black hat
USA 2024

Title
Bypassing ARM's Memory Tagging Extension with a Side-Channel Attack

Contributors
Juhee Kim (Seoul National University)
Jinbum Park (Samsung Research)
Sihyeon Roh (Seoul National University)
Jaeyoung Chung (Seoul National University)
Youngjoo Lee (Seoul National University)
Taesoo Kim (Samsung Research and Georgia Institute of Technology)
Byoungyoung Lee (Seoul National University)

Tracks
Hardware / Embedded; Exploit Development & Vulnerability Discovery

Format
30-Minute Briefings

Abstract
ARM Memory Tagging Extension (MTE) is a new hardware extension introduced in ARMv8.5-A architecture designed to detect memory corruption. Compared to previous mitigation techniques such as DEP, ASLR, and CFI, MTE can detect the root cause of memory corruption attacks. For this reason, MTE is considered the most promising path forward for improving C/C++ software security by many security experts, since its first adoption with Pixel 8 in October 2023.

In this talk, we show that despite high hopes, MTE is not yet the silver bullet for eliminating memory corruption attacks. Specifically, we introduce new exploitation techniques that leak the MTE tags through speculative execution. We demonstrate that the MTE-based protection in Google Chrome and the Linux kernel can be bypassed.

Our findings suggest that while MTE represents a significant advancement in memory safety, it is not yet safe against side-channel attacks, and further improvements are necessary to secure systems effectively.

Presentation Outline – Note the detailed outline.
Brief Outline
Introduction to ARMv8.5 Memory Tagging Extension (MTE)
TikTag: Speculative MTE oracles
Real-world MTE bypass attacks with TikTag
Guidelines for safe MTE usage

ARMv8.5 Memory Tagging Extension (MTE)
We'll first introduce the MTE, ARM's newly adapted hardware extension. MTE supports these primitives: (i) memory tagging and (ii) tag check at memory access. With MTE, a software system can be hardened against memory corruption attacks with random tagging-based solutions.

TikTag: Speculative MTE Oracles
The premise behind assigning a random tag is based on the assumption that attackers cannot predict the tag of a memory block. In other words, leaking the random tag would break the memory safety provided by MTE.

Would it be possible to leak the MTE tag? Previously, the PACMAN attack (https://pacmanattack.com/) utilized speculative execution to guess the correct Pointer Authentication Code (PAC) for any given pointer. We took a similar approach to leak the MTE tag using speculative execution.

Here, we'll show that MTE tags can indeed be leaked through speculative execution, by executing TikTag gadgets that leak the tag check result through the cache side-channel.

```
——————————————————————
// TikTag-v1
BR: ldr cond [cond_ptr]
cbz cond, End
CHECK: ldr r0, [guess_ptr]
ldr r0, [guess_ptr] ;; >= 2 loads
GAP: … ;; >= 20 cycles
TEST:
;; INDEP_LOAD
ldr r1, [test_ptr]
;; INDEP_STORE
str r1, [test_ptr]
;; DEP_LOAD
ldr r1, [test_ptr, r0]
;; DEP_STORE
str r1, [test_ptr, r0]
——————————————————————
```

Fig-1 TikTag-v1

TikTag-v1 (Fig-1), for instance, introduces an execution gap between the tag check and the memory access to leak the tag check result. Depending on the number of loads in CHECK and the CPU cycle latency introduced in GAP, the cache hit rate of `test_ptr` was dropped to at most 80%, providing enough side-channel attack possibilities.

We will also show another similar MTE oracle, TikTag-v2, that exploits the discrepancy of store-load forwarding behaviors depending on the MTE tag check results.

Real-World MTE Bypass Attacks with TikTag
Next, we'll demonstrate that TikTag gadgets can be exploited to bypass MTE in real-world settings, notably Google Chrome and Linux Kernel, each posing distinct challenges for the attacker.

Attacking the Google Chrome
Assuming a typical browser renderer RCE threat model, we first introduce a newly identified speculative V8 sandbox escape (Fig-2). The V8 sandbox does not ensure the complete sandbox guarantee in the speculative execution path, specifically when accessing TypedArray with a 64-bit value as an index.

```
——————————————————————
// JS code
idxArray = new Float64Array;
victimArray = new Float64Array;

idx = idxArray[0]; // idx: 64-bit double
val = victimArray[idx]

// Turbofan optimized assembly
…
ldur x3, [x2, #35] ;; victimArray.raw_length
ldur w4, [x2, #51] ;; victimArray.base_ptr
ldur x5, [x2, #43] ;; victimArray.external_ptr
add x5, x28, x5, lsr #27 ;; x28 (sandbox_base)
fcvtzs x6, d0 ;; d0 (idx) 64-bit double index
b.cmp x6, x3 ;; victimArray bound check → 64-bit compare
```

```
b.hs #+0x194 ;; out-of-bounds handling→ Speculatively bypassed
add x4, x4, x5 ;; victimArray.DataPtr = sandbox_base + base_ptr + external_ptr
ldr w3, [x4, x6, lsl #2] ;; load victimArray[idx] → Speculatively accessed
```
———————————————————————
Fig-2. The V8 speculative sandbox escape

