2022 PRAGUE

28 - 30 September, 2022 / Prague, Czech Republic

EXPLOIT ARCHAEOLOGY: A FORENSIC HISTORY OF IN-THE-WILD NSO GROUP EXPLOITS

Donncha O'Cearbhaill

Amnesty International, Ireland

Bill Marczak

Citizen Lab, Canada

donncha.ocearbhaill@amnesty.org
bill@citizenlab.ca

https://t.me/learningnets

ABSTRACT

It is well understood that devices can be compromised by visiting malicious websites: in 2021, 24 of the 57 zero-day exploits reported to be used in the wild targeted browsers such as *Chrome* and *WebKit*. Less well understood is the world of remote 'zero-click' attacks, pushed to and executed automatically on devices anywhere in the world, via cloud messaging services such as *iMessage* and *WhatsApp*.

While a few examples have come to light in recent years, there are still open questions about how common the attacks are, and strategies to detect them. Over the past two years, *Amnesty International* and *Citizen Lab* have forensically examined hundreds of devices, and have developed techniques to hunt for signs of compromise in phone logs.

Our results indicate that NSO Group, a company that sells mobile attack tools to governments, has successfully fielded zero-click zero-day exploits against every major version of *iOS* starting from *iOS 10*, up to and including *iOS 14*.

We explain how we were able to reconstruct a partial timeline of NSO Group's zero-click activity on *iPhone*s through forensic analysis of compromised devices. We also provide some technical details about the attacks, based on analysis of indicators left behind, and discuss some of the challenges and limitations of this 'digital archaeology'.

INTRODUCTION

NSO Group has been a regular item in the news since its Pegasus mobile phone spyware was first discovered and publicly analysed in 2016, when it was used against Emirati activist Ahmed Mansoor [1]. NSO Group sells Pegasus exclusively to governments, and advertises that Pegasus can break into the latest phones with 'zero clicks', meaning that the target does not need to take any action (such as clicking on a link) to become infected. NSO Group is reported to have some 700 employees, and research by *Citizen Lab* and *Amnesty International* has shown Pegasus in use by dozens of governments around the world, making NSO Group the largest known private sector mobile threat actor as of 2021.

Though NSO Group says that Pegasus is designed to facilitate surveillance of criminals and terrorists, the Pegasus spyware is routinely abused to target members of civil society. Combined, *Amnesty International* and *Citizen Lab* have analysed more than 1,000 devices, leveraging trust relationships with civil society organizations, and found hundreds with traces of Pegasus. We have documented Pegasus abuse by suspected government operators in more than a dozen countries, including Mexico, El Salvador, Spain, Poland, Hungary, Morocco, UAE, Saudi Arabia, Bahrain, Jordan, India, Israel and Palestine, Kazakhstan, Thailand, Azerbaijan, and Togo. We found devices infected with Pegasus from 2014 through 2021, and with zero-click exploits from 2017 through 2021. This offers us a rich set of previously infected devices on which to conduct 'exploit archaeology', the process of understanding what capabilities were historically fielded to compromise devices.

While *Amnesty International* and *Citizen Lab* typically work independently, with separate technical processes, this report is a joint analysis of forensic artifacts we have collected over several years. We began preparing this report in March 2022 by comparing our respective forensic observations. Our goal in this analysis was to study the evolution of a 'top-tier' mobile threat actor's capabilities over time. How often does NSO Group field zero-click capabilities against the latest mobile devices? How long do these capabilities last? How vulnerable are users who scrupulously install the latest updates? What vectors or techniques do the exploits use? We approached this exploit archaeology from two different angles. First, we looked for artifacts and generic signatures left behind by the spyware itself, including process names used by the Pegasus spyware. Second, we correlated these artifacts with additional artifacts apparently left behind by exploits.

We are able to identify at least six distinct zero-click exploits deployed against *iOS* versions 10 through 14, as well as the well-known *WhatsApp* zero-click deployed against *Android* devices. While the *WhatsApp* zero-click was detected and patched by *Meta* in April 2019, we found the first public technical indications that it was active as early as July 2018. However, we show that not all 'zero clicks' have this sort of longevity. Despite the mystique behind NSO Group, we find that there appear to be significant gaps in time – in one case eight months – where the company's customers appear unable to field zero-click exploits against the latest *iOS* versions, and must resort to 'one -click' exploits via malicious links.

Our paper has the following contributions and findings:

- **Zero-click characterization**: We show that NSO Group customers have fielded six distinct zero-click exploits for *iOS* in the wild since at least 2017. This predates the first public domain research about the *iMessage* zero-click attack surface by at least two years. While details of some of these exploits have been previously published, we provide new details.
- **Observed update deficit**: Where we have sufficient data, we define a per-exploit metric that we call '*observed update deficit*'. The metric encapsulates how many days fully up-to-date devices were vulnerable-in-practice to the exploit while it was being deployed in the wild.
- **Possible *Project Zero* bug collision**: Through use of our observed update deficit metric, we demonstrate that *Google Project Zero*'s 2019 investigation of the *iPhone*'s zero-click attack surface may have significantly disrupted NSO Group's activities.
- ***WhatsApp* exploitation in 2018**: By correlating *Android* crash logs from a known Pegasus target with Pegasus Project records, we identify a likely case of the *WhatsApp* missed-call hack (CVE-2019-3568) from July 2018. *Meta* detected and patched the exploit in April 2019.

https://t.me/learningnets

Zero-click attacks have actively been deployed against human rights defenders and civil society for more than half a decade with little public analysis. By focusing on one particular firm, NSO Group, we will outline what we can learn from exploit archaeology. Continued close collaboration between human rights defenders, civil society investigators and private sector security researchers can help all sides gain a better understanding of the capabilities of private sector offensive actors.

SECTION 1: WHAT ARE ZERO-CLICK ATTACKS?

In this paper we focus on *remote* 'zero-click' attacks, which we define as any exploits or techniques that allow a remote attacker to achieve arbitrary code execution on a target device without requiring interaction from the user being targeted. A remote zero-click attack can potentially target any attack surface on the device that is reachable over any network to which the device is connected. In practice, we have seen threat actors favour messaging apps such as *iMessage* or *WhatsApp*, which are widely used, and generally automatically process complex data pushed to them by untrusted contacts. In the past, exploits have targeted *iMessage*'s attachment processing and *WhatsApp*'s cryptographic negotiation for video calls.

Of course, any remote exploitation vector targeting a modern device will need to chain multiple distinct vulnerabilities responsible for different stages of the attack. These stages include gaining initial code execution, escaping any sandbox or process isolation, escalating to gain kernel code execution, and bypassing code signature verification on the device. For many of the cases we outline in this paper, we classify exploits based on their initial mechanisms and not based on any subsequent stages. If an exploit chain's subsequent stages *change* during the lifetime of the initial mechanism, we consider these to be the 'same exploit'. Conversely, subsequent stages may be *reused* even if the initial mechanism changes, and we will consider these to be 'different exploits'.

In this paper, we do not focus on techniques such as browser exploits delivered automatically via network injection. While these attacks are technically zero-clicks, we do not consider them to be remote, as they require the attacker to have a vantage point on the network. Since at least 2011, spyware companies have advertised network injection appliances that can be installed in ISP backbones or local networks [2]. Researchers have observed these devices used in Turkey [3], Egypt [4] and Morocco [5], to deliver spyware-laden executable files and browser exploits. This technique may become increasingly popular for domestic targeting as states deploy necessary physical infrastructure, and as costs for remote zero-click capabilities increase.

