Dark Firmware: A Systematic Approach to Exploring Application Security Risks in the Presence of Untrusted Firmware

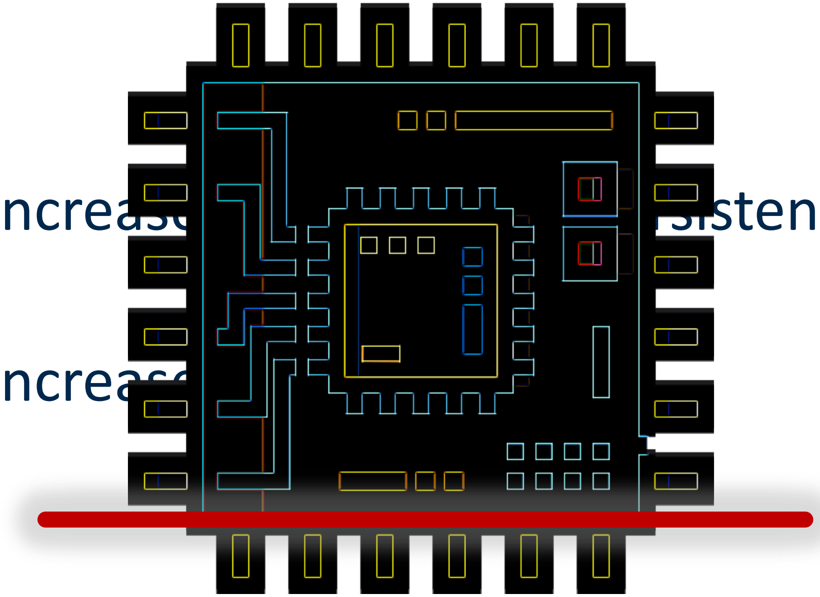
Duha Ibdah, Nada Lachtar, Abdulrahman Abu Elkhail, Anys Bacha, Hafiz Malik

UNIVERSITY OF MICHIGAN

Your Private Data is at Risk

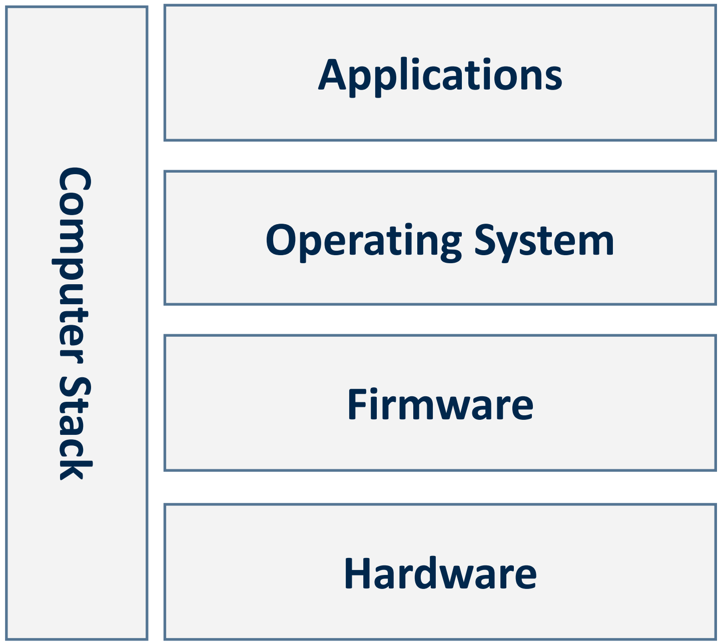
- Cloud based services are essential
- Increased concerns about confidentiality