After that, we'll show that a TikTag-v2 gadget can be constructed with JavaScript based on the V8 sandbox escape primitive. With this gadget, the MTE tag of any renderer memory can be leaked with a 95% success rate in less than 3 seconds.

Furthermore, we'll present a demo of a real-world renderer RCE attack exploiting a recent heap overflow vulnerability in the renderer (CVE-2023-5217) when MTE is enabled, utilizing the JavaScript TIkTag gadget to bypass MTE.

Attacking the Linux kernel
Next, we will demonstrate a real-world MTE bypass attack in a typical Linux kernel privilege escalation scenario. In the Linux kernel, attackers must identify and exploit TikTag gadgets present within the kernel code. We will outline the conditions necessary for TikTag gadgets in the Linux kernel to be exploitable. Additionally, we will introduce one potentially exploitable gadget we have discovered (Fig-3).

Based on the TikTag gadget found in the Linux kernel, we will demonstrate a demo of an MTE bypass attack. This attack exploits a custom double-free vulnerability without triggering tag check faults.

```
static ssize_t snd_timer_user_read(...) {
…
switch (tu->tread) {
default:
- return -ENOTSUPP;
+ break;
}
// BR: speculation branch with cond_ptr (tu)
switch (tu->tread) {
case TREAD_FORMAT_TIME32:
// CHECK: dereference guess_ptr (tread) with 4 loads
tread32 = (struct snd_timer_tread32) {
.event = tread->event,
.tstamp_sec = tread->tstamp_sec,
.tstamp_nsec = tread->tstamp_nsec,
.val = tread->val,
};
// TEST: dereference test_ptr (buffer)
if (copy_to_user(buffer, &tread32, sizeof(tread32)))
err = -EFAULT;
break;
…
default: // Correct branch destination
err = -ENOTSUPP;
break;
```
———————————————————————
Fig-3. TikTag-v1 gadget in snd_timer_user_read(). -/+ denotes the code changes to make the gadget exploitable.

Guidelines
Finally, we will propose guidelines for the safe deployment of MTE to mitigate the exploitation of TikTag gadgets.

In environments like Google Chrome, where untrusted code executes within the target system, the sandbox should be designed to be aware of speculative paths. Currently, both the Chrome V8 sandbox and Safari Webkit sandbox do not completely mediate speculative paths. By developing a speculative execution-aware sandbox, the speculative path within a sandbox would be unable to access non-sandboxed memory and TikTag gadgets cannot leak the renderer's MTE tag.

In scenarios where the attacker operates from a different privilege context than the target system, such as in the Linux kernel, implementing speculation barriers can mitigate MTE bypass attacks. For instance, by preventing speculative execution prior to accessing user space addresses in the kernel (e.g., copy_to_user()), the kernel can effectively block direct leakage of tag check results from the user space buffer, significantly raising the bar for attackers.

Is This Content New or Has it Been Previously Presented / Published?
This content is new and has not been previously presented or published.

Do You Plan to Submit This Talk to Another Conference?
We have submitted a paper describing the MTE side-channel attack to USENIX Security 2024, which is currently under review. While the paper examines the same speculative MTE oracles, this Black Hat talk will focus on the real-world MTE bypass exploitation leveraging TikTag.

What New Research, Concept, Technique, or Approach is Included in Your Submission?
Since the first adaptation of MTE on Google Pixel 8 in October 2023, multiple security researchers have attempted to discover MTE oracles, due to their strong potential for bypassing MTE at software systems hardened with MTE [7, 8]. This is the first to identify working speculative MTE oracles and demonstrate their effectiveness in real-world settings: Google Chrome and the Linux kernel.

[7] MTE As Implemented, Part 1: Implementation Testing, https://googleprojectzero.blogspot.com/2023/08/mte-as-implemented-part-1.html
[8] Sticky Tags: Efficient and Deterministic Spatial Memory Error Mitigation using Persistent Memory Tags, https://openreview.net/pdf?id=zsIPQAkqgO

Provide 3 Takeaways
1. We show that ARM MTE does not provide perfect security.
2. Even real-world targets including Chrome and Linux Kernel can be bypassed with our MTE bypass attacks.
3. Security guidelines for MTE users to prevent MTE bypass attacks.