A public history of 'zero-click' in the wild

The earliest discussion of zero-click attacks targeting mobile phones appears to be in a 2010 NSO Group proposal for Pegasus, which mentioned a zero-click capability targeting *BlackBerry* devices via the FOTA (firmware over-the-air) feature. However, it is unclear if this capability was truly remote, or whether it involved cooperation from the target's mobile provider.[1]

A 2013 Pegasus brochure describes an 'over-the-air' (OTA) installation technique: 'a push message is remotely and covertly sent to the mobile device.' The technique is said to require 'no cooperation or engagement of the target' [6]. This OTA installation technique may have made use of WAP Push messages.[2] Specifically, a WAP Push service-loading message can be used to force a target's device to open a web browser in the background and navigate it to a specified URL. The opened link could then automatically load a more traditional 'one-click' browser exploit.

Features such as WAP Push allow exploit developers to convert a one-click browser exploit into a zero-click capability, without the need for any additional exploits. Additionally, it is generally believed that browser exploits are easier to find than exploits for chat apps such as *iMessage* or *WhatsApp*, as a browser's attack surface is orders of magnitude larger.

Fortunately, most newer phones do not perform any action upon receipt of a WAP Push message. Nevertheless, we demonstrate that the browser remains an important component used by zero-click attacks. A number of novel zero-click exploits we have identified include a 'pivot-to-*WebKit*', where the focus of the initial zero-click phase appears to be solely on launching a browser instance and navigating it to a specified URL. Additional browser exploits then complete the exploitation of the target device.

Public documents and reporting show that other offensive actors beyond NSO Group have also been interested in developing and marketing zero-click capabilities against mobile devices. An undated brochure from German spyware company FinFisher describes a proposed zero-click capability targeting a 2016-2017 version of *iOS*, 10.0 [8]. Zerodium, an exploit broker, announced publicly in August 2017 that it would purchase *iPhone* remote zero-click exploit chains for up to 1.5 million USD [9]. Additionally, a 2019 report about Memento Labs, the successor company to Hacking Team, mentions *Android* spyware called 'Krait' that is designed to be delivered via a zero-click exploit [10].

Also, *Reuters* has published extensively on Project Raven, a hacking campaign linked to the UAE government that made use of zero-click exploits. Project Raven exploited at least two *iPhone* zero-click vulnerabilities, incorporated into spyware tools that the threat actors named KARMA and KARMA2. The vulnerability exploited by KARMA likely targeted *iOS 9*

[1] We located what appears to be a 2010 NSO Group Pegasus brochure on *VirusTotal*, SHA256: 8d7226d501d3964c9a51742a0ac3783f122bb0003c7d8822e2373b27fc8367bc.

[2] Some Pegasus samples for *Android* contain code to delete WAP Push messages from a device after successful infection, see [7].

https://t.me/learningnets

and was patched by *Apple* in September 2016. The vulnerability exploited by KARMA2 likely targeted *iOS 10* and was said to be patched around August 2017 [11], though we cannot identify any *iOS* update released in the month of August. One of the exploits used was reportedly developed and sold by a US-based company, Accuvant [12]. KARMA and KARMA2's reported targets included journalists at the *BBC* and *Al Jazeera* [13].

NSO Group's first post-WAP zero-click offering appears to have been its 'Pegasus 3' system, released in 2017 [14]. Subsequent NSO Group exploits have attracted media attention, including a 2019 *WhatsApp* exploit [15], and the 2021 ForcedEntry exploit [16].

SECTION 2: OUR EXPLOIT ARCHAEOLOGY METHODOLOGY

We began our exploit archaeology by identifying a set of common Pegasus identifiers, including items such as process names, that we attribute to NSO Group's Pegasus payload. As they are features of the payload and not the exploits, these identifiers are expected to be long-lived beyond a single exploit campaign. Identification and attribution of these artifacts to NSO Group is beyond the scope of this paper, but examples include [17] and [18], and the techniques used generally involve tracing the artifacts back to known ground truth attributed to NSO Group, including web servers from which the exploits were delivered.

We next look for exploit identifiers associated with the initial vector that are temporally adjacent or otherwise related to the common Pegasus identifiers. If suspected exploit identifiers are *not* related to the common Pegasus identifiers, we exclude them from our process. In this way, we exclude exploitation that may be associated with other threat actors.

Differentiation of exploits

For each exploit, we identify a specific test, which we believe corresponds with the exploit. Some specific tests may require gathering forensic data beyond what *Amnesty International* or *Citizen Lab* typically gather from target devices. For instance, some specific tests may require data only available after jailbreaking the device. In these cases, we sometimes extrapolate using a general assumption that only one NSO Group zero-click is actively deployed against a given *iOS* version at a given time.

For example, if we see compromises of *iOS 14* in 2021 that we link to NSO Group, and there were indications that Pegasus was delivered to the phone via some zero-click exploit, then we assume for the purposes of our analysis here that the Pegasus traces we observe are the result of the ForcedEntry exploit (see Section 8), even if we have not verified the specific presence of coretelephony cache files, which is our specific ForcedEntry test. We believe that this is a valid assumption, as we have verified our specific ForcedEntry test on a selection of phones with *iOS* versions 14.4 through 14.7.1 compromised between January and November 2021.

Determination of phone's iOS version on specific dates

To compute our observed update deficit metric, we must first identify which version of *iOS* the phone was running at the time of exploitation. While *iOS* does not have high-fidelity logs detailing the full update history of the device, there are a variety of other forensic artifacts that can be helpful in this regard. Nevertheless, the process of assigning specific *iOS* versions to dates can be somewhat difficult, particularly further back in time.

Limitations of forensic archaeology

We may not see all NSO Group exploits. NSO Group may maintain a suite of different Pegasus payloads after exploitation. It is not clear that we can detect all of their payloads, and some exploits might have been deployed in conjunction only with certain (more limited) payloads, rendering them invisible to us. This may cause us to miss certain exploits or underestimate certain NSO activity.

Our observed update deficit metric may also be an underestimate. The same exploits may be sold to different customers at different times, or authorized for use in different regions at different times. Thus, our observations might miss some of these early usages of an exploit. Additionally, transient regulatory or licensing concerns might impact territories authorized for deployment of a particular exploit.

Implicit in our metric is an assumption that exploitability of a target phone is a function of the version of its operating system or software. This seems a reasonable assumption where the exploit does not appear to rely on server-side behaviour, such as a malicious attachment delivered in an end-to-end encrypted fashion via *iMessage* (see Section 8). However, cases such as the *WhatsApp* exploit (see Section 9), in which the exploit is delivered via the control plane, can be mitigated without updates to client software.