The Quest for Persistent Malware

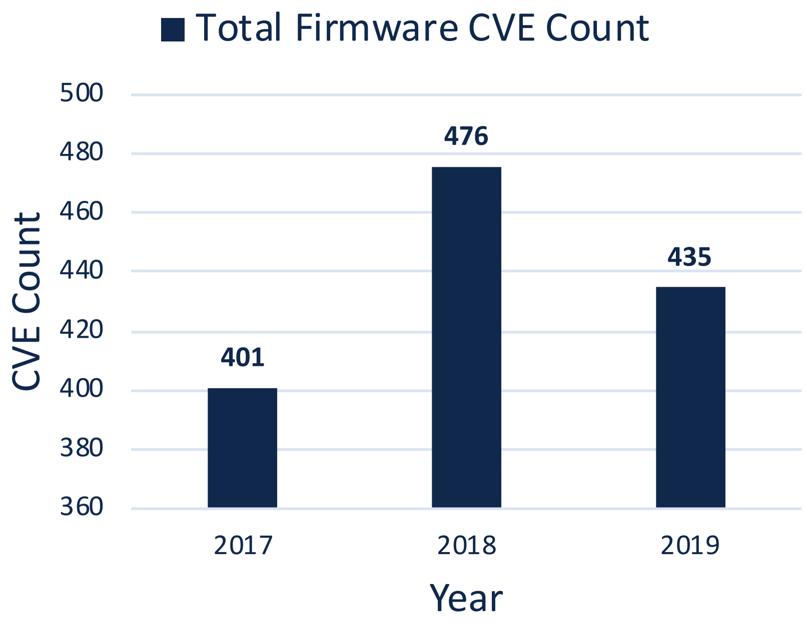


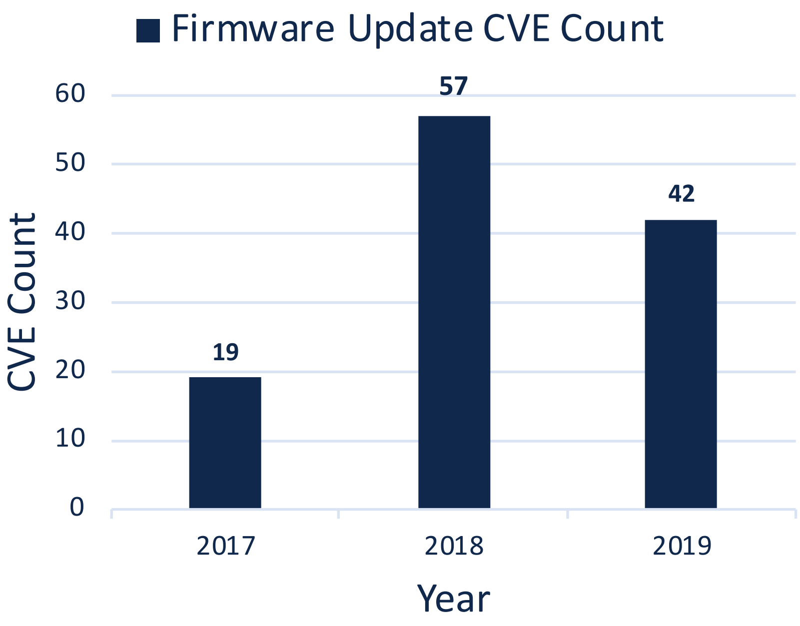
- Increas … sistence
- Increas

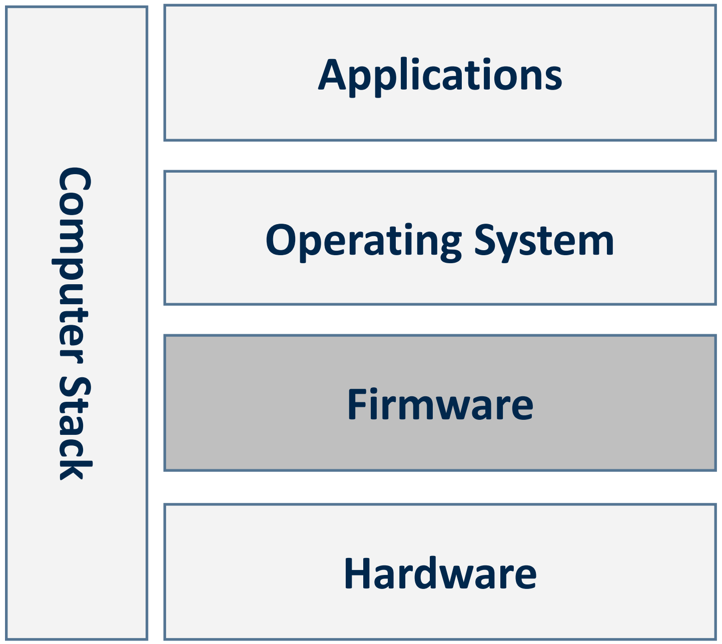
The Quest for Persistent Malware



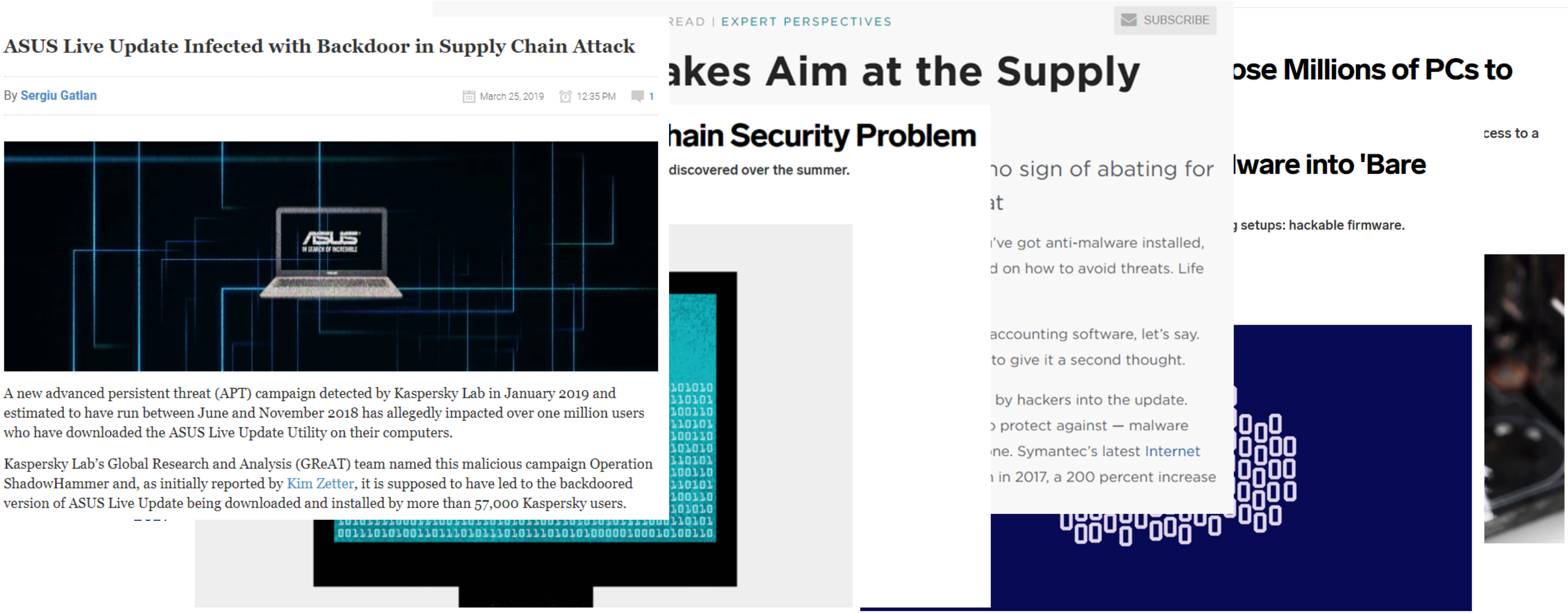
Firmware Vulnerabilities on the Rise







Firmware Vulnerabilities on the Rise



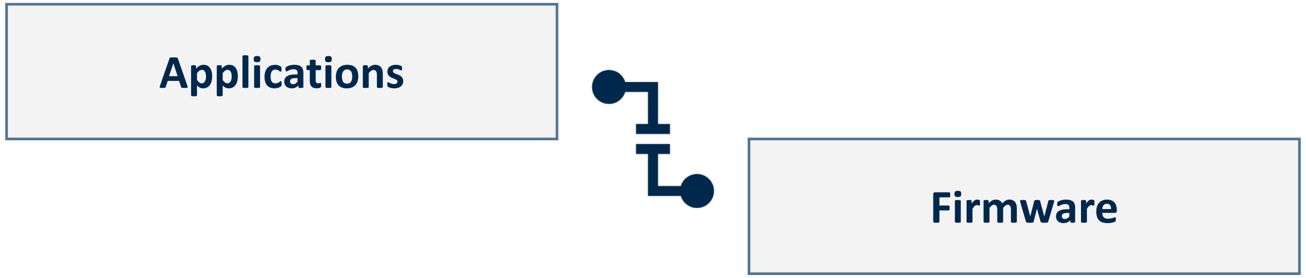
ASUS Live Update Infected with Backdoor in Supply Chain Attack

By Sergiu Gatlan March 25, 2019 12:35 PM 1

A new advanced persistent threat (APT) campaign detected by Kaspersky Lab in January 2019 and estimated to have run between June and November 2018 has allegedly impacted over one million users who have downloaded the ASUS Live Update Utility on their computers.

Kaspersky Lab's Global Research and Analysis (GReAT) team named this malicious campaign Operation ShadowHammer and, as initially reported by Kim Zetter, it is supposed to have led to the backdoored version of ASUS Live Update being downloaded and installed by more than 57,000 Kaspersky users.

Challenges with Firmware Attacks



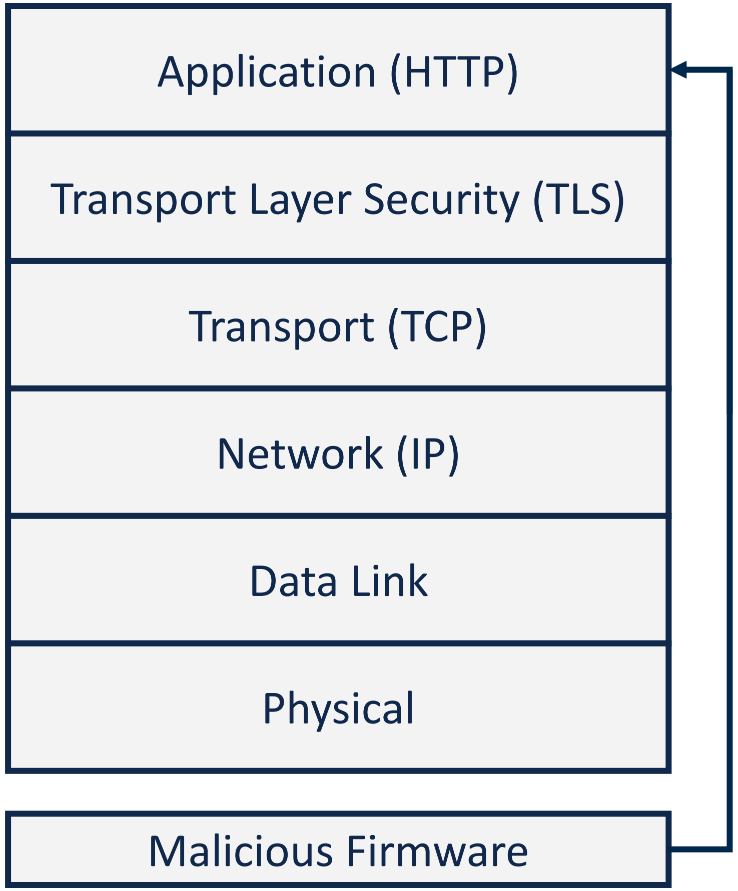
- Difficult to gain insight into application layer and how data is managed
- Limited execution cycles are allocated to firmware during runtime