What Problem Does Your Research Solve?
* Discover new speculative gadgets to break MTE
* Construct real-world attacks on the MTE-hardened systems
* Provide guidelines to minimize the impact of speculative MTE oracles

Will You Be Releasing a New Tool? If Yes, Describe the Tool.
We plan to release the TikTag gadget prototypes and the proof-of-concept of MTE bypass attacks on Google Chrome and the Linux Kernel. The resources will be released after the acceptance notification under the open-source license.

Is This a New Vulnerability? If Yes, Describe the Vulnerability.
TikTag is a new vulnerability in ARM Memory Tagging Extension (MTE) that allows an attacker to leak the result of speculatively executed MTE tag checks without reading the MTE tag using legitimate tag load instructions (e.g., `ldg`).

Google V8 speculative sandbox escape is a new vulnerability discovered by this work. The V8 JavaScript engine does not provide sandbox protection against speculative execution, so the attacker can leak both the memory data and MTE tags of the entire process.

If This is A New Vulnerability, Has It Been Disclosed to the Affected Vendor(s)?
We have disclosed TikTag-v1 and TikTag-v2 gadgets to ARM in November 2023. ARM has acknowledged the issue and encouraged the vendors to mitigate the issue [9].

We also disclosed the Google V8 sandbox escape vulnerability and MTE tag leakage vulnerability to Google in December

2023. Google has acknowledged the issues but decided not to fix them since information leakage (arbitrary memory read) and tag leakage are not in the threat model of the V8 sandbox. However, we disagree with this rationale behind the V8 sandbox's missing confidentiality guarantee, considering the security benefits of deploying MTE to a web browser and the security risks of not preventing memory and MTE tag leakage.

[9] https://developer.arm.com/documentation/109544/latest

Will Your Presentation Include a Demo? If Yes, Describe the Demo.
We will demonstrate MTE bypass attacks on real-world scenarios, Google Chrome and the Linux Kernel.

Google Chrome demonstration will show that the attacker can construct TikTag-v2 gadgets with the JavaScript code and leak the MTE tags of the renderer process memory. The demonstration will further show that the tag leakage can reliably trigger real-world memory corruption vulnerability in the renderer process (CVE-2023-5217).

Linux kernel demonstration will show that the attacker in the userspace can exploit a TikTag-v1 gadget residing in the kernel code to leak the MTE tag check result of the kernel memory. Leveraging the leaked tag check results, the attacker can trigger kernel memory corruption vulnerability without triggering tag check faults.

Does Your Company/ Employer Provide a Solution to the Issue Addressed? If Yes, Please Provide Details.
The tag leakage behavior of TikTag gadgets cannot be mitigated without hardware modification. Nevertheless, we will provide possible software fixes that can be made to prevent constructing or exploiting the MTE oracles in the Google Chrome and Linux Kernel. Furthermore, we will provide general guidelines for the MTE users to prevent exploiting MTE oracles in their software.

Why Did You Select the Above Track(s) for your Submission?
We chose the first track–Exploit Development & Vulnerability Discovery, as this work introduces a new side-channel vulnerability in ARM's MTE and develops the MTE bypass attacks on MTE-hardened systems.

We chose the second track–Platform Security since this work discovers new micro-architectural side-channel issues in the ARM's MTE hardware implementation.

White Paper
We will prepare the white paper at a later date.

Title
Listen to the Whispers: Web Timing Attacks that Actually Work

Speaker
James Kettle (Director of Research, PortSwigger Web Security)

Tracks
Application Security: Offense; Reverse Engineering

Format
40-Minute Briefings

Abstract
Websites are riddled with timing oracles eager to divulge their innermost secrets. It's time we started listening to them.

In this session, I'll unleash novel attack concepts to coax out server secrets including masked misconfigurations, blind data-structure injection, hidden routes to forbidden areas, and a vast expanse of invisible attack-surface.

This is not a theoretical threat; every technique will be illustrated with multiple real-world case studies on diverse targets. Unprecedented advances have made these attacks both accurate and efficient; in the space of ten seconds you can now reliably detect a sub-millisecond differential with no prior configuration or 'lab conditions' required. In other words, I'm going to share timing attacks you can actually use.

To help, I'll equip you with a suite of battle-tested open-source tools enabling both hands-free automated exploitation, and custom attack scripting. I'll also share a little CTF to help you hone your new skillset.

Want to take things further? I'll help you transform your own attack ideas from theory to reality, by sharing a methodology refined through testing countless concepts on thousands of websites. We've neglected this omnipresent and incredibly powerful side-channel for too long.

