2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

# Introduction of CHERI and how it works

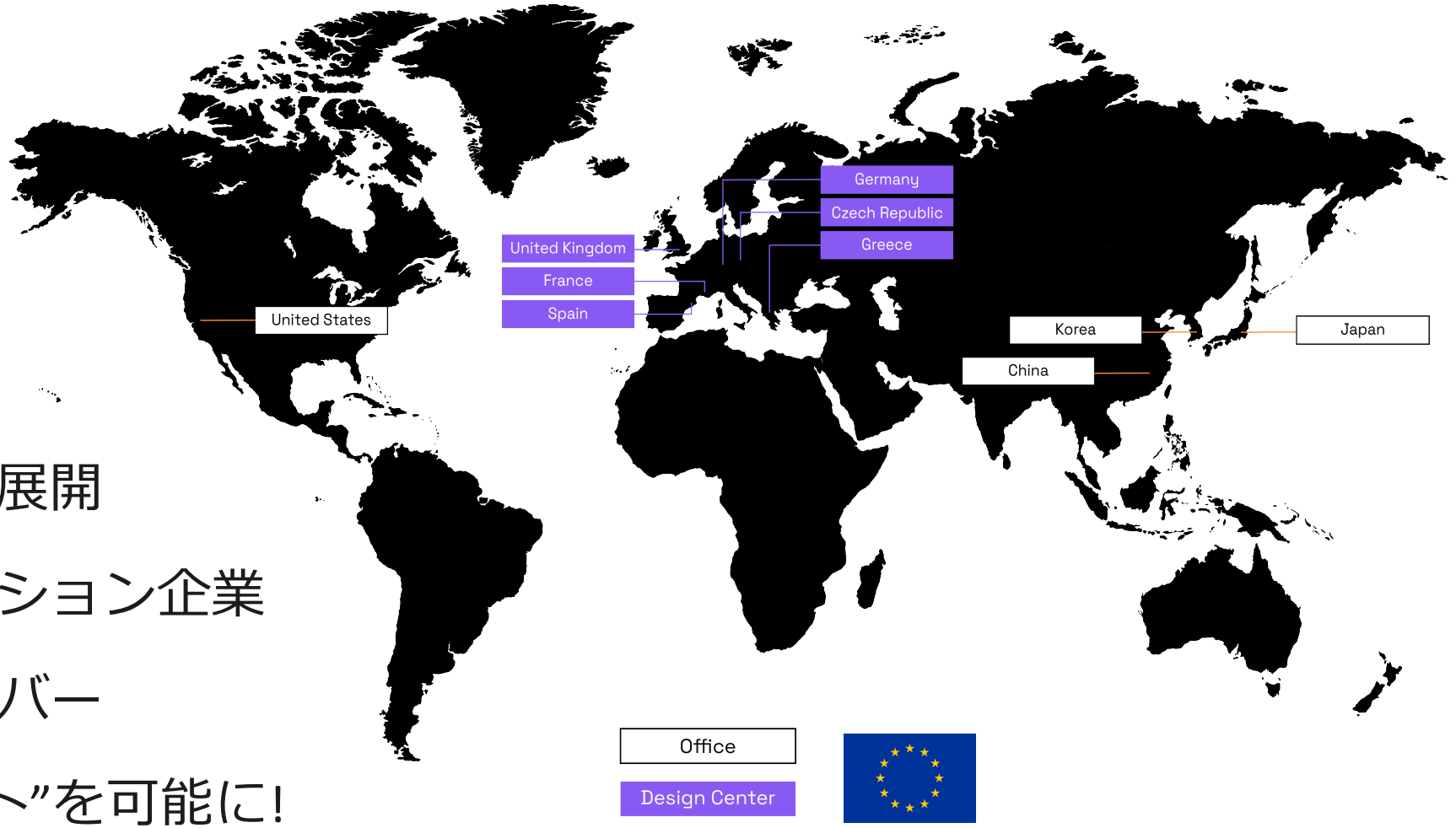
Takaaki Akashi, Country Manager - Japan

Codasip