Our methodology could theoretically conflate two different exploits. While we generally assume that there is a pre-exploit fingerprinting step before the exploit is launched, there may be cases where pre-exploit fingerprinting is not possible or desirable. Thus, multiple NSO exploits may be fired at the device simultaneously or quickly in sequence. This could cause cases where temporal correlation leads us to incorrectly link forensic identifiers from two distinct NSO Group exploits together.

https://t.me/learningnets

There is an inherent limitation in attempting to understand disruptions to NSO Group, such as one we describe in Section 5. Just as NSO Group attempts to operate silently, some defenders may attempt to disrupt NSO Group and may not disclose their efforts. For example, it appears that the Government of Turkey may have observed successful Pegasus attacks in 2019 and blocked some Pegasus installation websites and command-and-control servers [18]. Furthermore, third-party infrastructure vendors sometimes become aware of NSO Group activity [19] and take steps to ban the activity from their platforms. Because we cannot be aware of the full spectrum of efforts to disrupt NSO Group, a visibility bias may lead us to misattribute observed disruptions.

SECTION 3: IOS 10 GOLDENGATE IMESSAGE EXPLOIT

First observed deployment	5 July 2017 (successful)
Last observed deployment	December 2017 (successful)
First non-vulnerable version	*iOS 10.3.3* or earlier
Total observed update deficit	Insufficient data
Specific test	Presence of '`GoldenGate`' in historical data about running processes (e.g. DataUsage.sqlite. There may be related `com.apple.madrid` lookups related to the *iMessage* service in the `IDStatusCache` file.

The earliest Pegasus zero-click attack we observed occurred in July 2017 against *iOS 10*. It is difficult to identify the particularities of the attack vector due to the age of the attack but we can infer that the exploit delivery involved *iMessage*.

An attacker-controlled *iCloud* account was looked up by the targeted device shortly before successful Pegasus infection. The attacker email address was logged under the *iMessage* (com.apple.madrid) service in the '`com.apple.identityservices.idstatuscache.plist`' file. We will refer to this log file as the '`IDStatusCache`' file.[3]

Our specific test for this exploit is the presence of the process name '`GoldenGate`' in the *iPhone*'s '`com.apple.osanalytics.addaily.plist`' or '`DataUsage.sqlite`' file, followed by an additional Pegasus process name such as '`pcsd`'.

We have not observed a '`GoldenGate`' process more recently than September 2017, although some GoldenGate targets showed signs of related successful compromises until December 2017. It is unclear whether this vulnerability was patched by *Apple* or otherwise became inoperable.

One individual was compromised with this zero-click attack multiple times in the summer of 2017 until they upgraded to *iOS 10.3.3*. After the upgrade, the operator switched to sending one-click Pegasus exploit links over SMS. The evidence suggests this vector was either patched or not reliable against later versions of *iOS 10*.

SECTION 4: IOS 11 THUMPER EXPLOIT

First observed deployment	January 2018 (attempted), February 2018 (successful)
Last observed deployment	December 2018 (successful), January 2019 (attempted)
First non-vulnerable version	No successful attack observed targeting *iOS 12*
Total observed update deficit	Insufficient data
Specific test	Presence of '`com.apple.thumper`' lookup for malicious *iCloud* accounts in the `IDStatusCache` file.

During our forensic investigations we observed that suspicious *iCloud* accounts were interacting with Pegasus-infected *iPhones* using the '`com.apple.thumper`' service, which appears to be the internal *Apple* name for the Wi-Fi Calling (VoWiFi) feature on *iOS*.

The earliest presence of malicious Thumper lookups in the '`IDStatusCache`'[4] file began in mid-January 2018 and continued until at least January 2019. The last successful exploitation observed using this technique occurred in December 2018 targeting an *iOS 11* device.

Interestingly, the Wi-Fi Calling feature on *iOS* is configurable by the mobile carrier. Some carriers may not enable the use of Wi-Fi calling on their networks. It is unclear whether an operator configuration would limit the operation of the Pegasus VoWiFi exploit.

[3] The `IDStatusCache` file can contain historical records for accounts interfacing with a particular phone. NSO Group creates compartmentalized infrastructure such as domains and email addresses for each customer. The identification of the same Pegasus-linked *iCloud* account on multiple targets can be used to determine that all were targeted by the same Pegasus operator or customer. More information about this file is available at [17].

[4] The `idstatuscache.plist` file on *iOS* records when the *Apple* device interacted with other *iCloud* accounts. `/private/var/mobile/Library/Preferences/com.apple.identityservices.idstatuscache.plist`

https://t.me/learningnets

The VoWiFi calling protocol uses the SIP/SDP standard to negotiate a call connection between the two parties. In a later exploit we will see that NSO Group successfully exploited a flaw in the *WhatsApp* VOIP stack to deliver a different zero-click attack. One hypothesis is that NSO Group was able to find and exploit an earlier flaw in the VOIP stack used by *iOS*.

The VoWiFi calling feature also allows calling from non-cellular devices (such as laptops) that are connected to the same *iCloud* account as the cellular device. The VoWiFi protocol requires communication between the cellular device, the *iCloud* services and the mobile carrier, which opens a complex attack surface for potential exploitation by a remote attacker.

SECTION 5: IOS 12 IMESSAGE EXPLOIT

First observed deployment	March 2019
Last observed deployment	July 2020
Total observed update deficit	75 days
Earliest exploited version	12.1.4
First non-vulnerable version	Likely mitigated in *iOS 12.4.1*
Specific test	Presence of the file `/private/var/root/Library/Preferences/roleaccountd.plist`. This file is available in an *iOS* backup in the RootDomain. This file is written once upon first-run of the exploit on the device.

NSO Group successfully deployed a reliable zero-click exploit targeting *iOS 12* from 14 March 2019 at the latest. The exploit was successfully deployed as a zero-day against *iOS 12.1.4* in late March 2019. Extensive attacks using this exploit were observed targeting vulnerable devices until at least July 2020.

Version	First known exploitation of version	Next iOS version released	Observed update deficit
iOS 12.1.4	2019-03-22	2019-03-25	4 days
iOS 12.2	2019-04-18	2019-05-13	26 days
iOS 12.3.1	2019-06-10	2019-07-22	43 days
iOS 12.4	2019-08-25	2019-08-26	2 days

This attack appears to have been highly reliable. Artifacts indicating successful Pegasus infection were observed seconds after the attack was initiated against the target device.

Records from IDStatusCache files show that Pegasus-linked *iMessage* accounts interacted with the target device shortly before successful exploitation in many cases. This strongly suggests that this particular attack was again delivered over *iMessage*. A distinctive `roleaccountd.plist` file was dropped by Pegasus during the first successful use of this exploit against a target device.

The exploit shows evidence of the pivot-to-*WebKit* technique. Process database records indicate that the *Apple Podcast* app was often launched seconds after the initial *iMessage* connection from the attacker. This attack continued to be deployed against still-vulnerable versions of *iOS 12* until at least summer 2020. In July 2020 the attacker switched from launching the *Apple Podcast* app to instead opening the *Apple Music* app during the exploitation process.

A high-profile Azerbaijani journalist was targeted with this exploit in July 2020. A jailbreak of the journalist's device revealed a web request cache file tied to *Apple Music* [17]. This cache file contained records of HTTP requests to a previously identified Pegasus infection domain.

The URL extension suggests that an HTML-based browser exploit was loaded inside the *Apple Music* process space. The infection URL also shows clear structural similarities to URLs identified in other NSO Group zero-click attacks using different attack vectors.