Presentation Outline – note the detailed outline
I'll open with the story of my attempt to replicate a timing attack writeup that looked incredibly powerful but, from what I can tell, was actually impossible.

After the intro, I'll provide a brief overview of the current timing technique landscape, focusing on practicalities such as accuracy and efficiency:
- Which timing technique works best for HTTP 1, 2 & 3
- When to use the 'fat-packet' approach from Timeless Timing Attacks instead of the 'single-packet attack' from Smashing the State Machine

- How to avoid the misleading-benchmark trap

For the first and simplest new attack class, I'll show the audience how to exploit timing oracles to reveal hidden attack-surface. I'll illustrate this with two case-studies:
- Detecting the tell-tale delay from DNS lookups, in order to identify hidden HTTP headers that enable IP address spoofing. This affects numerous websites
- Spotting a vulnerability in *redacted* by recognising an 'early-exit' in a server-side code path caused by an invalid input in a secret header

Next I'll take the 'early-exit' concept further and show how to use timing analysis to detect server-side injection vulnerabilities by triggering exceptions that short-cut code paths. This approach can reveal vulnerabilities that are

otherwise near-impossible to find, like blind JSON injection. I'll illustrate this with three case studies:
- Detecting blind SQL injection in a trading website with a completely static response, via timing analysis on the classic ' vs '' payload pair
- Detecting blind server-side JSON response injection in *redacted*
- Detecting blind server-side parameter pollution on *redacted*

For the final and most severe attack class, I'll introduce a widespread reverse-proxy misconfiguration that gives attackers access to internal systems. I'll show how to map out flawed routing rules using calculated timing probes that exploit DNS lookups, wildcard handling and DNS caching. I'll illustrate the impact with three case studies:
- Access to the *redacted* Admin panel
- Access to Redbull's media publishing platform
- Routing around a front-end to gain access to *redacted* internal admin panel

Then I'll live-demo iteratively discovering a complete BigIP WAF bypass without triggering a single detection, using nothing but response timing information. Please refer to the demo section for full details.

Next I'll pass the baton to the audience and share a methodology for taking your own novel timing attack concept from theory to reality. As a worked example, I'll use the classic but mostly impractical non-constant-time string-comparison attack. This section will cover:
- Benchmarking your attack's accuracy requirements
- Solving unrealistic accuracy requirements with delay-expansion techniques
- Evaluating the technique in the wild: common pitfalls & workarounds
- Using non-blind findings in tandem to assist understanding & exploitation
- Promising unexplored concepts & ideas for further research

Then I'll take a look at defence from multiple perspectives, covering current threats and how they may evolve in the future:
- How to mitigate timing attacks as a webserver, WAF, or application
- Timing attack roadmap: which currently-theoretical attacks are the closest to becoming practical, and will arrive in the wild first

I'll wrap up with the key takeaways and leave 5 minutes for questions.

Is This Content New or Has it Been Previously Presented / Published?
Completely new.

Do You Plan to Submit This Talk to Another Conference?
I plan to submit this to DEFCON 32 only - Black Hat USA will be the first public presentation of this.

What New Research, Concept, Technique, or Approach is Included in Your Submission?
I'll introduce a ground-breaking timing-analysis technique for mapping out exploitable reverse-proxy routing rules. Allowing routing to arbitrary subdomains is a common, often high-severity misconfiguration that is currently widely overlooked due to the difficulty of detecting it from a black-box perspective.

I'll also introduce new timing-analysis techniques to uncover otherwise-undetectable attack-surface, and discover server-side datastructure injection vulnerabilities like JSON injection.

I'll also share a detailed methodology that the audience can use to develop their own timing attacks for custom objectives. This will include hard-won lessons, and novel techniques like delay-expansion and synchronised-exploitation.

Provide 3 Takeaways

1. Websites and servers are riddled with timing oracles leaking valuable information on vulnerabilities, configuration & attack-surface.
2. The advent of single-packet HTTP/2 timing techniques has made exploiting these oracles fast, reliable, and easy - and new oracles are around the corner.

3. Timing attacks are actively revealing vulnerability classes that have lingered and become endemic because there was no way to detect them previously.

Will You Be Releasing a New Tool? If Yes, Describe the Tool.
Yes. On the day of the presentation, I will release two major tool updates. Both tools are open-source Apache-licensed plugins for Burp Suite, fully compatible with the free community edition, and already in the top ten most popular extensions.

Param Miner will be updated to:
- Detect reverse-proxy misconfigurations that expose internal systems via timing analysis
- Detect invisible parameters/cookies/headers via timing analysis
- Detect server-side data/code injection via timing analysis

Provide a clean API for the audience to develop and share their own timing attacks

Turbo Intruder will be updated to better support exploiting timing-based vulnerabilities, technique benchmarking, and development of custom timing attacks.