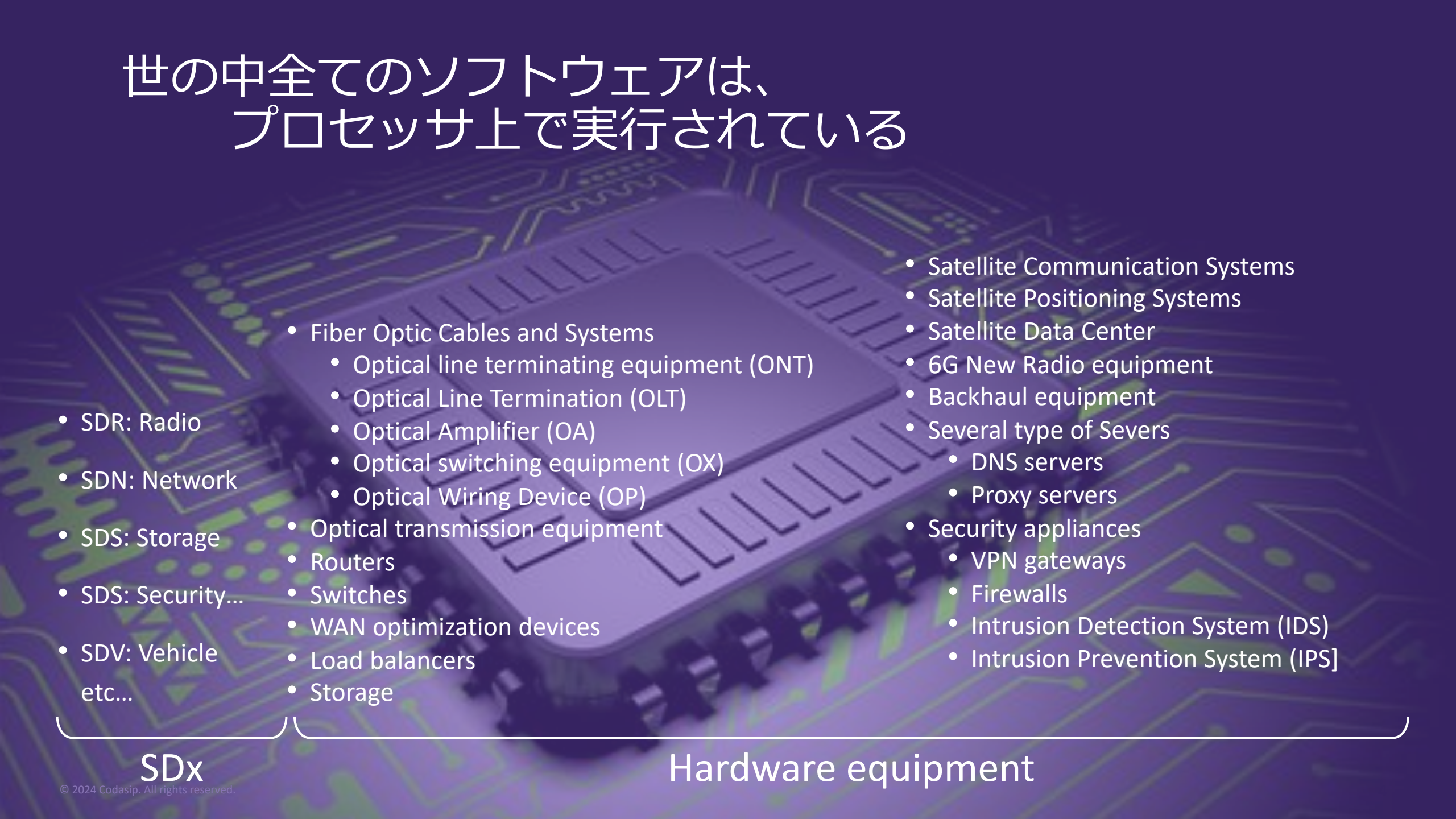
accellera SYSTEMS INITIATIVE

# → コダシップ概要

- 2014年設立
- 独ミュンヘン本社
- 約250名の社員
- ヨーロッパで開発
- グローバルなオフィス展開
- プロセッサ・ソリューション企業
- RISC-V創設メンバー
- “カスタム コンピュート”を可能に!