Our Work

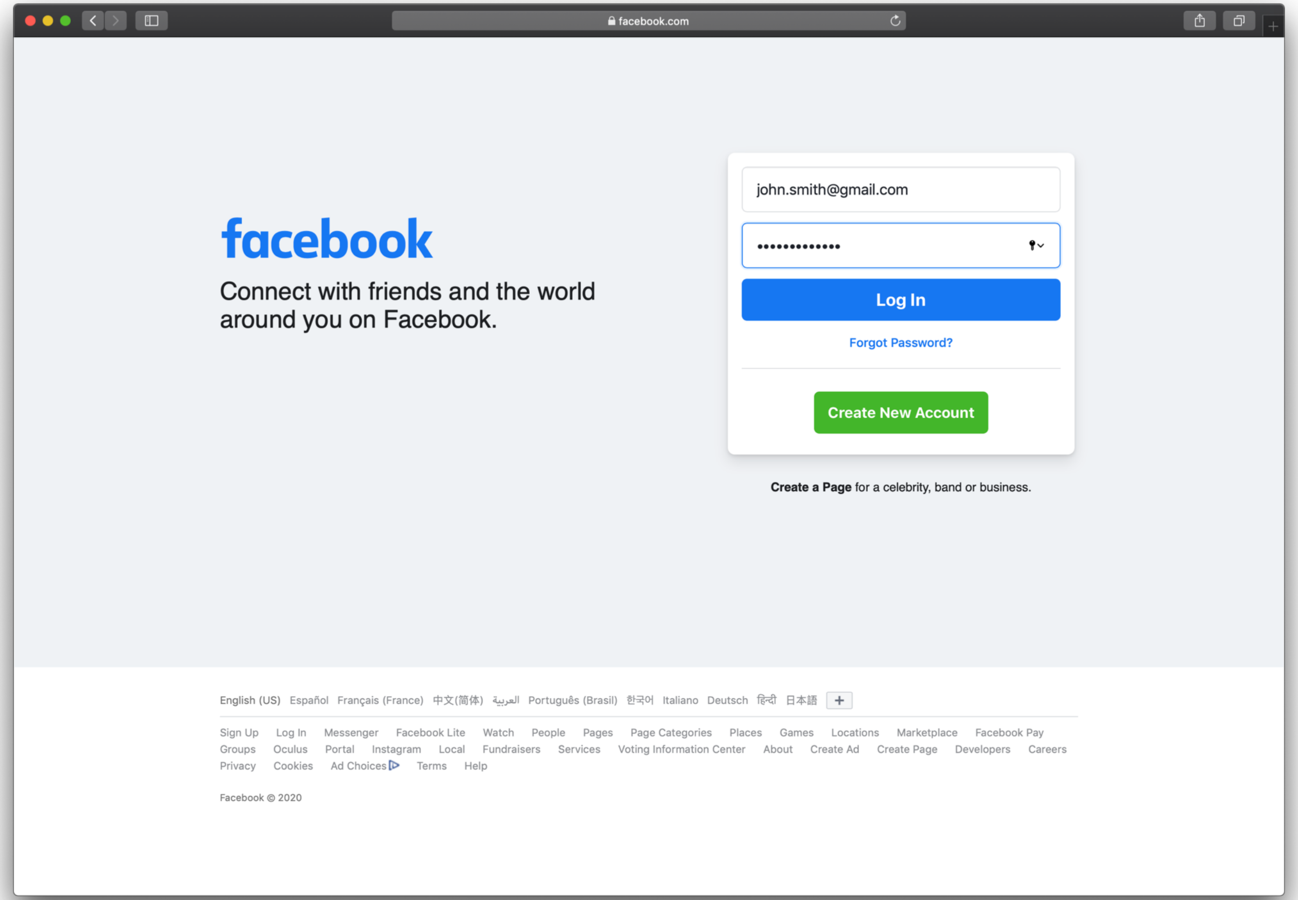
- Present a novel attack that harnesses platform management cycles to reliably and efficiently collect sensitive user data
- Conduct a proof-of-concept implementation of the attack using real firmware configured to run on desktop (Ubuntu) and mobile (Android) platforms
- Characterize the robustness of the proposed attack across desktop and mobile systems by extensively testing our attack under stressful app usage conditions
- Devise a low overhead mechanism that does not disrupt normal functionality by limiting its parsing to user accessible pages that are dirty

Harvesting Data in Presence of HTTPS

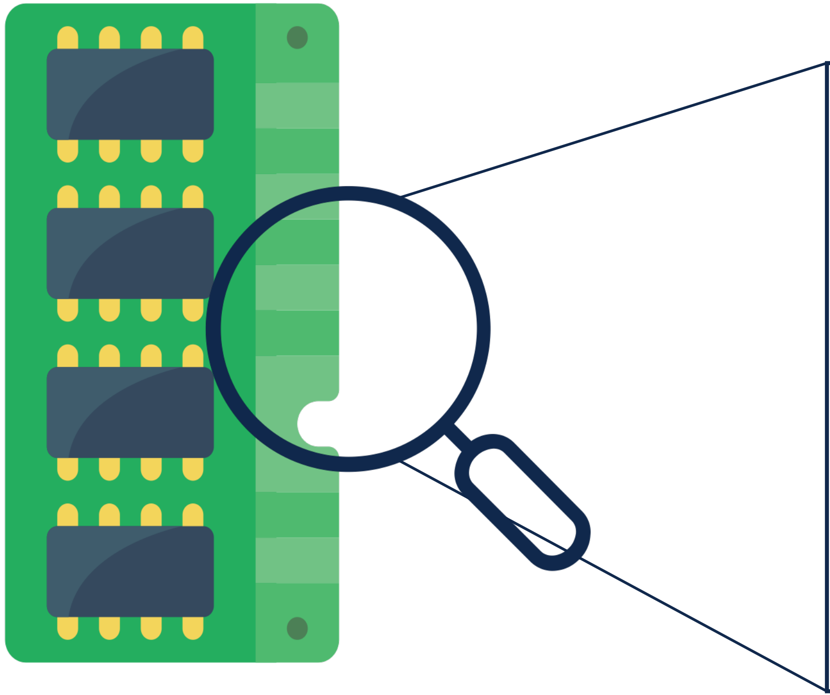
- Web services and mobile apps rely on HTTPS to securely exchange data
- HTTPS is achieved through adding a TLS layer
- Access data before encryption



Harvesting Data in Presence of HTTPS



Harvesting Data in Presence of HTTPS

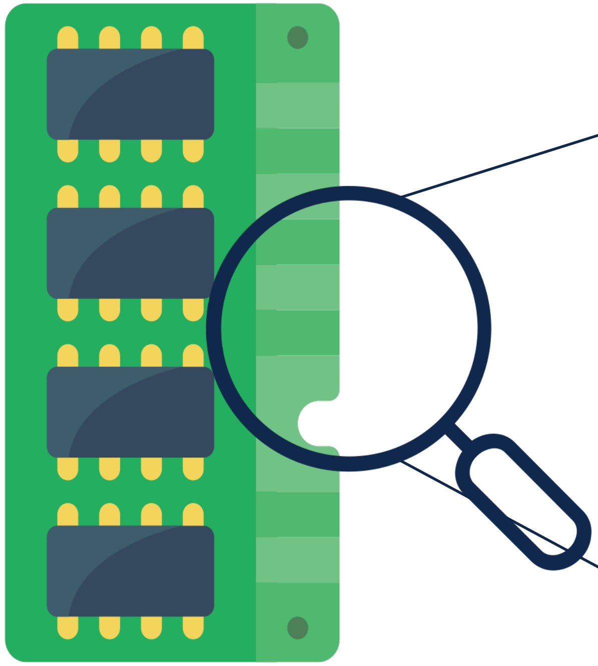


```
Host: www.facebook.com
Method: POST
Path: /login/device-based/regular/login
Content-Type: application/x-www-form-
urlencoded

jazoest=2697&lsd=AVrRLRxH&eemail=johnsmith
%40gmail.com&pass=pass123&timezone=300&lgn
dim=eyJ3IjoyNTYwLCJoIjoxNDQwLCJhdyI6MjU2MC
```