Pegasus URL loaded by Apple Music
`https://`**`[RANDOM_SUBDOMAIN]`**`.php78mp9v.opposedarrangement[.]net:37`**`XXX`**`/`**`[uniqueid]/stadium/`**`pop2.html?key=501_4&n=7`
`https://`**`[RANDOM_SUBDOMAIN]`**`.php78mp9v.opposedarrangement[.]net:37`**`XXX`**`/`**`[uniqueid]/stadium/`**`pop2.html?key=501_4&n=7`

https://t.me/learningnets

The case of this Azerbaijani journalist also reveals information about NSO Group's exploitation priorities. The journalist was using an *iPhone 6*, which was not supported by *iOS 13*. The individual was exploited very extensively with Pegasus for 17 months until they upgraded to *iOS* version 12.4.7, which included the *iOS 12.4.1* patch. Their device remained on an *iOS 12* version for the next year and was not successfully exploited again. This case suggests that NSO Group did not prioritize developing a new zero-click capability against the older *iOS 12* release after the initial *iOS 12* vulnerability was patched.

Did a bug collision disrupt hacking of civil society?

In July 2019 *Google Project Zero* disclosed a number of remotely exploitable vulnerabilities in *iMessage* involving the deserialization of `NSKeyedUnarchiver` objects when parsing *iMessage* payloads. The disclosures were the result of a zero-click exploit chain *built* by *Project Zero* in order to test *iMessage*'s exploitability, rather than one discovered in the wild. While *Apple* introduced mitigations or fixes for many of these specific vulnerabilities in *iOS 12.4*, one of the underlying design flaws which allowed the deserialization child classes during NSUnarchiving [20] was not mitigated until *iOS 12.4.1*, and was not fully fixed until *iOS 13.2* [21].

We did not observe any successful use of NSO Group's *iOS 12 iMessage* exploit chain against any devices running *iOS* version 12.4.1 or higher. Furthermore, NSO Group's initial *iOS 13* zero-click exploit (see Section 6) appears to have been fixed in *iOS 13.2*, and this fix appears to have preceded a period of at least eight months during which we can identify no zero-day, zero-click exploits fielded by NSO Group customers (see Section 7).

Though it is hard to be sure, as we are not in possession of NSO Group's *iOS 12* and *iOS 13* exploits, this apparent disruption suggests that researchers at *Project Zero* may have independently identified a remote attack surface under active exploitation by a mercenary threat actor. By identifying and disclosing these vulnerabilities to *Apple*, *Google Project Zero* may have disrupted NSO Group's exploit pipeline, and in doing so, may have disrupted NSO Group customers' ongoing global attacks against journalists and civil society.

This is an indication that offensive security research of the type undertaken by *Project Zero* may provide direct safety benefits to end-users by blocking real-world attacks. It also demonstrates the value of vulnerability root-cause analysis. The hardening of high-risk or exploitable design features can disrupt multiple offensive exploit chains and can require significant investment from threat actors to find new exploitable attack surfaces.

SECTION 6: IOS 13.1.3 PHOTOSTREAM EXPLOIT

First observed deployment	29 October 2019
Last observed deployment	May 2020 (successful), September 2020 (attempted)
Total observed update deficit	10 days
Earliest exploited version	*iOS 13.1.3*
First non-vulnerable version	*iOS 13.2*
Specific test	Evidence of `com.apple.mediastream.mstreamd` *WebKit* cache files containing Pegasus domain names (successful exploitation). Evidence of `com.apple.private.alloy.photostream idstatuscache` lookups containing Pegasus email addresses (targeting attempt).

NSO Group continued to develop zero-click capabilities against *iOS 13* devices following the suspected patching of the *iOS12 iMessage* chain in *iOS 12.4.1*. Our forensic evidence shows that NSO Group pivoted from *iMessage* to a different attack surface: the *iPhone*'s built-in functionality for photo sharing.

Version	First known exploitation of version	Next iOS version released	Observed update deficit
iOS 13.1.3	2019-10-18	2019-10-28	10 days

Beginning in late October 2019, we observed lookups for suspicious *iCloud* email addresses in the `IDStatusCache` file under the `com.apple.private.alloy.photostream` service. The `photostream` service is responsible for the shared album feature in *Apple Photos*, where a user can create a shared photo album and invite other *iCloud* users to view or contribute to the shared album.

After the initial `photostream` lookup, network traffic from the `com.apple.mobileslideshow` app was recorded in the `DataUsage` database, often for the first time. In case of successful exploitation, we observed traces associated with the Pegasus '`bh`' process on the phone. Our research has repeatedly observed '`bh`' process records following successful one-click browser exploits.

With this Photostream attack we again see evidence of the pivot-to-*WebKit* technique. After apparently delivering an exploit via the `photostream` service, the exploit launched a *WebKit* instance in the `com.apple.mediastream.mstreamd` process.

https://t.me/learningnets

The *WebKit* instance then fetched JavaScript scaffolding from a known Pegasus infection domain. The scaffolding was fetched from `/[uniqueid]/stadium/goblin`. After performing tests to identify the local device model, the scaffolding fetched the *WebKit* exploit from `/[uniqueid]/stadium/eutopia`.

```
var t = document.createElement("div");
Object.defineProperty(t, "id", {
    get: () => {
        pi("555");
        window.location = "https://[redacted].apiweb248.theappanalytics.com:25[redacted]/[redacted]/stadium/eutopia"
    }
});
console.log(t)
```

We recovered the JavaScript scaffolding from an infected phone. The scaffolding includes a Pegasus infection domain name that we previously identified. The browser exploit was downloaded from a URL containing path `/[uniqueid]/stadium`. This URL path was also observed in the *iOS 12 iMessage* attack.

Apple introduced Lockdown Mode in *iOS 16*, which aims to protect users most at risk from sophisticated targeted attacks such as Pegasus. As part of the mitigations, *Apple* has disabled the shared albums feature in *Apple Photos* and new shared album invitations will be blocked while Lockdown Mode is active.

SECTION 7: IOS 13.5.1 IMTRANSCODERAGENT EXPLOIT (KISMET)

First observed deployment	July 2020
Last observed deployment	December 2020
Total observed update deficit	14 days
Earliest exploited version	*iOS 13.5.1*
First non-vulnerable version	Likely unexploitable in *iOS 14* due to introduction of the Blastdoor sandbox
Specific test	Particular contents of `IMTranscoderAgent.plist` file

Our exploit archaeology shows a zero-click campaign using an *iOS13 iMessage* exploit with deployments between July 2020 and December 2020. We characterize this exploit by traces that it leaves behind in the `IMTranscoderAgent.plist` file on the phone, which is visible in an *iTunes* backup.

We identified two periods during which the exploit was widely deployed as a zero-day. We saw deployments against *iOS 13.5.1* beginning on 5 July 2020. This was the latest available version of *iOS* for 11 days, through 15 July 2020, when *iOS 13.6* was released. Starting on 14 September 2020, we saw deployments of the exploit against *iOS 13.7*, which was the latest version for three more days. We saw no deployments against *iOS 13.6* when it was the latest *iOS* version, though we are unsure why. We do not believe that the issue is due to a lack of visibility, as forensics showed the exploit used as a zero-day by multiple suspected customers, including customers we suspect are linked to Bahrain, the UAE, Saudi Arabia, El Salvador and Spain. We compute the observed update deficit below.