# 世の中全てのソフトウェアは、プロセッサ上で実行されている



- SDR: Radio
- SDN: Network
- SDS: Storage
- SDS: Security…
- SDV: Vehicle

etc…

SDx

- Fiber Optic Cables and Systems
  - Optical line terminating equipment (ONT)
  - Optical Line Termination (OLT)
  - Optical Amplifier (OA)
  - Optical switching equipment (OX)
  - Optical Wiring Device (OP)
- Optical transmission equipment
- Routers
- Switches
- WAN optimization devices
- Load balancers
- Storage

- Satellite Communication Systems
- Satellite Positioning Systems
- Satellite Data Center
- 6G New Radio equipment
- Backhaul equipment
- Several type of Severs
  - DNS servers
  - Proxy servers
- Security appliances
  - VPN gateways
  - Firewalls
  - Intrusion Detection System (IDS)
  - Intrusion Prevention System (IPS]

Hardware equipment

# → サイバーセキュリティ対策には非常にコストがかかる

Codasip

**>$500M**
**(750億円)**

Heatbleed
バッファーオーバーフロー
への対策コスト

**6X**

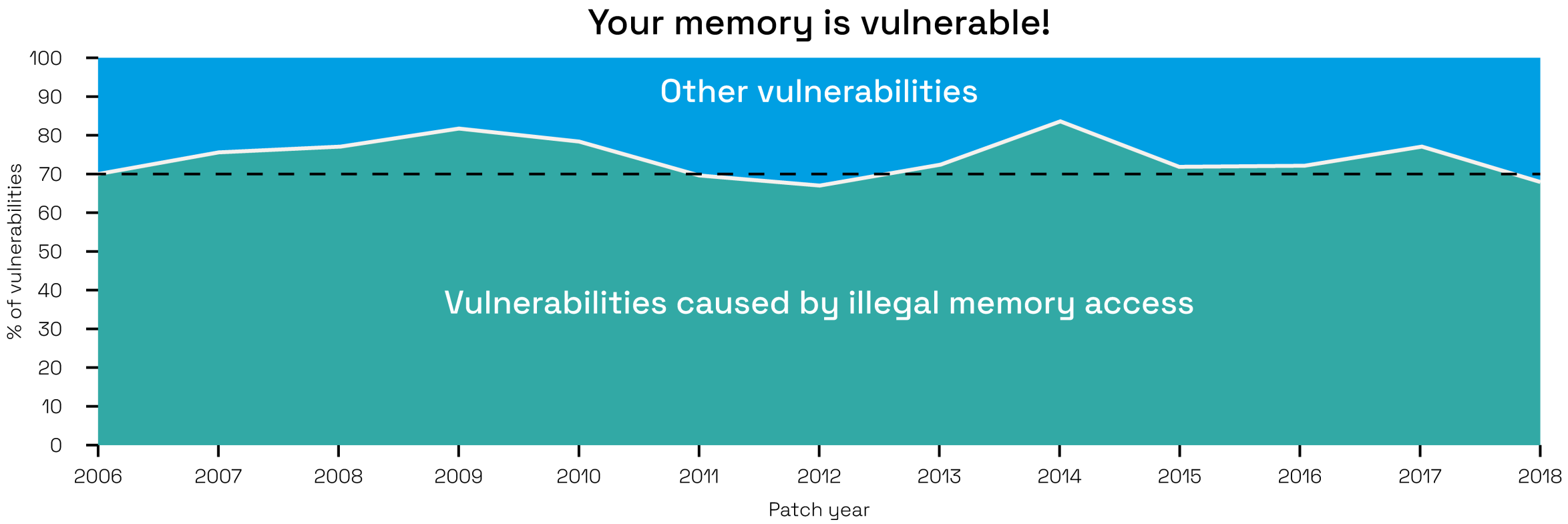
2017年から2021年にかけて
NISTが報告した
ファームウェア攻撃の増加率