POST Request

Harvesting Data in Presence of HTTPS



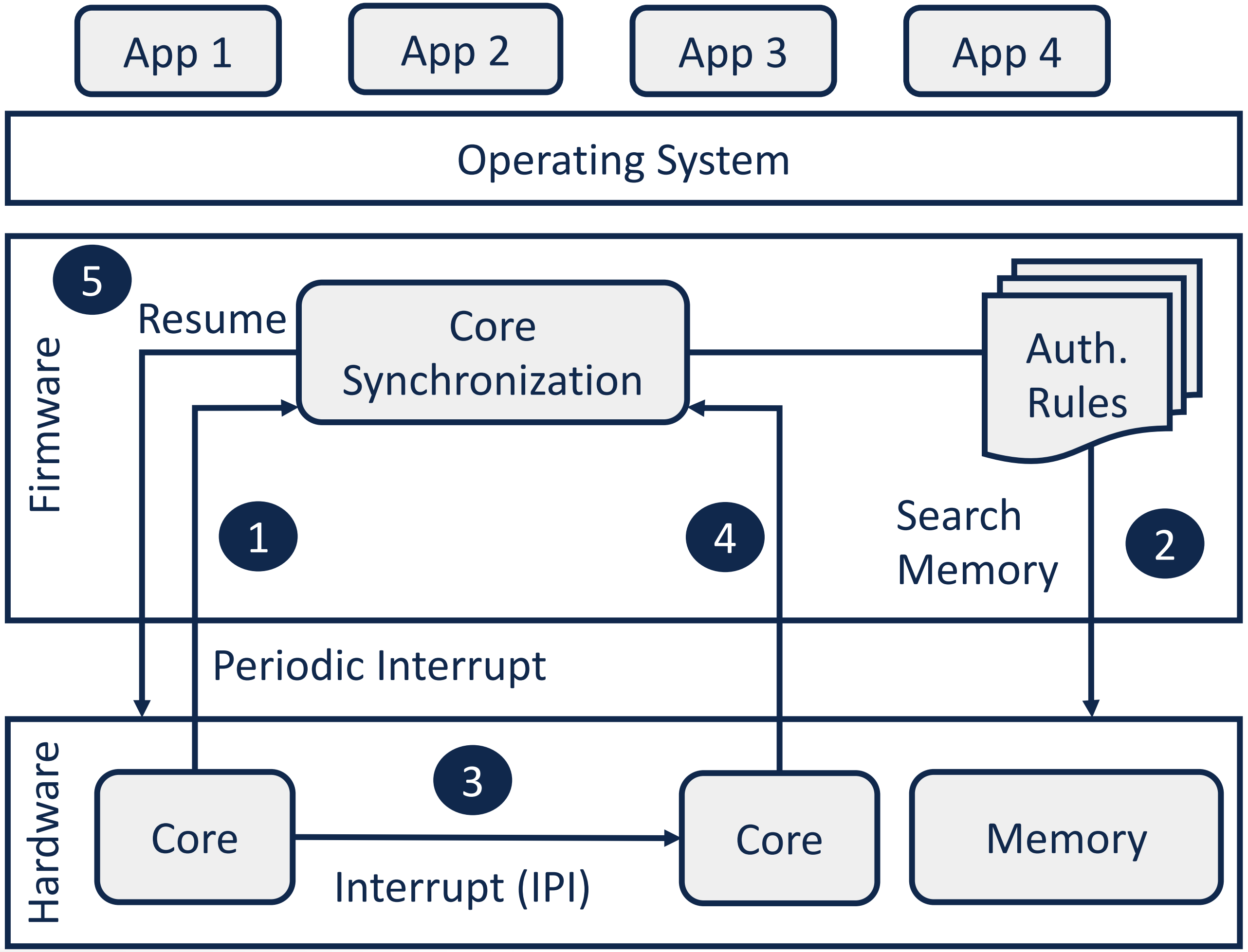
```
Host: www.facebook.com
Method: POST
Path: /login/device-based/regular/login
Content-Type: application/x-www-form-
urlencoded

jazoest=2697&lsd=AVrRLRxH&eemail=johnsmith
%40gmail.com&pass=pass123&timezone=300&lgn
dim=eyJ3IjoyNTYwLCJoIjoxNDQwLCJhdyI6MjU2MC
```

POST Request

Dark Firmware: Our Proposed Attack

- Hardware interrupt invokes CPU core
- CPU core parses memory for HTTP requests
- IPI is issued to the next core
- Cores synchronize to continue the search process
- New core resumes memory parsing



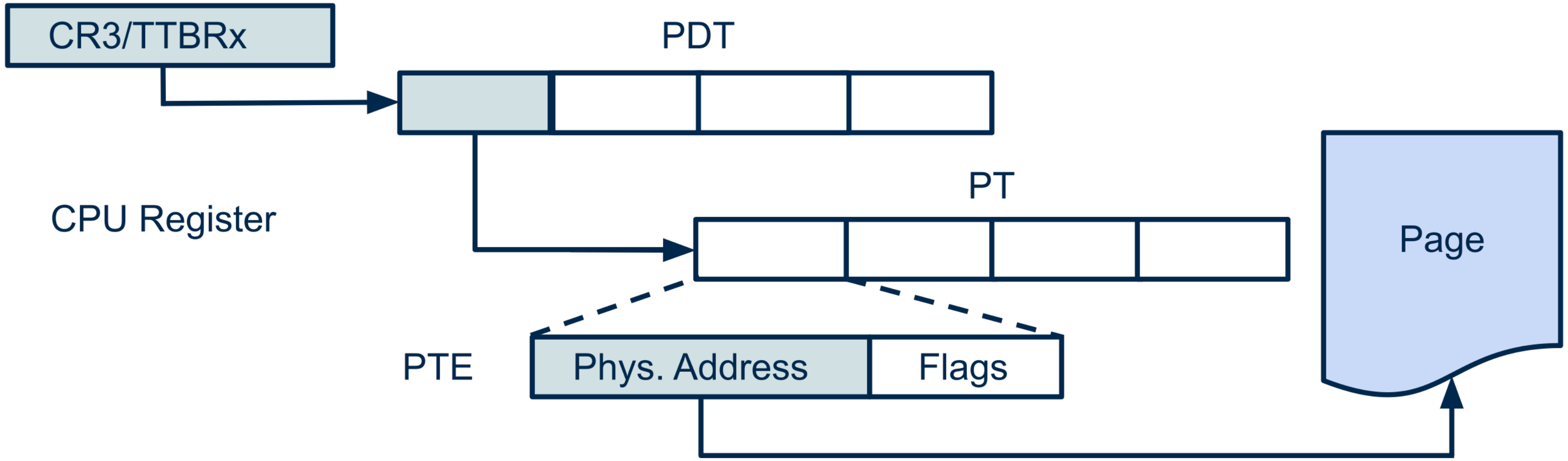
Dark Firmware: Our Proposed Attack

- Hardware interrupt invokes CPU core.
- CPU core [illegible] requests.
- IPI is issued to the next core
- Cores synchronize to continue the search process.
- New core resumes memory parsing.

Full Memory Search is Expensive

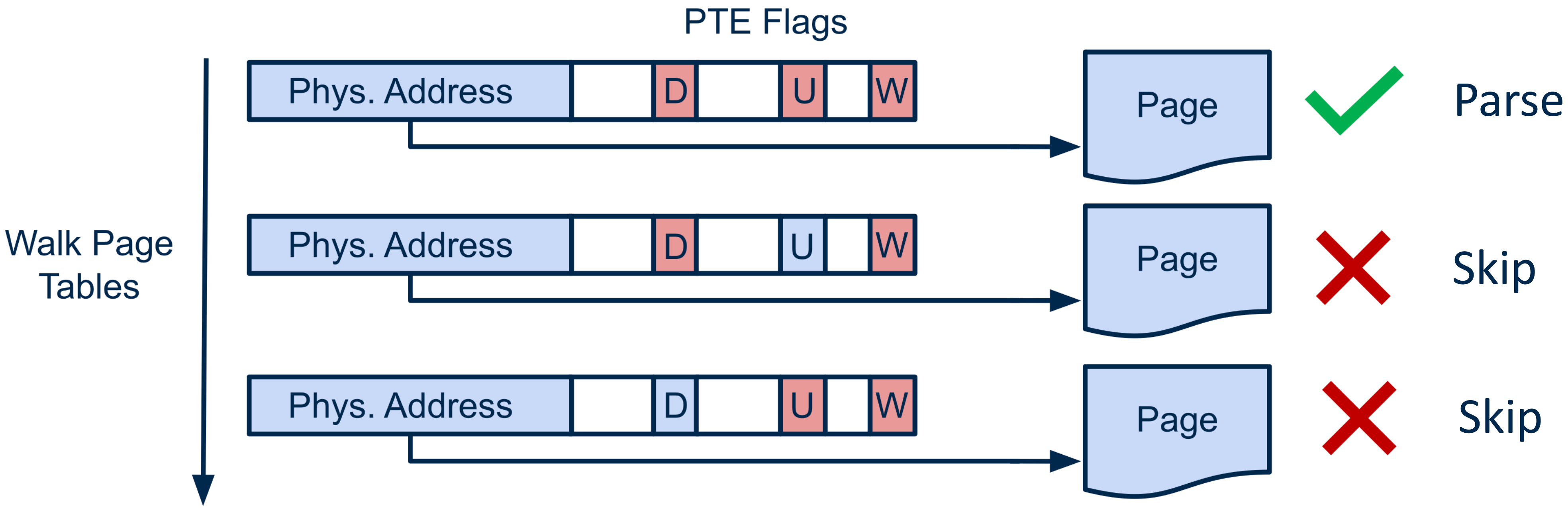
Dark Firmware: A Page-Aware Approach

- Selectively search memory based on flags of page table entries



Dark Firmware: A Page-Aware Approach

- Selectively search memory based on flags of page table entries
- CPU only parses pages with D, U, and W flags set