Version	First known exploitation of version	Next iOS version released	Observed update deficit
iOS 13.5.1	2020-07-05	2020-07-15	11 days
iOS 13.7	2020-09-14	2020-09-16	3 days

Deployment of the exploit involved sending malicious attachments to a target *iPhone* via *iMessage*. These attachments were processed by `IMTranscoderAgent`, the *iMessage* component that handles generation of previews for media in attachments. Somehow, the malicious attachments appear to have coaxed `IMTranscoderAgent` to automatically open a *WebKit* instance in the background, and navigate to a Pegasus infection URL.

The exploit consisted, broadly, of two different types of attachments. Type A attachments were a JPEG gadget of 2,393 bytes whose sole purpose appeared to be causing a trivial crash in `IMTranscoderAgent`, and Type B attachments were pairs of attachments of varying sizes that we suspect contained malicious code. Before the exploit was fired, 25 Type A attachments were delivered to the device, in order to cause 25 trivial crashes in the `IMTranscoderAgent` process, thus triggering a crash reporter rate-limit and disabling further crash reporting for this process. When the exploit was fired, each pair of Type B attachments was preceded with a Type A attachment, likely to ensure that `IMTranscoderAgent` was freshly initialized to a predictable state.

Every Type A attachment caused the following trivial crash, which we believe is essentially immaterial to the exploit, in the same way that ForcedEntry's PSD gadget turned out to be immaterial to that exploit (see Section 8).

https://t.me/learningnets

```
Thread 1 name: Dispatch queue: com.apple.IMTranscoderPreviewGenerationQueue
Thread 1 Crashed:
  0: CoreFoundation 0x1803a6b48 _CFDataGetLength + 16
  1: ColorSync      0x1871e13c4 _copy_description_from_DSCMTag + 44
  2: ColorSync      0x1871e1ca4 _ColorSyncProfileCopyASCIIDescriptionString + 636
  3: ColorSync      0x1871e3ab4 _ColorSyncVerifyAdobeRGBData + 284
  4: CoreGraphics   0x1873d95a8 _CGColorSpaceCreateWithICCData + 344
  5: ImageIO        0x180de1fa0 _CGColorSpaceCreateWithCopyOfData + 56
  6: ImageIO        0x180db93b8 __ZN19AppleJPEGReadPlugin10initializeEP13IIODictionary + 5844
  7: ImageIO        0x180cf0908 __ZN13IIOReadPlugin14callInitializeEv + 220
  8: ImageIO        0x180d58a1c __ZN20IIO_Reader_AppleJPEG17initImageAtOffsetEP13CGImagePluginmmm + 172
  9: ImageIO        0x180c62390 __ZN14IIOImageSource13makeImagePlusEmP13IIODictionary + 936
 10: ImageIO        0x180c61bcc __ZN14IIOImageSource28getPropertiesAtIndexInternalEmP13IIODictionary + 68
 11: ImageIO        0x180c61b44 __ZN14IIOImageSource21copyPropertiesAtIndexEmP13IIODictionary + 16
```

We identified two instances of the following crash from one subtype of Type B attachments. No other Type B attachments appear to have caused crashes:

```
IMTranscoderAgent: (CoreFoundation) +[NSKeyedUnarchiver
bserverWithCenter:queue:name:object:block:]: unrecognized selector sent to class 0x20af6b4b0
```

The selector specified in the above log message is missing the 'o' letter in the identifier `observerWithCenter`. We are unsure what caused this, but it could be due to a memory offset error.

Eventually, an instance of Type B attachments caused `IMTranscoderAgent` to open a *WebKit* instance and download multiple items including: a 51KB initial payload and a 528KB subsequent payload. Approximately ten seconds after the 528KB payload was downloaded, we again saw traces associated with the Pegasus `bridgehead` process ('`bh`') on the phone. We show log excerpts below to illustrate this flow:

Date/time	Entry
23:53:43.001647	imagent: (IMFoundation) [Warning] OOP preview generation failed in imagent with error (null)
23:53:43.047130	IMTranscoderAgent: (UIKitCore) [com.apple.UIKit:DisplayConfigurationFetch] Failed to load a display context, even though we are not a headless app.
23:53:43.071312	IMTranscoderAgent: (WebKit) [com.apple.WebKit:ProcessSuspension] 0x1330041c0 - [PID=0, throttler=0x131f05598] ProcessThrottler::Activity::Activity: Starting background activity / 'WebProcess initialization'
23:53:47.053146	com.apple.WebKit.Networking: (libnetwork.dylib) [com.apple.network:] [**REDACTED** Hostname#**REDACTED**:13553 tcp, bundle id: com.apple.imtranscoderagent, pid: **REDACTED**, url hash: **REDACTED**, tls] cancelled [C1.1 **REDACTED** **REDACTED**:61844<->IPv4#**REDACTED**:13553] Connected Path: satisfied (Path is satisfied), interface: en0, ipv4, ipv6, dns Duration: 3.693s, DNS @0.001s took 0.314s, TCP @0.321s took 0.197s, TLS took 0.454s bytes in/out: 51127/1328, packets in/out: 50/3, rtt: 0.154s, retransmitted packets: 0, out-of-order packets: 0
23:53:47.272542	com.apple.WebKit.WebContent: (Metal) compiling shader
23:53:47.645965	IMTranscoderAgent: (WebKit) [com.apple.WebKit:PerformanceLogging] 0x12ff4a568 - WebProcessPool::handleMemoryPressureWarning:
23:53:57.044981	com.apple.WebKit.Networking: (libnetwork.dylib) [com.apple.network:] [**REDACTED** Hostname#**REDACTED**:13553 tcp, bundle id: com.apple.imtranscoderagent, pid: **REDACTED**, url hash: **REDACTED**, tls] cancelled [C9.1 **REDACTED** **REDACTED**:61852<->IPv4#**REDACTED**:13553] Connected Path: satisfied (Path is satisfied), interface: en0, ipv4, ipv6, dns Duration: 4.247s, DNS @0.000s took 0.000s, TCP @0.002s took 0.149s, TLS took 0.337s bytes in/out: 528717/1376, packets in/out: 490/3, rtt: 0.145s, retransmitted packets: 0, out-of-order packets: 2
23:54:06.389689	(spyware process '`bh`' launched)

This particular exploit no longer worked against *iOS 14*, perhaps because of *Apple*'s BlastDoor mitigation, though it was deployed against out-of-date devices through at least December 2020. Like NSO's previous exploit against *iOS 13*, this exploit appeared to be deployed as a zero-day for a very short period of time, demonstrating that the *iOS 13* era was likely an era of significant challenge for NSO.

https://t.me/learningnets

SECTION 8: IOS 14 FORCEDENTRY/MEGALODON EXPLOIT

First observed deployment	January 2021
Last observed deployment	November 2021
Total observed update deficit	131 days
Earliest exploited version	*iOS 14.4*
First Non-vulnerable version	*iOS 14.8*
Specific test	Evidence of `com.apple.coretelephony` cache files containing Pegasus domain names (14.6 and prior). No specific test for 14.7 and higher.[5]

Our exploit archaeology identified a long-lived *iOS 14* exploit, deployed from January through November 2021. The exploit was deployed against at least four *iOS* versions while these versions were the latest available. *Amnesty International* and *Citizen Lab* independently observed this *iOS 14* zero-click vulnerability being used in the wild beginning in early 2021. *Amnesty International* refers to this exploit as Megalodon and *Citizen Lab* refers to this exploit chain as ForcedEntry.

