# Tour de Vulnerability Disclosure

What to do and what not to do with your 0-day

# Disclaimer

There are no hard and fast rules for vulnerability disclosure and most of this presentation is based off of my personal experiences.  If you are looking for a standard, I recommend the Google Project Zero write up:

https://googleprojectzero.blogspot.com/2020/01/policy-and-disclosure-2020-edition.html

# What is Vulnerability Disclosure?

In short Vulnerability Disclosure is the process of transferring knowledge of a security issue in software from the discoverer to the software owner.

Usually this is a researcher sending information to a vendor.

# What does Vulnerability Disclosure look like?

Every disclosure scenario is different, but roughly these are the steps:

1. You find a security vulnerability that you can reproduce.
2. You reach out to the owner of that software (vendor, open source project, etc) and report the vulnerability.
3. The owner triages the vulnerability
4. The vulnerability gets fixed
5. You publish a report of your findings.

# Schools of Thought

There are several different schools of thought for vulnerability disclosure:

- “Responsible” Disclosure
- Coordinated Disclosure
- Full Disclosure

Each has their Pros and Cons.

# “Responsible” Disclosure

Contacting the vendor and following all vendor guidelines and suggestions.  This generally includes either not publishing the vulnerability, or if publishing is allowed, publishing only when the vendor allows you to.

This really puts control of the process in the vendor’s hands, giving the researcher little to no control. Likewise, if you as a researcher do anything outside of the “Responsible” Disclosure paradigm, you will be labeled “irresponsible” by the vendor.

# “Responsible” Disclosure Pros and Cons

**Pros:**

If strictly adhered to, there will be no issue with the vendor because the vendor will control the entire process.

**Cons:**

No control over any aspect of the process.

# Coordinated Disclosure

A collaborative approach to vulnerability disclosure.  This takes the model that the researcher and the owner are working together to fix the problem in a manner that benefits all parties involved. Typically after triage, the owner and the researcher agree on a disclosure timeline for publishing a report.  Also, the owner generally handles coordination with CERT and filing a CVE

# Coordinated Disclosure Pros and Cons

**Pros:**

Gives both parties equal control over the process.

Generally doesn't prevent publishing findings.

**Cons:**

Entire process can take months and in some cases years.

# Full Disclosure

There is minimal if any contact with the vendor.  The entirety of your research is published publicly before there is a fix available.

# Full Disclosure Pros and Cons

**Pros:**

Motivates owner to fix problem as soon as possible

**Cons:**

Will generally get you as a researcher blacklisted in some form by the owner.

Certain community members will criticize you.

# Examples

CVE-2018-7084 - Unauthenticated command execution in Aruba Product

Coordinated Disclosure - Worked with Product teams to develop a fix and agreed upon disclosure publishing details during triage.

https://www.arubanetworks.com/assets/alert/ARUBA-PSA-2019-001.txt

# More Examples

CVE-2016-10540 - Regular Expression Denial of Service in Minimatch JavaScript library

Coordinated Disclosure - I worked with the npm developer for minimatch and coordinated publishing with them upon a proper fix being released.

https://nvd.nist.gov/vuln/detail/CVE-2016-10540

# Full Disclosure Examples

https://kb.netgear.com/000061582/Security-Advisory-for-Signed-TLS-Certificate-Private-Key-Disclosure-on-Some-Routers-PSV-2020-0105

Netgear Signed TLS Private Key Disclosure

Credits: Tom Pohl and Nick Starke

# Story Time!

Tom Pohl and I were perusing netgear firmware when we started seeing expired but signed private keys for TLS certificates in the firmware images.

That lead us to hunt for unexpired TLS private keys that were validly signed.

With a little digging we found two validly signed TLS certificates with their corresponding private keys.

# What was the vulnerability?

Private Keys are meant to be kept private.  In this circumstance they were posted in a public place that required no authentication and only a rudimentary means of extracting the data.

The Private keys were contained within firmware images meant to be downloaded and flashed onto home routers.

# Exploitability

For the most part this vulnerability was only exploitable if an attacker had a man-in-the-middle position on a network.  This includes the ability to ARP spoof, DNS spoof, and/or intercept network traffic.

There turned out to be an exception to this rule that was developed later as subsequent art to the initial findings.

# Extraction Technique

Firmwage images come as a binary large object (BLOB) and must be extracted to reveal the contents of the BLOB.  For this project, we used Binwalk to handle the extraction in an automated manner:

https://github.com/refirmlabs/binwalk

# Contact?

We tried to establish contact with the vendor (Netgear) but were not able to establish a communications channel outside the Netgear Bug Bounty program.

# Why not use the Bug Bounty Program

We didn’t want to use the bug bounty program because the bug bounty program does not allow disclosure.  We wanted to make sure the TLS certificates were revoked, and the only way we could assure that is through public disclosure.

# No Contact

After six days, we had not been able to establish any contact with the vendor.  We decided there was no benefit in waiting any longer and we published our findings:

https://gist.github.com/nstarke/a611a19aab433555e91c656fe1f030a9

# Response

The response was immediate:

- Some hailed us as heroes
- Some hailed us as irresponsible script kiddies trying to rack up publicity.

https://news.ycombinator.com/item?id=22096659

We were on the front page of hacker news for over 24 hours.

# Vendor Response

The vendor response came two days after disclosure.  They publicly thanked us for alerting them to the issue, but privately fumed at us not going through the bug bounty program.

# Certificate Authority Response

When a signed TLS private key is leaked the Certificate Authority is responsible for revoking the certificate within 24 hours of notification.

The CA in this case took about 32 hours to revoke the certificate, and this was noted by browser vendors who keep track of these sorts of stats when deciding what CA's to include in the browser CA Bundle.

# Press Response

We were featured in a few articles, including two from the register:

https://www.theregister.co.uk/2020/01/20/netgear_exposed_certificates/

https://searchsecurity.techtarget.com/news/252477198/Netgear-under-fire-after-TLS-certificates-found-in-firmware-again

# Subsequent Art

https://saleemrashid.com/2020/02/09/exploiting-netgear-routerlogin/

Unrelated developer created a proof of concept using service workers to attack routers which contained this private key.

# Remediation Result

The vendor issued a beta firmware version within a few days of the initial advisory.

Within one week consumers had the option of updating to the beta firmware to keep themselves safe from this vulnerability.

# Who is the winner in this situation?

The consumer.

No one comes out ahead in this situation really other than the consumer who benefits from the quick patch cycle.

The researcher(s), the vendor, and the certificate authority all lose.

# Questions?