ-475/images/pegasus-exploits-technical-details.pdf`.

[13] Mozilla. Object.defineproperties(). https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperties.

[14] jndok. Analysis and exploitation of pegasus kernel vulnerabilities (cve-2016-4655 / cve-2016-4656). https://jndok.github.io/2016/10/04/pegasus-writeup/.

[15] Siguza. tfp0 powered by pegasus. https://blog.siguza.net/cl0ver/.

[16] Lookout. Technical analysis of pegasus spyware. https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-technical-analysis.pdf.

[17] Amnesty International. Massive data leak reveals israeli nso group's spyware used to target activists, journalists, and political leaders globally. https://www.amnesty.org/en/latest/press-release/2021/07/the-pegasus-project/.

[18] Ian Beer and Samuel Groß of Google Project Zero. A deep dive into an nso zero-click imessage exploit: Remote code execution. https://googleprojectzero.blogspot.com/2021/12/a-deep-dive-into-nso-zero-click.html.

[19] Press Release. Apple sues nso group to curb the abuse of state-sponsored spyware. *Apple Newsroom*, 2021.

[20] Office of Public Affairs. Commerce adds nso group and other foreign companies to entity list for malicious cyber activities. *U.S. Department of Commerce*, 2021.

[21] ASH OBEL and AP. Nso group's hulio steps down as ceo of spyware firm, 100 employees let go. *The Times of Israel*, 2022.