Version	First known exploitation of version	Next iOS version released	Observed update deficit
iOS 14.4	2021-02-10	2021-03-08	26 days
iOS 14.4.2	2021-04-07	2021-04-26	27 days
iOS 14.6	2021-05-31	2021-07-19	50 days[6]
iOS 14.7.1	2021-08-17	2021-09-13	28 days

The exploit's extraordinary longevity (there were approximately 131 days in 2021 that your phone was vulnerable even if you scrupulously installed all latest updates) may be attributable to its novel architecture.

As in Section 7, the exploit consisted, broadly, of two different types of attachments. Type C attachments were an *Adobe* PSD gadget of 748 bytes, and Type D attachments were *Adobe* PDF files of varying sizes containing JBIG2 streams. Before the exploit was fired, crash reporting was disabled with 25 Type C attachments. When the exploit was fired, each Type D attachment was preceded with a Type C attachment, likely to ensure that `IMTranscoderAgent` was freshly initialized to a predictable state. We observed instances where a Type D attachment arrived before a Type C attachment, or the same attachment arrived twice.

A *Google Project Zero* write-up on a copy of the exploit captured in March 2021 revealed a novel custom Turing machine architecture implemented using JBIG2 segment commands [22]. Multiple JBIG2 segment commands are chained in the exploit to implement a sequence of logical bit operations on arbitrary memory. The logical bit operations are then abstracted to create a higher level computer architecture to search memory and perform arithmetic operations.

This computing environment is used to launch a sandbox escape and gain code execution outside the `IMTranscoderAgent` sandbox. *Google Project Zero* published a separate write-up of the ForcedEntry sandbox escape [23]. Interestingly, the sandbox escape uses only logic bugs, avoiding the need for further memory corruption bugs, and bypassing certain memory-corruption mitigations such as Pointer Authentication Codes (PAC) and Memory Tagging.

After escaping the sandbox, the ForcedEntry exploit downloaded a second-stage payload from a Pegasus infection server via cloud fronting services, including *Amazon*'s *CloudFront*. We recovered `Cache.db` files from multiple devices targeted or compromised by ForcedEntry at the path `/private/var/wireless/Library/Caches/com.apple.coretelephony/Cache.db`, which included the URL and headers of the HTTP GET request [17].

Example Pegasus Installation URL loaded by ForcedEntry

```
https://d38j2563clgblt.cloudfront[.]net/[uniqueid]//stadium/
megalodon?m=iPhone9,1&v=18C66
```

Prior to *iOS 14.7*, ForcedEntry temporarily stored an encrypted ~250KB second-stage payload in the `fsCachedData` subfolder under the `com.apple.coretelephony` cache directory. Forensic analysts cannot access this location of the phone unless the phone is jailbroken. This payload is encrypted with AES 128. The code launched by the `.gif` files decrypts this second-stage payload via the AES128DecryptWithPassword method in `OpusFoundation.framework` using

[5] NSO Group may have tweaked ForcedEntry to avoid generating these cache files following the publication of the Pegasus Project.
[6] We saw no deployment of ForcedEntry between 2021-07-12 and 2021-07-19, which may be related to the lead-up to the publication of the Pegasus Project on 2021-07-19. Thus, *iOS 14.6* devices *may not* have been vulnerable in practice for these seven days, and a more conservative estimate for this value may be 43 days.

https://t.me/learningnets

a key hard coded in the `.gif` files. The key appears to be randomized in each deployment of the exploit, meaning that decrypting the payload requires *both* a copy of the `.gif` files and the item from the `com.apple.coretelephony` cache directory. The second-stage payload has not yet been publicly described or analysed.

Once the second stage was successfully decrypted and loaded, a subsequent Pegasus process was launched. In one instance, this process was called `gatekeeperd`. On one phone, we identified a `/private/var/root/Library/Caches/gatekeeperd/Cache`.db file that records a single HTTP request to the same *CloudFront* domain. This URL request may represent a callback to the Pegasus installation server confirming that the installation had completed successfully.

Example Pegasus URL loaded by ForcedEntry second stage
`https://d38j2563clgblt.cloudfront.net/`**`[uniqueid]//stadium/`**`wizard/01-00000000`

We observed an early ForcedEntry variant on 15 January 2021 that appeared to involve a pivot-to-*WebKit after* ForcedEntry exploitation. The initial stages of the attack appear identical. After the sandbox escape, the `com.apple.coretelephony` process again downloaded the second-stage payload from the Pegasus installation server.

This second stage was launched with the process name '`payload`' rather than a more innocuous-sounding name, as is typical, suggesting the attack may still have been in development at the time. This `payload` stage subsequently launched a process called '`bh`', that we previously observed only during instances of exploitation involving *WebKit*.

We did not observe a '`bh`' process in any subsequent variants of the ForcedEntry attacks after January 2021. The presence of the '`bh`' process indicates this attack used a pivot-to-*WebKit* step to complete the later stages of exploitation at this time. Later attacks show no evidence of '`bh`', suggesting the pivot-to-*WebKit* stage was either replaced outright, or that NSO Group decided to change the name.

SECTION 9: 2018-2019 ANDROID WHATSAPP EXPLOIT

In April 2019 *WhatsApp* discovered and fixed a zero-click vulnerability (recorded as CVE-2019-3568) in *WhatsApp for Android* video calling functionality, which was actively exploited by NSO Group Pegasus customers. *WhatsApp* also filed a lawsuit against NSO Group for the targeting of its systems and customers. *WhatsApp* later notified more than 1,400 targets that it says were targeted with the exploit between March and April 2019. However, through exploit archaeology, we believe we have uncovered a case of this zero-click exploit being used in July 2018, almost nine months before *WhatsApp* fixed the vulnerability.

On the phone of a dissident's family member, we identified two crashes four minutes apart on a single day in July 2018. The crashes were segfaults in the *WhatsApp* app. The former was a segfault in the main thread, and the latter was a segfault in the 'VoIP Signaling' thread.

The individual had previously been targeted with Pegasus one-click attacks against their *Android* device. Pegasus Project data shows that the target's number was selected for targeting three times on the day of the crashes, including one instance that occurs 59 seconds before the first crash.

We additionally located an old copy of *WhatsApp*'s `com.whatsapp_preferences.xml` file on the phone, which we excerpt below. The file indicates that a video call from `35796005384@s.whatsapp.net` ended two seconds after the first crash.

```
<string name="previous_call_peer_id">35796005384@s.whatsapp.net</string>
<boolean name="previous_call_video_enabled" value="true" />
<long name="previous_call_end_time" value="[two seconds after first crash]" />
<string name="version">2.18.203</string>
```

In the latter crash's tombstone file, we found indications of the same (ASCII-encoded) identifier, `35796005384@s.whatsapp.net`, in registers D21 - D23 of the '`Messages Async`' thread.

```
d20 0800000800000800  d21 3735330008080000
d22 3438333530303639  d23 73746168772e7340
```

We additionally found a fragment of the `call_log` table from an old copy of `msgstore.db` stored on the phone that records six incoming calls from `35796005384@s.whatsapp.net` around the time of the crashes.

WhatsApp's lawsuit against NSO mentions that attackers registered *WhatsApp* accounts used to send the malicious video calls using phone numbers bearing the country codes of at least six countries, including Cyprus [24]. *WhatsApp*'s lawsuit indicates that *WhatsApp* believes NSO Group conducted activities in relation to the attack as early as January 2018, though *WhatsApp*'s notifications only covered a two-week period in March and April 2019.