Is This a New Vulnerability? If Yes, Describe the Vulnerability.
The attacks and vulnerabilities in this presentation are known in theory, but rarely discovered in black-box testing in practice. For example:

Blind SQL injection can typically be detected using a 'sleep' statement, but blind injection into data-structures like JSON and server-side HTTP query strings is almost impossible to detect without the techniques in this presentation.

Reverse proxies that route to arbitrary hosts can be detected using DNS pingbacks as shown in 'Cracking the lens', but front-ends that require a fixed suffix like ".example.com" can't be detected in this manner.

If This is A New Vulnerability, Has It Been Disclosed to the Affected Vendor(s)?
The *redacted* attack-surface case-study has been reported and patched.

The three reverse-proxy misconfiguration case studies have all been reported and acknowledged by the respective vendors, as has the SQL injection.

I have not yet reported the JSON injection and server-side parameter pollution vulnerabilities as they need more development to maximise impact.

If any vulnerabilities are still unpatched at the time of the presentation, I will swap in an alternative case-study, or redact the target name.

Will Your Presentation Include a Demo? If Yes, Describe the Demo.
In the demo, I will prove timing attacks are practical by targeting a live remote system, rather than one in a lab environment. To make this possible, I'm going to demo a Web Application Firewall (WAF) bypass discovery on a *redacted* hostname, rather than an outright vulnerability.

In this demo, I'll use timing analysis with Turbo Intruder to:
- Discover a secret parameter that makes a *redacted* WAF do additional processing
- Discover that the delay stacks when the parameter is repeated
- Discover that the delay stops stacking after a certain number of repetitions
- Discover that this indicates a length cutoff after which you can inject arbitrary payloads without the WAF blocking them

This will showcase just how versatile timing analysis can be, and how much you can learn from timing in the face of a static response.

There is a risk that the WAF flaw will be patched prior to the presentation. To mitigate this, I will prepare a more traditional back-up demo targeting a remote misconfigured nginx reverse-proxy that I control. I'll publish this back-up as a CTF afterwards.

Provide the Names of the Speakers Presenting and Their Previous Speaking Experience.
James 'albinowax' Kettle

Most recent video sample: https://youtu.be/tKJzsaB1ZvI

Speaking experience:
Black Hat USA 2023, 2022, 2021, 2020, 2019, 2018, 2017, 2015
Black Hat EU 2021, 2018, 2016

For a full list with recordings please refer to https://jameskettle.com/#showtalks

Does Your Company/ Employer Provide a Solution to the Issue Addressed? If Yes, Please Provide Details.
This presentation is focused on novel timing attacks and techniques rather than any specific tools. The two proof-of-concept Burp Suite plugins I'll release will be open source and compatible with the free edition - Burp is just providing the network stack, which could be swapped for an open-source alternative like https://github.com/nxenon/h2spacex which I'll reference. If you have any concerns, it may help to watch one of my previous presentations such as 'Smashing the State Machine' from Black Hat USA 2023: https://youtu.be/tKJzsaB1ZvI

Why Did You Select the Above Track(s) for your Submission?
This talk is primarily about web security offence techniques so I choose AppSec Offense as the main track.

At a deeper level, it's about using timing techniques to reverse-engineer black-box systems, so I choose reverse engineering as the secondary track.

Message for Review Board Only
My primary goal is to inspire the audience to try timing attacks for themselves, and integrate them into their regular workflows. Hopefully some people will be further inspired to research and publish their own novel timing attacks.

I'm keenly aware that there have been a lot of conference presentations about timing attacks over the years, but none have really stuck. I have crafted the title, abstract and content to communicate why this session will be different.

Two major elements have combined to make timing attacks practical now. The first is my three novel attack-types, which cause much larger time delays than classic timing attacks like token-extraction. The second is the single-packet attack - I originally developed this technique for race condition attacks, but it actually outperforms all published timing attack techniques in most scenarios. However, because I already discussed it in detail last year in "Smashing the State Machine", the implementation details are not a focus of this presentation. Instead, I'll focus on how it works in practice.

I plan to focus my research over the next 2-3 months on improving the severity of the case-studies, ideally resulting in some major bug bounties to further motivate the audience.

Selected Submission – Black Hat USA 2023

Title Nothing but Net: Leveraging macOS's Networking Frameworks to Heuristically Detect Malware

Speaker Name Patrick Wardle

Tracks Network Security; Malware

Format 40-Minute Briefings

Why Did You Select the Above Track(s) for your Submission?

