APPLIED CRYPTOGRAPHY #1

Exploiting CVE-2016-4657 to Jailbreak the Nintendo Switch

April 2, 2017

Idan A.
Avraam I.
Netanel O.

Beware!

This article and the attached source code is meant for academic purposes only. We do not encourage any kind of malicious usage of the code and/or the knowledge shared in this document.

The Application

The Nintendo Switch™ is a gaming console released by the Nintendo Company in early 2017.

The device allows for playing video games both in the home environment & the outdoors. Games for this console are distributed region-free on both physical media (game carts) & as downloadable content.

The operating system running on the console is a domain-specific OS, created by Nintendo for the Switch based on the Linux kernel.

Outside of the built-in software developed by Nintendo, and the games distributed by certified partners; no external applications can be loaded on the console. A closed-garden practice is being used to ensure intellectual property is protected.

Although the Nintendo Switch provides internet connectivity for playing multi-player games, it does not ship with a dedicated web browser application. However, a hidden browser is present for a special occasion, detailed later in this document.

The Exploit

Hidden Web Browser

First, let us explain what a Captive Portal is. According to Wikipedia:

*“A **captive portal** is a ‘landing' web page, presented by a Layer 3 brand or Layer 2 Operator and shown to users before they gain broader access to URL or http-based Internet services”*

i.e. it is a login screen, built in HTML, which usually prompts the user to accept usage conditions, before allowing him to use the network to surf the web. Usage of Captive Portal is very common on public wireless networks, such as those available on hotels, restaurants & public transportation.

As mentioned before, the Nintendo Switch does not ship with a web-browser application, but it does have a hidden one, used for logging into Wi-Fi network which utilize a Captive Portal.

When a user tries to connect to such a wireless network, a minimal web-browser is launched, sending an **HTTP GET** request to “**http://conntest.nintendowifi.net/**”, expecting a response with HTTP header of “**X-Organization: Nintendo**”.

If the response does not meet this condition, the browser determines that the network is using a captive portal, and redirect to the portal’s landing page provided in the HTTP response header.

By running a proxy server on a separate machine, we can spoof the response received by the Switch’s browser and deceive it into treating the connectivity as captive-portal based and load our own HTML webpage.

All that is required to make this work, is to run the Burp Suite[1] program on the host machine, then set the proxy address on the switch to the host’s IP address. Running python’s **simple-http-server** allows us to server static files from the host file-system.

This mechanism is what allows us to load the bug activation code onto the Switch.

[1] Free/Commercial Software developed by “PortSwigger”.

The Bug

The "Common Vulnerabilities & Exposures" database described **CVE-2016-4657** as:

> *"WebKit in Apple iOS before 9.3.5 allows remote attackers to execute arbitrary code or cause a denial of service (memory corruption) via a crafted web site."*

The bug was first identified as an integral part of the Pegasus malware, alongside with **CVE-2016-4655** & **CVE-2016-4656**. It was rated 8.8/10 in the CVVS[2] v3 metric by the NVD[3].

This vulnerability in the WebKit web-browser engine, allows for read/write access to the entire memory section of the operating process and potentially even outside the process.

It is using a bug in way the engine handles memory segments referencing and garbage collection. It is written in JavaScript & can be ran silently as part of any website.

Although the CVE database lists the vulnerability as one in the Mobile Safari browser shipped as part of iOS 9.3, it can affect any browser using the WebKit engine in versions older than 9.3.5.

It appears the Nintendo Switch is running an old version of WebKit, that is vulnerable to CVE-2016-4657. That allows hackers & modders to run homebrew code, which both hurts the intellectual property of Nintendo & Co. (Running unlicensed apps & games), and risks the users.

Pegasus Malware

Pegasus was a malware which allowed hackers to run malicious code silently on a target's iOS device, remotely collecting the user's location, data & getting access to the cameras and sensors.