Experimental Framework

- Real firmware based on UEFI from the Tianocore open source project
- Attack tested on desktop (Ubuntu 18.04 LTS) and mobile (Android 8.1) systems
- QEMU 3.0.50 used for testing multiple platform configurations

Experimental Framework

Category	Desktop Applications (Ubuntu)
Social	Corebird (Twitter), Reddit, LinkedIn, Pinterest, Facebook, Twitter, Linkedin, Pinterist Ramme (Instagram), Tumblr, Nextdoor, Wattpad
Communication	Slack, Skype, Signal, Whatsdesk (WhatsApp), Discord, Viber
Productivity	Calc (Excel), Impress (Power Point), Writer (Word), Draw (Visio), Gimp, Gmail, Dropbox, Calendar, Todoist PDF, Overleaf, Gmail, Thunderbird, Calendar, Dropbox, Box, Peek (Screen Recorder), Everpad (Evernote), Android Studio, GitKraken, Eclipse, VirtualBox, Toggl, Qualtrics
Travel & Local	Airbnb, Google Maps, TripAdvisor, Expedia Travel, Uber, Google Maps, TripAdvisor, Uber, Yelp, Lyft, Grubhub
Health & fitness	WebMD, LiveStrong, MyFitnessPal
Entertainment	YouTube, Angry Birds, Candy Crush, Spotify, Steam

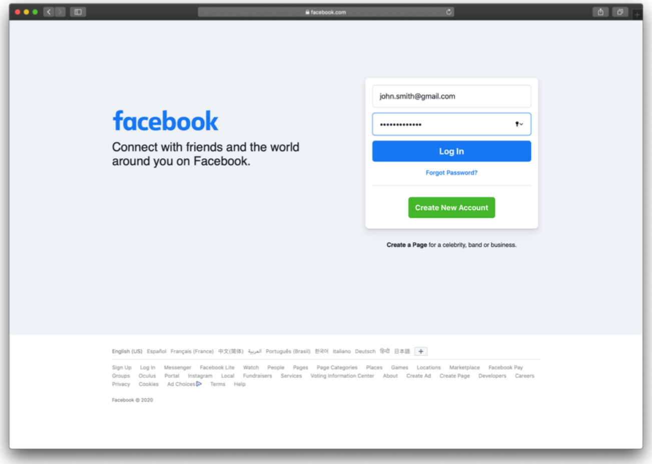
Persistence vs. Application Usage

- Tested the persistence of authentication data after launching different application mixes
- Each application had different memory requirements and stressed the memory subsystem differently

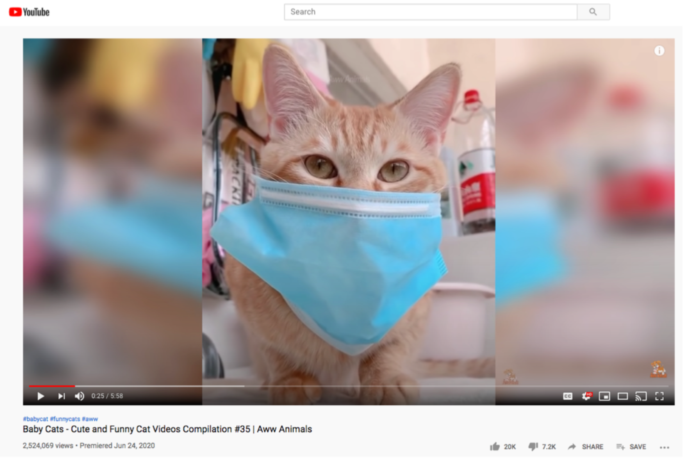
Mix	Category Sequence
1	social, communication, productivity, travel & local, health & fitness, entertainment
2	communication, productivity, travel & local, health & fitness, entertainment, social
3	productivity, travel & local, health & fitness, entertainment, social, communication
4	travel & local, health & fitness, entertainment, social, communication, productivity
5	health & fitness, entertainment, social, communication, productivity, travel & local
6	entertainment, social, communication, productivity, travel & local, health & fitness