- The network security track makes sense for a talk about creating network monitoring tools.
- Also, as a secondary track "malware" makes sense, as the goal of the networking tools is to detect malware.

Abstract

As the majority of malware contains networking capabilities, it is well understood that detecting unauthorized network access is a powerful detection heuristic. However, while the concepts of network traffic analysis and monitoring to detect malicious code are well established and widely implemented on platforms such as Windows, there remains a dearth of such capabilities on macOS.

This talk aims to remedy this situation by delving deeply into a myriad of programmatic approaches capable of enumerating network state, statistics, and traffic, directly on a macOS host. We will showcase open-source implementations of relatively overlooked low-level APIs, private frameworks, and user-mode extensions that provide insight into all networking activity. And, by leveraging these techniques, you will learn how to efficiently and generically detect both known and unknown threats targeting macOS!

Presentation Outline

/* Part 0x0: Introduction */

In this (brief) part of the talk, we will highlight:
A. The importance of detecting unauthorized network access in the context of malware detection (especially noting the fact that host-based (vs. on the network) approaches provide the responsible process).

We'll also give specific examples of recent macOS malware that accesses the network …ranging from payloads used in 3CX supply chain attack, to previously undetected implant updaters.

B. Explanation of the lack of network traffic analysis and monitoring capabilities on macOS. Showing this is due to a few reasons, such as changes at the OS level (e.g. deprecation of network kernel extensions), lack of documentation of newer approaches, or even the fact that Apple has chosen to keep certain frameworks and APIs private.

/* Part 0x1: Network State and Statistics */
We'll then dive in, here, in this part of the talk focusing on methods that provide a snapshot of the network (including listening sockets, network connections, traffic stats, and more).

A. Enumerating Network State (per process) via proc_pid APIs.
Starting with proc_pidinfo & PROC_PIDLISTFDS to get a list of all file descriptors currently opened by a process, then focusing on file descriptors whose type are PROX_FDTYPE_SOCKET.

B. Enumerating Network State and Statistics via the private NetworkStatistics framework
We'll show how to extract APIs and symbols from the framework, and then via (private) APIs such as NStatManagerCreate, NStatManagerQueryAllSourcesDescriptions, show how globally enumeration the network state and statistics such as bytes transmitted down/up.

/* Part 0x1: Network Monitoring */
In this part of the talk, we'll focus on approaches that provide real-time monitoring capabilities of the network, on macOS.

A. Introduction to System Extensions and the Network Extension Framework
…including the entitlements required, how to craft an application bundle (containing an extension), and then how to install & activate your extension.

B. Creating a DNS monitor (com.apple.networkextension.dns-proxy)
After highlighting the benefits of host-based DNS monitoring (e.g. it provides a light-weight approach to detecting malware resolving domains, etc etc), we'll dive into building a DNS monitor, atop Apple's Network Extension Framework.

(Fun fact, I've been working on an open-source implementation of a DNS monitor, see https://github.com/objective-see/DNSMonitor), for the last year or so. AFAIK this is (was?) the first open-source implementation using macOS' new(ish) Network Extension Framework. Lots of lessons learned and comms w/ Apple engineers as their documentation had some inconsistencies and was incomplete. This open-source tool will act as a comprehensive case-study for this section of the talk).

As DNS is predominately sent over UDP, in this talk we'll focus solely on UDP flows, whose type will be NEAppProxyUDPFlow. We'll show how to our DNS monitor must:

1. Open the flow, with the local endpoint
2. Read from the flow
3. Open the remote endpoint
4. Send the read datagrams (from step #2) to the remote endpoint
5. Read any response from the remote endpoint
6. Write any response from the remote endpoint to flow

At step two, we'll have a list of DNS packets (likely a DNS "question") while at step five, we'll have another DNS packet (likely a DNS "answer" to the "question").

We'll also show how the process responsible for the DNS request (e.g. malware) can be identified from the flow's audit token.

C. Comprehensive Network Monitoring (com.apple.networkextension.filter-data)
Next, we'll show how to build a full network monitor or filter, (again, atop the Network Extension Framework). We'll show how to filter on all protocols and either only outgoing traffic (or incoming as well).

Using this (and information from the NEFilterSocketFlow), we'll show how a fully featured firewall can be built.

D. Tools vs. Malware
We'll then pit various malware specimens against the tools and techniques presented in this talk. For example, we'll show how the simpler "snapshot"-based approaches can reveal listening sockets tied to reverse-shells. Of course, we'll also show how the network monitors (DNS and full protocol) can flag implant exfiltration, C&C tasking, and more.

Note that for each tool, we'll provide a full open-source implementation!