Because the first crash we identify is tightly correlated with Pegasus Project data, because both crashes involve the same Cypriot number, because *WhatsApp* mentioned Cyprus as one of the country codes used by the attack accounts, and because the crashes are otherwise consistent with CVE-2019-3568 (e.g. they occurred while processing video calls), we

https://t.me/learningnets

believe that the crashes are representative of July 2018 use of CVE-2019-3568 or a similar vulnerability against the target by an NSO Group customer.

Total time delta	Event
+0s	Selection in Pegasus Project data
+45s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table
+46s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table
+59s	Crash #1 (WhatsApp main thread)
+61s	`previous_call_end_time` for call for video call from `35796005384@s.whatsapp.net`
+287s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table
+288s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table
+308s	Crash #2 (WhatsApp 'VoIP Signaling' thread) indicating `35796005384@s.whatsapp.net` in registers of 'Messages Async' thread
+355s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table
+413s	Call from `35796005384@s.whatsapp.net` per fragment of `call_log` table

SECTION 10: BROWSER EXPLOITS

This paper has focused primarily on the zero-click exploit chains deployed by NSO Group for use by its customers. While zero-click vectors are highly prized, our forensic archaeology shows that NSO Group has also maintained an effective one-click capability targeting all major versions of *iOS* since at least *iOS 9*.

Browser exploits for *WebKit* or *Chrome* can target a large, complex and ever-changing attack surface. The scripting environment available in web browsers can also ease the exploitation process when compared with (semi-)blind zero-click attack surfaces. As outlined in Section 1, browser exploits can also be used as pseudo zero-click vectors when delivered using tactical network injection hardware (such as a malicious Wi-Fi network) or strategically from network injection equipment deployed upstream at an ISP.

In earlier sections we have also shown that browser exploits may be components of zero-click chains, where an initial zero-click attack surface is targeted before a pivot-to-*WebKit* to complete further exploitation and privilege escalation.

In most cases we do not have detailed information about the exploited vulnerabilities. Browser exploits interact with few services on the device, making forensic investigation more difficult. NSO Group also implements a 'clean-up' step, which deletes browser cache files and other artifacts that could reveal the attack vector [25].

The following table outlines the browser exploits we have observed from NSO Group in the wild. The targeted *iOS* version information is not exhaustive.

Time	Affected version	Artifacts
Before Aug 2016	Zero-day against iOS 9.3.3 (CVE-2016-4657)	*WebKit* vulnerability exploited as part of 'Trident' chain. Use-after-free in JavaScriptCore enginet [25]. JIT functionality was abused to load shellcode into read/write/execute memory during exploitation.
Feb 2018	iOS 10.3.3	Network traffic observed from process '`mount_nfs`' after one-click exploitation
April 2018	iOS 11.3	Pegasus process '`bh`' launched after one-click exploitation; `mediaremoted.plist` file dropped during exploitation
Feb 2019 - Sept 2019	iOS 12.1.2, iOS 12.3.1	Suspected vulnerability in *WebKit* `IndexedDB` code. Multiple empty `IndexedDB` databases were dropped on devices following network injections which redirected to a Pegaus installation domain [5]. `IndexedDB` use-after-free vulnerabilities have been exploited in the wild [26]. Very similar vulnerabilities have also been identified in the *Chrome* `IndexedDB` code.
May 2020	iOS 13.X	Suspected *WebKit* vulnerability exploited during PhotoStream zero-click attack (see Section 6)
July 2020	iOS 13.5.1	Suspected *WebKit* vulnerability exploited during *iMessage* `IMTranscoderAgent` zero-click attack (see Section 7).
Dec 2021	iOS 14.8.1	Unknown *WebKit* vulnerability exploited during successful one-click attack.

https://t.me/learningnets

CONCLUSION

Our investigations and forensic archaeology process have uncovered clear evidence that zero-click exploits have been abused in the wild for at least five years to target human rights activists and other members of civil society. The duration and scale of zero-click targeting is longer and more extensive than previously documented in public sources.

This research shows that a single company, NSO Group, was able to maintain an operational zero-click capability against all major *iOS* versions from *iOS 10* through *iOS 14*. We identified six distinct chains which enabled NSO Group customers to discreetly exploit targets extensively over years, at a global scale. The longevity of some zero-click chains likely reflects the security community's lack of visibility into mobile attacks.

There are few security solutions available to meaningfully analyse phones for traces for advanced targeted attacks [27]. Users often rely on their device manufacturers to find and patch exploit chains, but vendors may not be able to detect on-device targeting due to limits in visibility and telemetry. While some one-click exploitation may be detectable by a cautious user, or with network monitoring systems, there is a lack of fine-grained telemetry and logging needed to detect or block zero-click exploitation and other advanced attacks against mobile devices.

There are positive signs that defending against remote zero-click attacks may be more tractable than the wider field of computer security defence. Potential zero-click attack surfaces need to be remotely accessible, often must be exploited non-interactively without a scripting environment, and ideally also be silent or at least non-alerting to the target.

In July 2019, *Google Project Zero* independently discovered an *iMessage* zero-click attack surface that appears to have been actively exploited by NSO Group customers at the time. The fact that multiple researchers may have identified the same attack surface suggests the range of practically exploitable attack surfaces may be much smaller in the zero-click context. A systematic effort by device vendors and security researchers to enumerate, analyse and mitigate potential attacks against these attack surfaces has the potential to significantly reduce the scope of zero-click attacks.

Recent security engineering efforts by *Apple* with its opt-in 'Lockdown Mode' also provide a model for further mitigations [28]. Lockdown Mode adds significant mitigations which may have disrupted some of the zero-click chains outlined in this research. Two of the attack chains targeted users with malicious attachments sent over *iMessage*. These attacks may have been disrupted by Lockdown Mode, which by default blocks incoming messages from any unknown contact. Photo album sharing is also disabled, which may have prevented the attack outlined in Section 6.

Apple has also significantly reduced the browser attack surface in Lockdown Mode. The Pegasus *WebKit* exploit for *iOS 9* abused JIT optimizations to gain arbitrary code execution. This exploitation technique is no longer possible under Lockdown Mode as JIT is disabled. Other vendors have explored similar mitigations, such as the opt-in Super Duper Secure Mode in *Microsoft Edge* [29].

Lockdown Mode is an important acknowledgement that not all users have the same threat model and security needs. Our research with civil society shows that some individuals and communities are persistently targeted with advanced cyber-surveillance systems and zero-day exploits from multiple vendors over many years.

Major technology vendors should follow *Apple*'s lead and provide solutions that are tailored to the unique threats faced by civil society, including journalists and human rights activists because of who they are or the work that they do. These attacks can threaten the fundamental functioning of societies. The significant risks that these individuals face must not be dismissed as a rounding error in the security engineering efforts of major vendors.

Browser and operating system maintainers should provide opt-in enhanced mitigations and increased security telemetry to help at-risk individuals protect themselves from advanced targeted attacks. Technology companies can also play a crucial role in informing users about the potential attacks they face.

The harms caused by the mercenary surveillance industry cannot be solved by technology alone. Targeted threat notifications can enable victims to seek accountability and redress from surveillance companies and the governments that abuse these tools. Notifications should be as timely and descriptive as possible, to help users understand the threats they face and to find expert help to document and mitigate the attack.