Persistence vs. Application Usage

1 Log in to Facebook



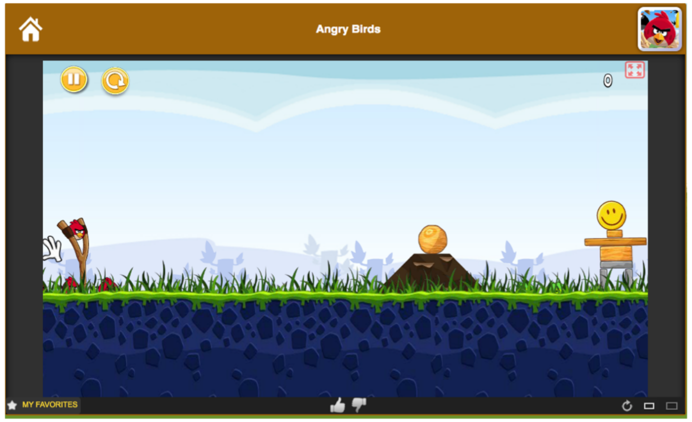
2 Watch cat videos for 30 seconds



Search for password

✓

3 Play Angry Birds

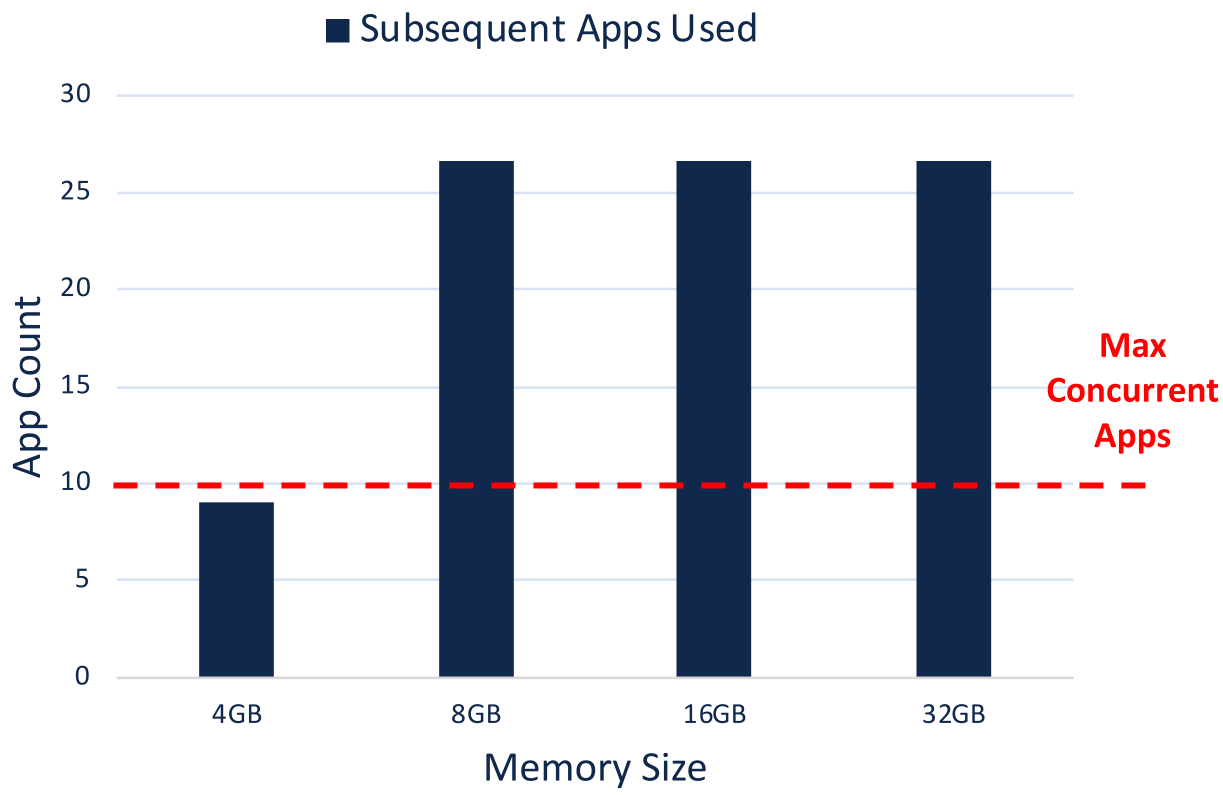


Search for password

✗

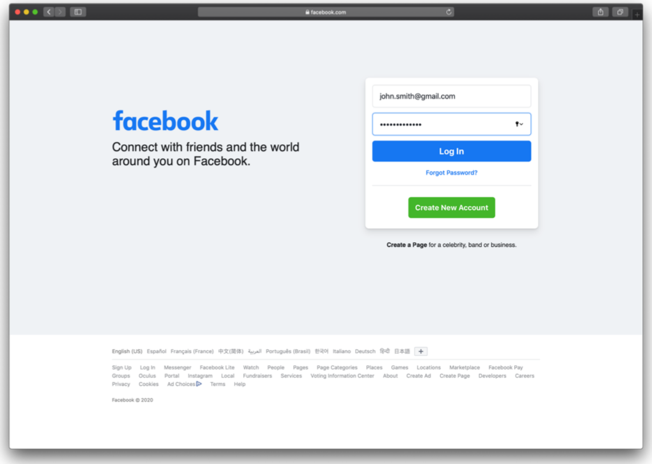
Persistence vs. Application Usage

- Users typically run no more than 10 applications concurrently
- Attack was successful after utilizing an average of 22 apps
- 3x increase in vulnerability factor when doubling memory capacity from 4GB to 8GB

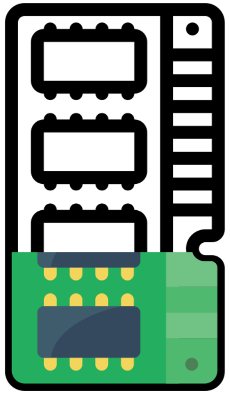


Persistence vs. Memory Consumption

1 Log in to Facebook

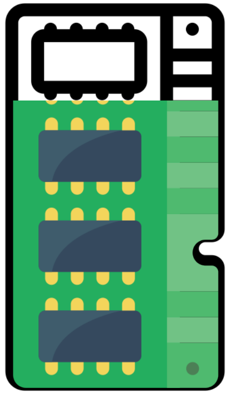


2 Increase memory stress a 100 MB/minute



Search for password

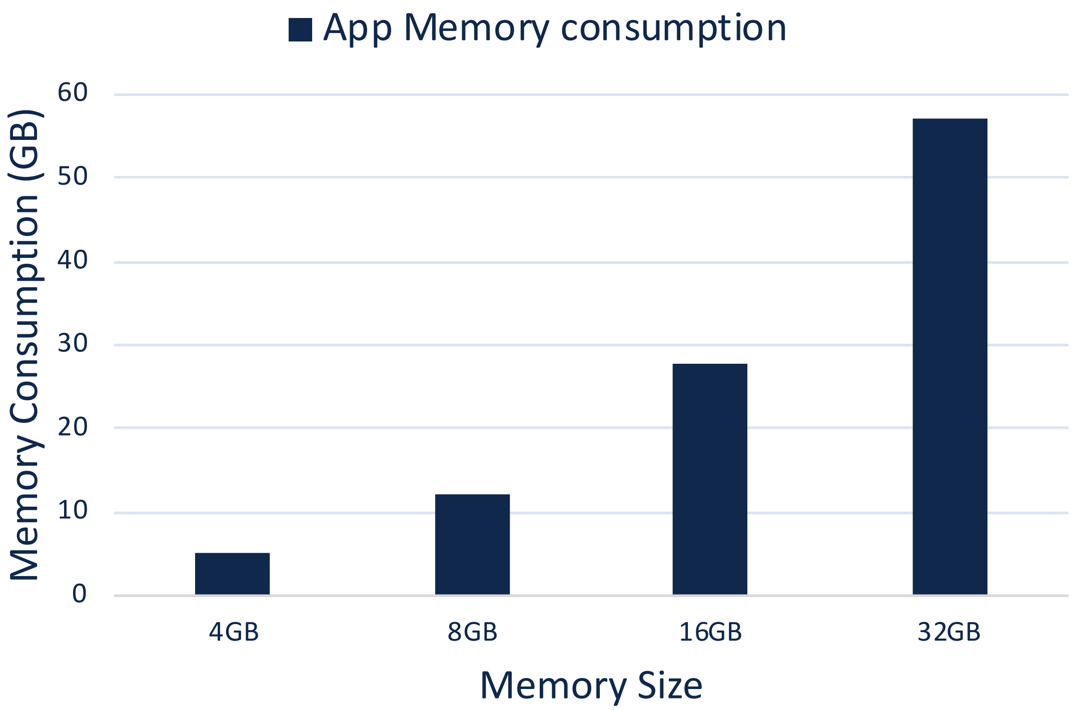
3



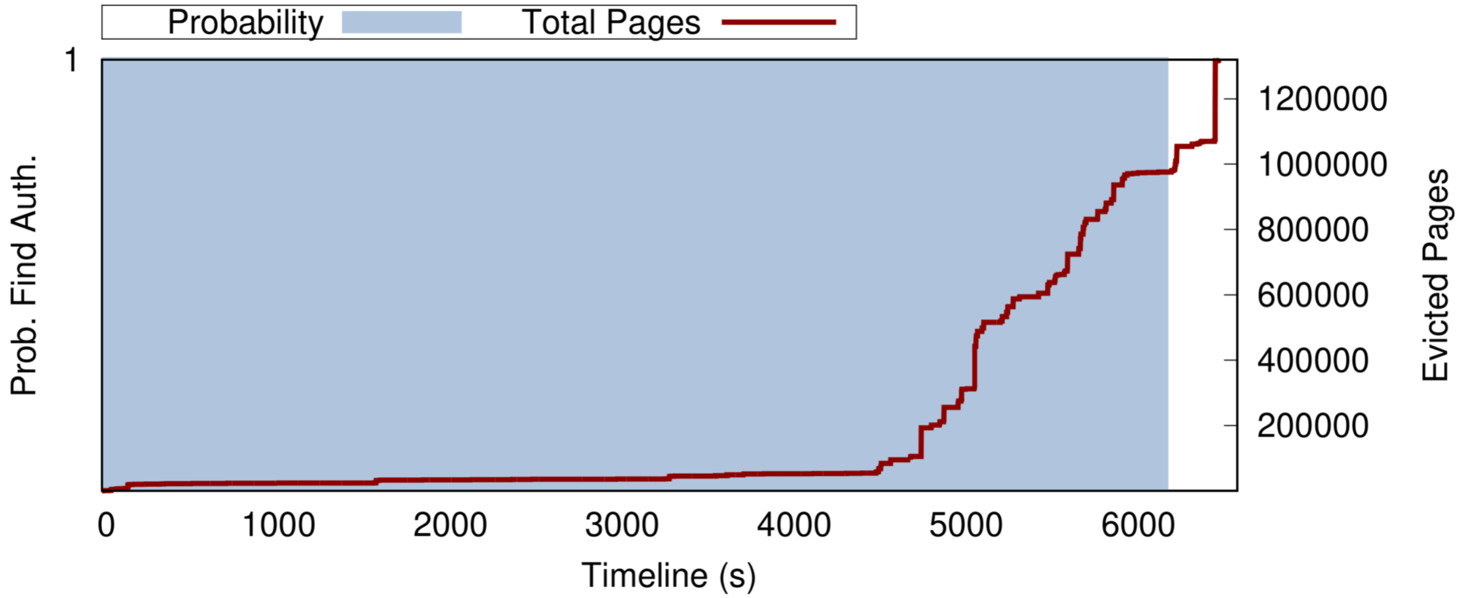
Search for password

Persistence vs. Memory Consumption

- Persistence of the credentials depend on the memory consumption
- Eviction occurs after consuming at least 150% of memory capacity
- A significant amount of swap space is consumed before credentials are evicted