(Fun fact two, I've submitted the networking tools I've been building and covered in this talk, to BH Arsenal. If this talk and the Arsenal session get accepted, attendees could be able to both come to this talk to get a conceptual understanding of the tools and their approaches, as well as then come to the Arsenal session to try them out in a hands-on manner).

E. Conclusions / Takeaways
We'll wrap up the talk, with a brief discussion of the importance of incorporating these capabilities into cybersecurity strategies for macOS environments

We'll also summarize the talk's key points and the tools presented.

What new research, concept, technique, or approach is included in your submission?

- Surprisingly there hasn't been a lot of talk / research on host-based network monitoring on macOS (especially in the context of malware analysis). This is maybe because Windows has long been the dominant OS in the enterprise and/or a lot of network monitoring is done off host (e.g. on the network). Now Macs are super prevalent in the enterprise, and host-based monitoring has many advantages (access to unencrypted data, ability to identify the responsible process, etc etc). Thus, the majority of the content in this talk is new …or at the very least, has not been widely discussed nor understood.

Takeaways

1. Host-based network monitoring is a very efficient and powerful approach to detecting malware on macOS.

2. macOS supports a wide range of mechanisms to build host-based network tools (though some are found within private frameworks/APIs or are not particularly well documented).

3. The approaches and open-source tools presented in this talk can (and should be) leveraged to build enterprise-grade EDR products for macOS …bringing parity with Windows tooling.

If applicable, what problem does your research solve?

- The dearth of information regarding networking monitoring / detection tools on macOS.

Will you be releasing a new tool? If yes, is the tool open source or commercial?

- Yes(ish). I've been working on the DNS monitor - and will be releasing a new (and fist non-beta) version with a host (ha!) of new features / capabilities such as DNS filtering / blocking, etc. etc.
- It's open-source (GPL v3)

If the speaker(s) listed above are previous Black Hat Briefings speakers, when/where did they speak?

- BH USA 2020 https://www.blackhat.com/us-20/briefings/schedule/#office-drama-on-macos-20854
- BH USA 2021: https://www.blackhat.com/us-21/briefings/schedule/#armd-and-dangerous-23772
- BH USA 2022: https://www.blackhat.com/us-22/briefings/schedule/#dj-vu-uncovering-stolen-algorithms-in-commercial-products-27673

So the Review Board is able to get a sense of their presentation style(s), please provide a link to a video sample.
My talk on exploiting Zoom 0days at DefCon last year was recently posted: https://youtu.be/jk1OMo8Gcsk?t=18

Message for Review Board Only

- Thank you for considering my submission!
- I'm a passionate Mac malware analyst (virologist?) and have recently spent a lot of time diving into network-based detection, directly on the host.
- I'd be stoked to share my findings with the BlackHat attendees and provide actionable takeaways so we're all better prepared to both detect & thwart malware targeting macOS!

Selected Submission – Black Hat USA 2021

Title: Government-Mandated Front Doors?: A Global Assessment of Legalized Government Access to Data

Speaker: Andrea Little Limbago, Interos

Abstract (Provide a concise, yet detailed description of your presentation - 300 word maximum):

Who needs a backdoor when front door access is required? From Tesla to the U.S. tech giants, there has been growing focus on whether private sector companies are obliged to turn over data to a foreign government in exchange for market access. This can take the form of source code reviews to unfettered access upon request and increasingly may pose a risk to intellectual property and personal data as digital authoritarian frameworks proliferate.

This comes at a time when significant supply chain disruptions have prompted many in the private sector to reassess their global footprint, with cybersecurity a top priority and motivator when exploring greener pastures elsewhere. Integrating government data access policies must become core to these considerations as corporations reshore and transform their global footprint.

But how do these policies compare from one country to the next? Has the GDPR inspired more progeny or is the Chinese model spreading faster as many contend? To address these questions, this presentation will introduce a new global index of countries based on government-mandated data access requirements. We will discuss the data and factors driving the index, as well as elicit community recommendations for improving the model. With such significant global transformations underway, government-mandated data access warrants greater attention when exploring the full range of global cyber risks.

Notes:
•Detailed, yet concise abstract
•Defines a problem and offers a solution(s) that will be examined during the session.

Presentation Outline (Show the progression of your presentation.)

I. Introduction
a. Common adage that tech outpaces policy, but there are significant policy shifts across the globe
b. With the growing Splinternet/Balkanization of the internet and digital and physical supply chains, I was curious how this was playing out with regard to government-mandated data access. Are companies reshoring from one high security risk location to another when it comes to government access to data?
c. Are Russian mandatory source code reviews and Chinese local data storage and security reviews an anomaly or spreading to other regimes? What about Australian backdoor legislation or data protection similar to that found in the GDPR?
d. Building on and integrating the growing literature on encryption, cross-border data flows, and the techno-dictator/digital democracy archetypes, I created a global ranking of government access to data. This talk will share the findings and seek collaboration and feedback to continue to help improve the index and highlight this shifting landscape.