It was sent directly to the target via a text message, containing a link. Once the target opened the link in the web browser on his device, the malware started silently jailbreaking the device, allowing it to run malicious code on the kernel level of the operating system.

Pegasus is deemed by many cybersecurity experts to be one of the most sophisticated attacks seen until today, because of how it takes advantage of conditions co-existing only on mobile devices.

It was discovered by the Citizen Lab research facility, after the Human-Rights activist Ahmed Mansoor received text message containing a link to supposedly evidence of detainees tortured in UAE jails.

[2] Common Vulnerability Scoring System.

[3] National Vulnerability Database.

Bug Activation

To do so the script makes use of the dynamic nature of JavaScript, to manipulate the properties of objects and primitives in the language.

We abuse the implicit type conversion mechanism available in JavaScript, by overriding an object's **O.toString** method, making it clear references to objects in memory and making it look like a number primitive.

Then by overriding the **length** calculated property of a JavaScript array **R**, we can make sure that accessing the array's elements, will cause **O.toString** method to be called and free up the array on the same time.

These means we're both allocating & freeing the memory reserved for the array, which releases the memory, but keeps the reference intact and usable.

We then create a typed **UInt32Array** array which allows for working directly with the memory in JavaScript, and a regular JavaScript array. Our first step is to make those two arrays' memory overlap.

If everything is done correctly we should have a dynamic reference which overlaps with the finite typed array, allowing us to read/write data to the entire data segment of the process.

By manipulating function pointers in the process data segment, we can crash the process, leaving us with access to the operating system code.

We can now run arbitrary (and potentially malicious) code on the device, namely the Nintendo Switch in our case.

Requirements Compliance

The Switch had a few core-requirements, of which compliance has not been achieved:
The requirement of not providing a web-browser application, or preventing web-browsing on the device have not been met. Users can make use of design flaws in the connectivity section to launch a fully-functional browser.

Second, the browser that was used for this so called "limited task", is using a version on the WebKit engine which is already known for being vulnerable to very risky attacks.
This goes against the strict security policies which Nintendo took on itself and should comply with.

Furthermore, the Switch's software distribution policy of "closed-garden", requires all the applications to be approved (signed) and licensed by Nintendo. However, this policy has been provoked, allowing external, unsigned, and unlicensed software to be ran on the device.

It's important to mention that a patch is yet to be delivered by Nintendo, even though the only measure need to do so is to upgrade the WebKit engine to a newer, patched, version.

If those requirements were taken into consideration
Then the vulnerability described was not present on the Nintendo Switch operating system.

Optional additional requirements
Increase the security level used by the connectivity browser, so that it blocks usage of certain JavaScript functionality like using typed arrays.

This will not affect the captive portal pages which use simple JavaScript practices, and will prevent heavy-duty sites from running on the Switch, thus limiting the security risk.

Bibliography

- National Vulnerability Database (NVD) – “CVE-2016-4657”
  https://nvd.nist.gov/vuln/detail/CVE-2016-4657

- Common Vulnerabilities and Exposures (CVE) – “CVE-2016-4657”
  https://cve.mitre.org/

- Wikipedia – “Pegasus (spyware)”
  https://en.wikipedia.org/wiki/Pegasus_(spyware)

- Wikipedia – “Captive portal”
  https://en.wikipedia.org/wiki/Captive_portal

- Lookout – “Technical Analysis of Pegasus Spyware (Paper)”
  http://goo.gl/XOFYmL

- Lookout Blog – “Sophisticated, persistent mobile attack against high-value targets on iOS”
  https://blog.lookout.com/blog/2016/08/25/trident-pegasus/

- LiveOverflow @ YouTube – “What do Nintendo Switch and iOS 9.3 have in common?”
  https://www.youtube.com/watch?v=xkdPjbaLngE

- Qwertoruiop @ Twitter – Developer of iOS 9.3 “Jailbreak Me” software
  https://twitter.com/qwertyoruiopz