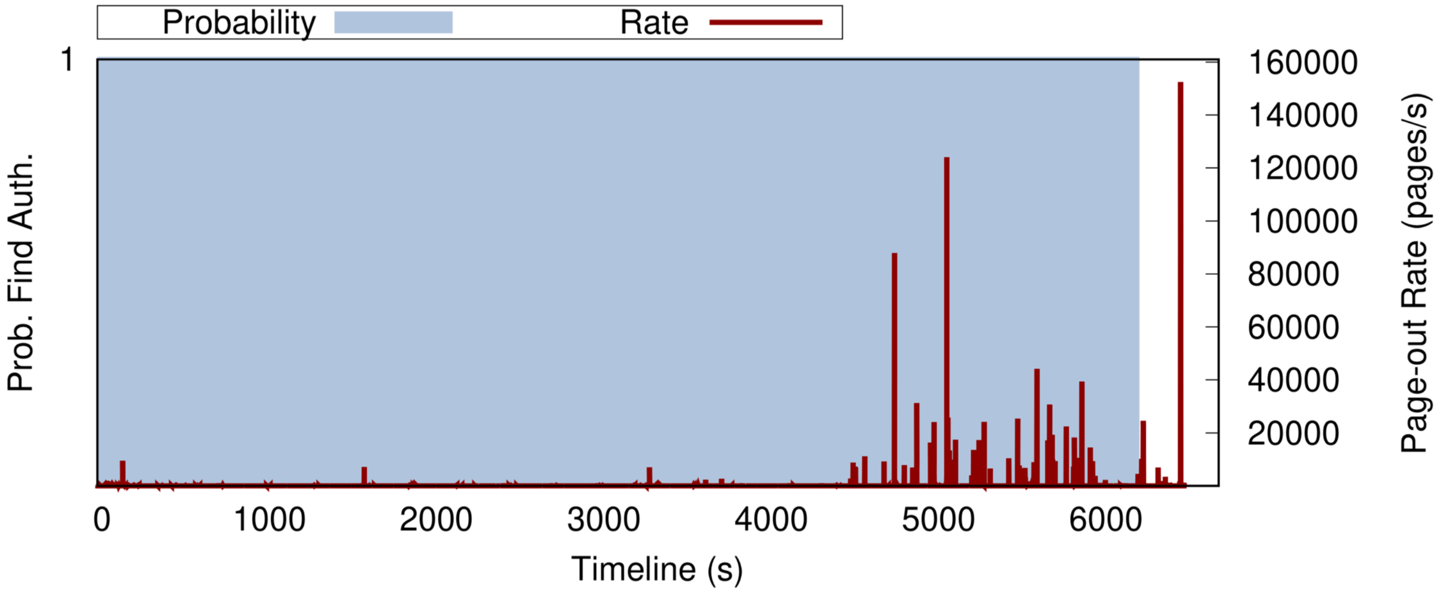


Persistence vs. Memory Consumption



- Firmware was unable to locate credentials after 900K pages evicted (after 104 min)

Persistence vs. Memory Consumption

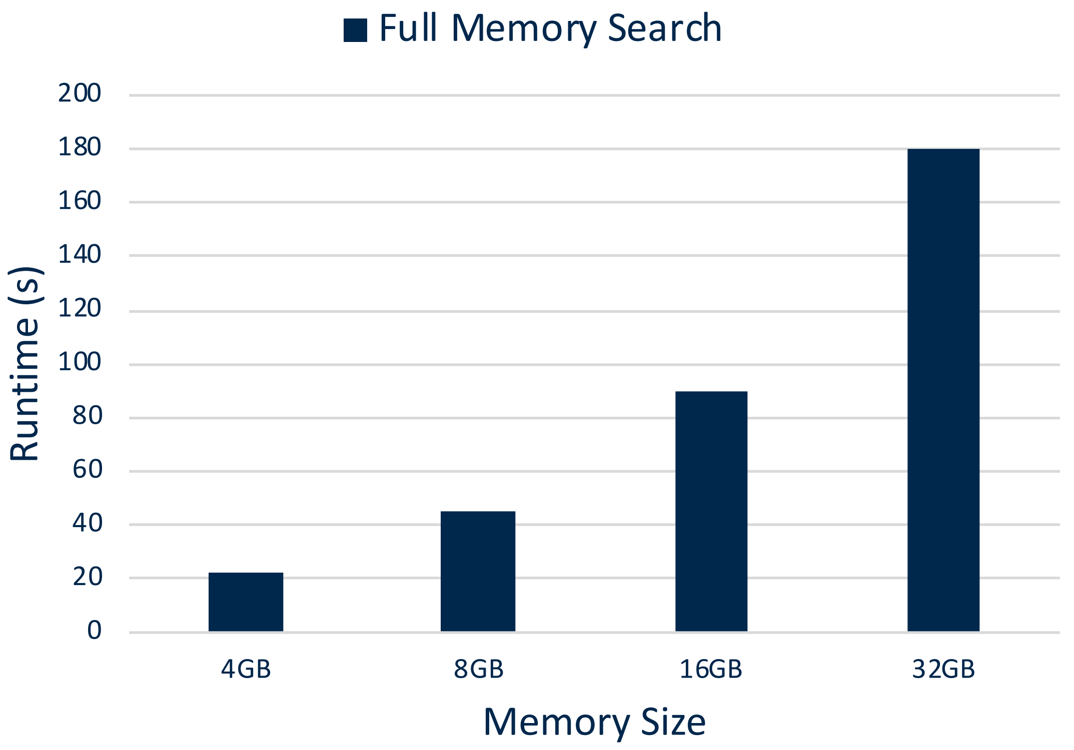


- Aggressive page-out rate was observed before credential eviction

Performance Analysis

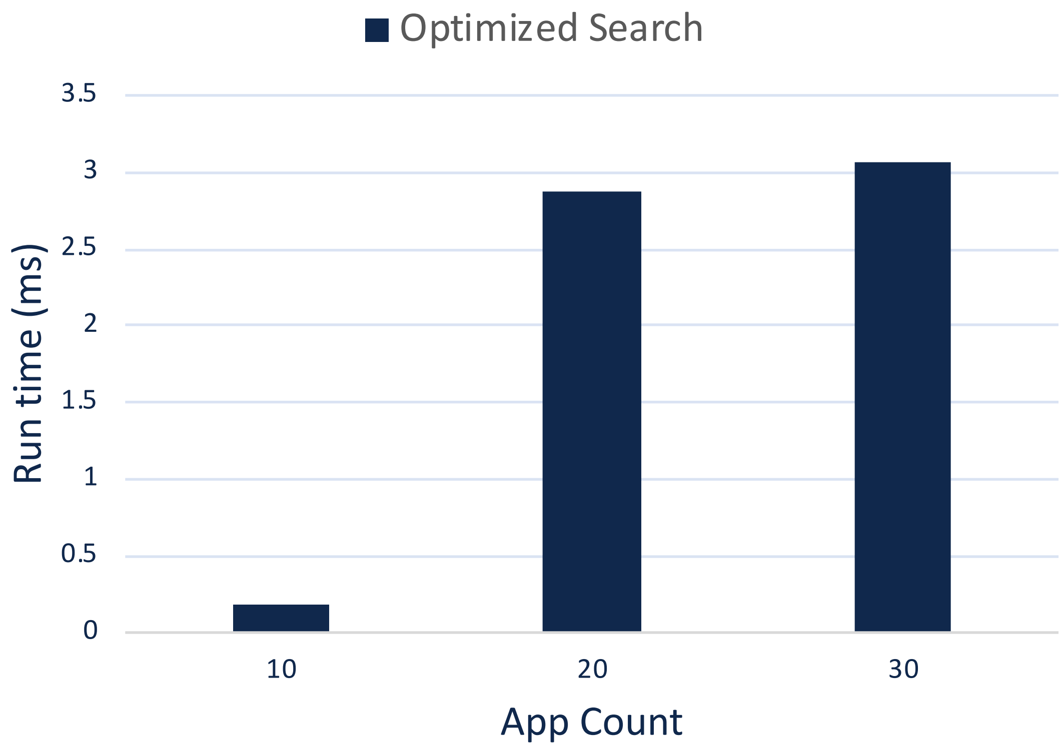
- Performance overhead increases linearly with memory capacity when doing full search