II. Shifting regulatory landscapes in context -
a. The rise of data protection laws and individual data rights - GDPR and its global offspring
b. Data sovereignty - growing requirements to store data locally and/or provide government access to data
c. The various motivations -> national security largely the underlying justification achieved through various means
d. Variations in implementation with examples - ranging from mandated certificates and unfettered access to transparent and limited access with a warrant

III. Why this matters for supply chain resilience
a. Quick overview of the stats and growth of reshoring and onshoring over the last few years

IV. The Model Framework
a. Rankings coded and based on a range of factors:
1) Degree of data localization and storage requirements
2) Degree of legalized government access -> from no access to unfettered access
3) Transparency of government access
4) Joint venture requirements for a local presence
5) Source code access requirements
6) The existence of a federal data protection law and independent data protection authorities

V. The Findings
a. Global country rankings
b. Interesting trends or non-intuitive results
c. All summarized in a white paper

VI. Next Steps
a. Gather feedback from the community for improvement
b. Iterate on the analysis by end of the year
c. Motivate broader thinking about the range of cyber risks and data protection considerations as globalization transforms

Notes:
•Clearly conveys progression of talk.
•Helps the Review Board visualize the presentation in its entirety.
•Gives an explanation of each area of the presentation.

Attendee Takeaways (What are three actionable takeaways included in your presentation?)
1) Government-mandated access to data poses a growing risk across the globe
2) Rethinking cyber risk to include government policies is increasingly necessary and introduces new risks to IP and PII
3) As supply chains transform and reshore, government data policies should be a top cyber consideration when assessing locations

Notes:
•Submitter fulfilled requirement of providing 3 takeaways
•Explains relevance to the audience
•Cleary emphasizes the participant benefits

Selected Submission – Black Hat USA 2021

Title: Can You Hear Me Now? Remote Eavesdropping Vulnerabilities in Mobile Messaging Applications

Speaker: Natalie Silvanovich, Security Engineer, Google

Abstract:

On January 29, 2019, a serious vulnerability was discovered by multiple parties in Group FaceTime which allowed an attacker to call a target and force the call to connect without user interaction from the target, allowing the attacker to listen to the target's surroundings without their knowledge or consent. While this remarkable bug was soon fixed, it presented a new and unresearched attack surface in mobile applications that support video conferencing. This presentation covers my attempts to find similar bugs in other messaging applications, including Signal, JioChat, Mocha, Google Duo, and Facebook Messenger.

Notes:

- *Detailed, yet concise abstract*
- *Defines a problem and offers a solution(s) that will be examined during the session.*

Presentation Outline

- Introduction
    - Explain FaceTime bug that caused call to be answered without user interaction
    - Goal: determine how these bugs happen and could they occur anywhere else
- WebRTC
    - What is WebRTC? (the video conferencing library used by most apps other than WhatsApp and Facetime)
    - How does WebRTC signaling work? What needs to happen for a microphone or camera to send content
- Analysis techniques (How to determine whether an app uses WebRTC and what the state machine looks like)
    - Documentation
    - Frida
    - Reversing with IDA or apktool
- Results
    - Signal and Facebook Messenger bugs
    - Both of these bugs allow audio to be transmitted without user consent
    - Both of these bugs occur due to not checking whether the software is in the correct state before moving to the next one
    - Mocha and JioChat bugs
    - Both bugs occur due to developers not understanding WebRTC functionality
    - Both bugs allow both audio and video to be transmitted without user consent
    - Google Duo bug
    - Occurred due to bad threading
    - Allowed the camera to take photos and transmit them with no user interaction
- Conclusions
    - Basically, everything I looked at had a similar bug to the Group FaceTime bug

    - Root causes are lack of attention to security in design, lack of understanding of video conferencing bugs and lack of understanding of this type of vulnerability
    - A lot more research is needed

Notes:

- *Clearly conveys progression of talk.*
- *Helps the Review Board visualize the presentation in its entirety.*
- *Gives an explanation of each area of the presentation.*

Attendee Takeaways (What are three actionable takeaways included in your presentation?)

1. Security researchers should investigate calling state machines when looking for bugs
2. Developers should be careful when implementing calling state machines as they are very vulnerability prone
3. Application design is very important to how these bugs work, and small changes can go a long way in preventing these types of bugs

Notes:

- *Submitter fulfilled requirement of providing 3 takeaways*
- *Explains relevance to the audience*
- *Cleary emphasizes the participant benefits*