The major technology vendors should continue to work closely with civil society researchers to help investigate and disrupt these surveillance campaigns. All vendors, including non-mobile vendors, have visibility into various stages of the attacks from companies such as NSO Group. By working closely, we can shorten the lifetime of targeted attack campaigns and help protect global civil society communities.

REFERENCES

[1] Marczak, B.; Scott-Railton, J. The Million Dollar Dissident NSO Group's iPhone Zero-Days used against a UAE Human Rights Defender. The Citizen Lab. August 2016. https://citizenlab.ca/2016/08/million-dollar-dissident-iphone-zero-day-nso-group-uae/.

[2] Remote Monitoring & Infection Solutions. FinFly ISP. https://wikileaks.org/spyfiles/files/0/297_GAMMA-201110-FinFly_ISP.pdf.

https://t.me/learningnets

[3] Kafka, F. StrongPity2 spyware replaces FinFisher in MitM campaign – ISP involved?. We Live Security. December 2017. https://www.welivesecurity.com/2017/12/08/strongpity-like-spyware-replaces-finfisher/.

[4] Marczak, B.; Dalek, J.; McKune, S.; Senft, A.; Scott-Railton, J.; Deibert, R. BAD TRAFFIC. Sandvine's PacketLogic Devices Used to Deploy Government Spyware in Turkey and Redirect Egyptian Users to Affiliate Ads?. The Citizen Lab. March 2018. https://citizenlab.ca/2018/03/bad-traffic-sandvines-packetlogic-devices-deploy-government-spyware-turkey-syria/.

[5] Amnesty International. Moroccan Journalist Targeted With Network Injection Attacks Using NSO Group's Tools. June 2020. https://www.amnesty.org/en/latest/research/2020/06/moroccan-journalist-targeted-with-network-injection-attacks-using-nso-groups-tools/.

[6] Guarnieri, C. NSO Pegasus. https://www.documentcloud.org/documents/4599753-NSO-Pegasus#document/p15/a437978.

[7] Lookout. Pegasus for Android Technical Analysis and Findings of Chrysaor. April 2017. https://info.lookout.com/rs/051-ESQ-475/images/lookout-pegasus-android-technical-analysis.pdf.

[8] Otto, G. How sloppy OPSEC gave researchers an inside look at the exploit industry. Cyberscoop. January 2019. https://www.cyberscoop.com/mobile-zero-days-lookout-shmoocon-2019-android-barracuda-ios-stonefish/.

[9] Goodin, D. Wanted: Weaponized exploits that hack phones. Will pay top dollar. Ars Technica. August 2017. https://arstechnica.com/information-technology/2017/08/wanted-weaponized-exploits-that-hack-phones-will-pay-top-dollar/.

[10] O'Neill, P. H. The fall and rise of a spyware empire. MIT Technology Review. November 2019. https://www.technologyreview.com/2019/11/29/131803/the-fall-and-rise-of-a-spyware-empire/.

[11] United States Department of Justice. Three Former U.S. Intelligence Community and Military Personnel Agree to Pay More Than $1.68 Million to Resolve Criminal Charges Arising from Their Provision of Hacking-Related Services to a Foreign Government. September 2021. https://www.justice.gov/opa/pr/three-former-us-intelligence-community-and-military-personnel-agree-pay-more-168-million.

[12] O'Neill, P. H. This US company sold iPhone hacking tools to UAE spies. MIT Technology Review. September 2021. https://www.technologyreview.com/2021/09/15/1035813/us-sold-iphone-exploit-uae/.

[13] Schectman, J.; Bing, C. American hackers helped UAE spy on Al Jazeera chairman, BBC host. Reuters. April 2019. https://www.reuters.com/investigates/special-report/usa-raven-media/.

[14] Harel, A.; Levinson, C. Kubovich, Y. Revealed | Israeli Cyber Firm Negotiated Advanced Attack Capabilities Sale With Saudis, Haaretz Reveals. Haaretz. November 2018. https://www.haaretz.com/israel-news/.premium-israeli-company-negotiated-to-sell-advanced-cybertech-to-the-saudis-1.6680618.

[15] Srivastava, M. WhatsApp voice calls used to inject Israeli spyware on phones. Financial Times. May 2019. https://www.ft.com/content/4da1117e-756c-11e9-be7d-6d846537acab.

[16] Kirchgaessner, S. Israeli spyware firm targeted Apple devices via iMessage, researchers say. The Guardian. September 2021. https://www.theguardian.com/technology/2021/sep/13/nso-group-iphones-apple-devices-hack-patch.

[17] Amnesty International. Forensic Methodology Report: How to catch NSO Group's Pegasus. July 2021. https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-nso-groups-pegasus/.

[18] Marczak, B.; Scott-Railton, J.; Al-Jizawi, N.; Anstis, S.; Deibert, R. The Great iPwn. Journalists Hacked with Suspected NSO Group iMessage 'Zero-Click' Exploit. The Citizen Lab. December 2020. https://citizenlab.ca/2020/12/the-great-ipwn-journalists-hacked-with-suspected-nso-group-imessage-zero-click-exploit/.

[19] Cox, J. Amazon Shuts Down NSO Group Infrastructure. Vice Motherboard. July 2021. https://www.vice.com/en/article/xgx5bw/amazon-aws-shuts-down-nso-group-infrastructure.

[20] Groß, S. (@5aelo). Twitter. September 2019. https://twitter.com/5aelo/status/1172534071332917248.

[21] Groß, S. Remote iPhone Exploitation Part 1: Poking Memory via iMessage and CVE-2019-8641. Google Project Zero. January 2020. https://googleprojectzero.blogspot.com/2020/01/remote-iphone-exploitation-part-1.html.

[22] Beer, I.; Groß, S. A deep dive into an NSO zero-click iMessage exploit: Remote Code Execution. Google Project Zero. December 2021. https://googleprojectzero.blogspot.com/2021/12/a-deep-dive-into-nso-zero-click.html.

[23] Beer, I.; Groß, S. FORCEDENTRY: Sandbox Escape. Project Zero. March 2022. https://googleprojectzero.blogspot.com/2022/03/forcedentry-sandbox-escape.html.

[24] Cox. J. WhatsApp/Facebook v. NSO Group. https://www.documentcloud.org/documents/6532441-WhatsApp-Facebook-v-NSO-Group.

https://t.me/learningnets

[25] Lookout. Technical Analysis of the Pegasus Exploits on iOS. https://info.lookout.com/rs/051-ESQ-475/images/pegasus-exploits-technical-details.pdf.

[26] Stone, M. CVE-2021-30858: WebKit use-after-free in IndexedDB. Google Project Zero. September 2021. https://googleprojectzero.github.io/0days-in-the-wild/0day-RCAs/2021/CVE-2021-30858.html.

[27] mvt-project / mvt. https://github.com/mvt-project/mvt/.

[28] Apple Newsroom. Apple expands industry-leading commitment to protect users from highly targeted mercenary spyware. July 2022. https://www.apple.com/newsroom/2022/07/apple-expands-commitment-to-protect-users-from-mercenary-spyware/.

[29] Norman, J. Super Duper Secure Mode. Microsoft Browser Vulnerability Research. August 2021. https://microsoftedge.github.io/edgevr/posts/Super-Duper-Secure-Mode/.

https://t.me/learningnets