- Naïve approach doesn't scale well to larger memory capacities



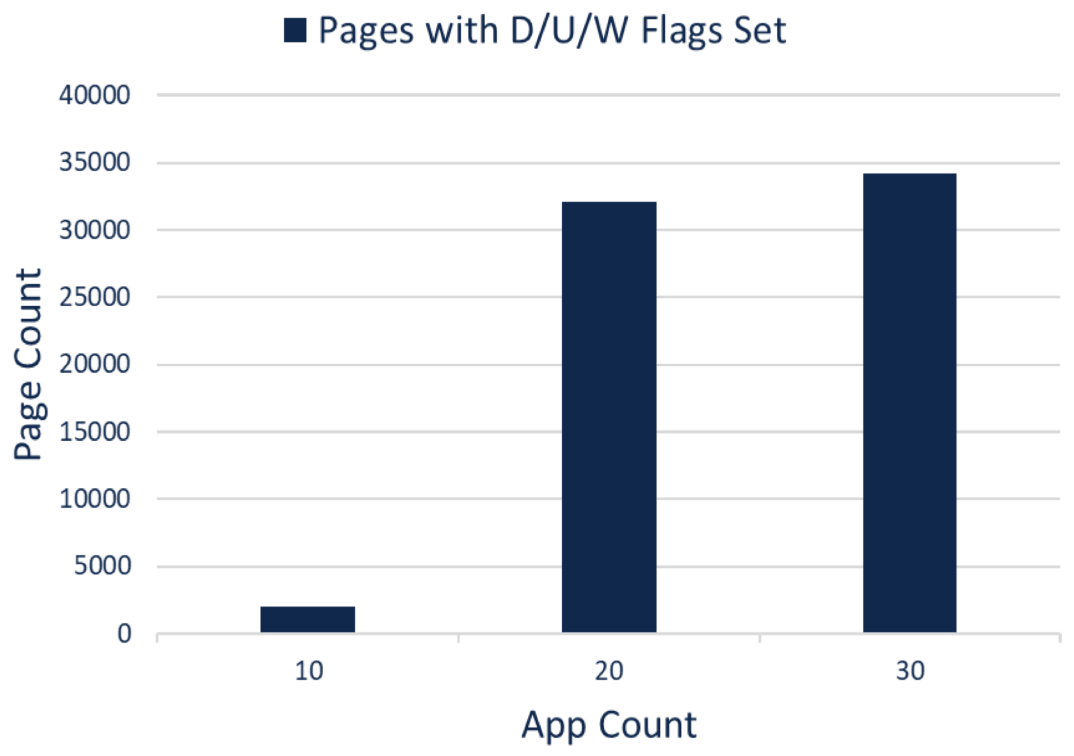
Performance Analysis

- Overhead increases as more applications are launched, irrespective of the memory capacity



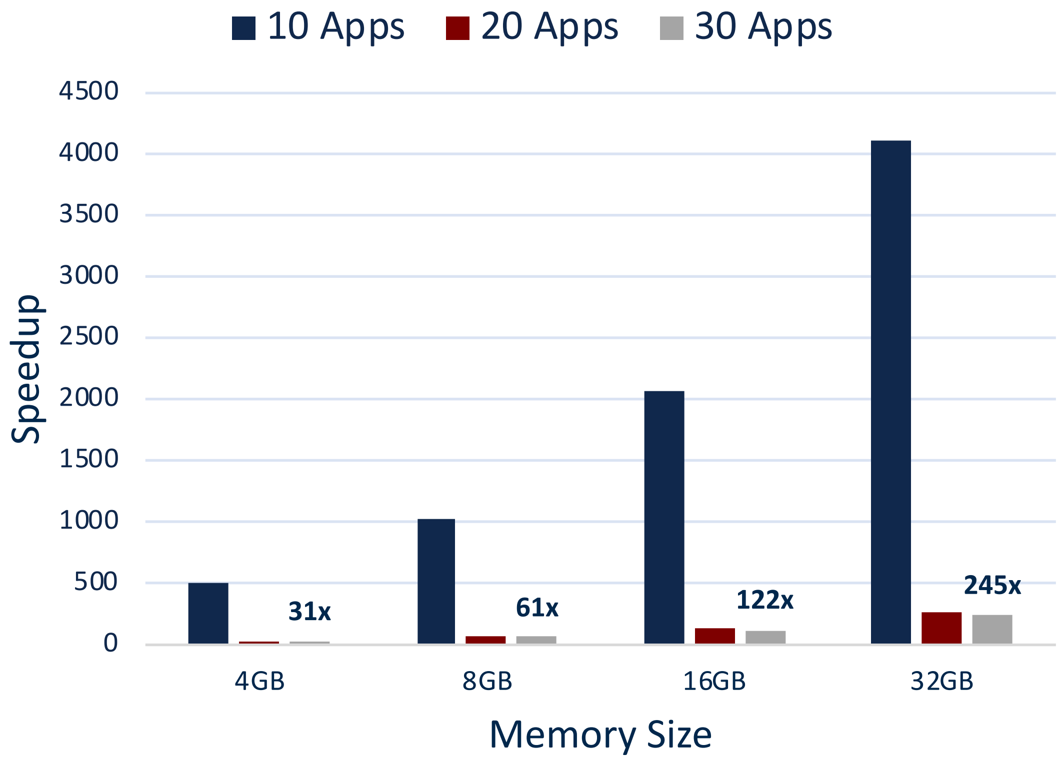
Performance Analysis

- The number of searched pages is invariant to the memory capacity
- The number of searched pages scales with the number of running apps



Performance Analysis

- Our selective search significantly improves search time



Conclusion

- Propose an attack that covertly leverages platform management cycles to extract sensitive data from the application layer
- Demonstrate that firmware can reliably extract sensitive data without disrupting the normal execution of launched applications
- Discuss a page-aware approach that is up to $4.\ 10^3$x faster than a full memory search implementation
- Realize a proof of concept to show the attack is practical

UNIVERSITY OF MICHIGAN

Questions?

Backup

Related Work

Reference, Year	Approach	Limitations
[1], 2011	[1] Introspection through hypervisors for extracting sensitive data	Rely on hypervisors whereas our firmware attack can exist on every computing device
[2], 2019 [3], 2016 [4], 2014 [5], 2014	[2-5] Installing malicious peripherals for reading memory through DMA attacks.	Such attack requires physical access while our attack can be carried out remotely through firmware update or supply chain
[6], 2015 [7], 2014 [8], 2017	[6] Altering firmware through overcoming write protection mechanism [7] Injecting malicious firmware through updates [8] Injecting malicious firmware through overcoming security features such as secure boot	We leverage the attacks [6-8] to insert untrusted firmware into the system examine application layer in the presence of untrusted firmware

References

[1] J. Hizver and T.-c. Chiueh, “An introspection-based memory scraper attack against virtualized point of sale systems,” in International Conference on Financial Cryptography and Data Security. Springer, 2011, pp. 55–69.

[2] A. T. Markettos, C. Rothwell, B. F. Gutstein, A. Pearce, P. G. Neumann, S. W. Moore, and R. N. Watson, “Thunderclap: Exploring vulnerabilities in operating system iommu protection via dma from untrustworthy peripherals.” in NDSS, 2019.

[3] U. Frisk, “Dma attacking over usb-c and thunderbolt 3,” Blog, October 2016, http://blog.frizk.net/2016/10/dmaattacking- over-usb-c-and.html.

[4] U. Frisk, “Direct memory attack the kernel,” Proceedings of DEFCON, vol. 24, 2016.

[5] J. FitzPatrick and M. Crabill, “Stupid pcie tricks,” 2014.

[6] C. Kallenberg and R. Wojtczuk, “Speed racer: Exploiting an intel flash protection race condition,” 2015, white Paper.

[7] C. Kallenberg, J, X. Kovah. “Bios necromancy: Utilizing “dead code” for bios attacks,” in Hack in the Box, 2014.

[8] A. Matrosov and E. Rodionov, “Uefi firmware rootkits: Myths and reality,” in Black Hat Asia, 2017.