**~$10T**
**(1,500兆円)**

サイバー攻撃による全世界の
年間被害額
(and growing fast)

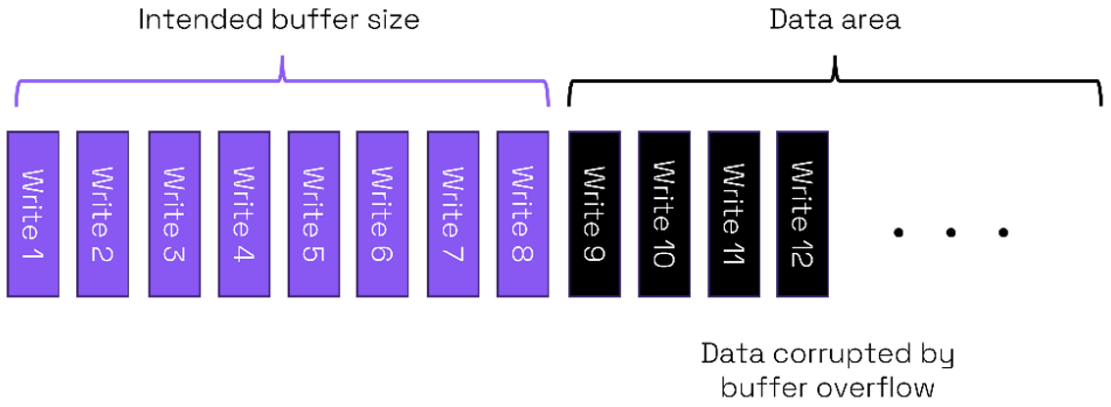
# → 脆弱性の70%はメモリ起因

Codasip

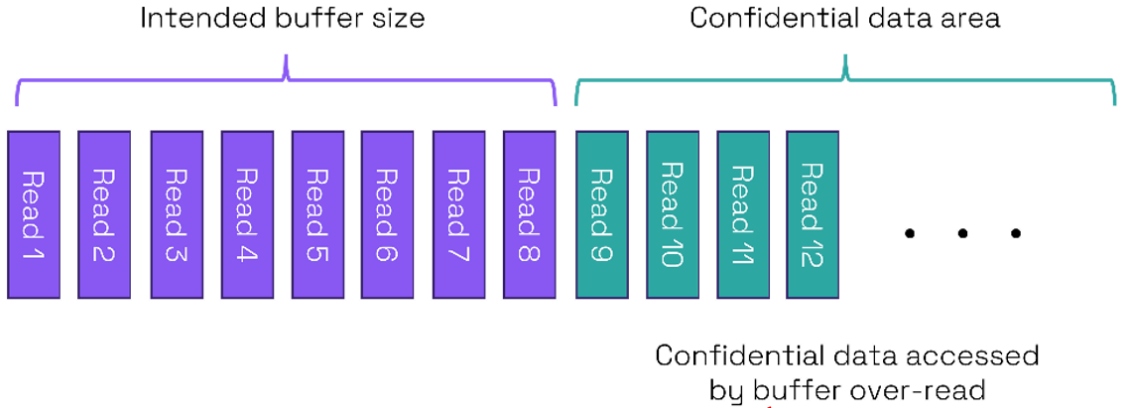



Source: "Trends, challenges and shifts in vulnerability mitigation", Matt Miller (MSRC), BlueHat IL 2019.

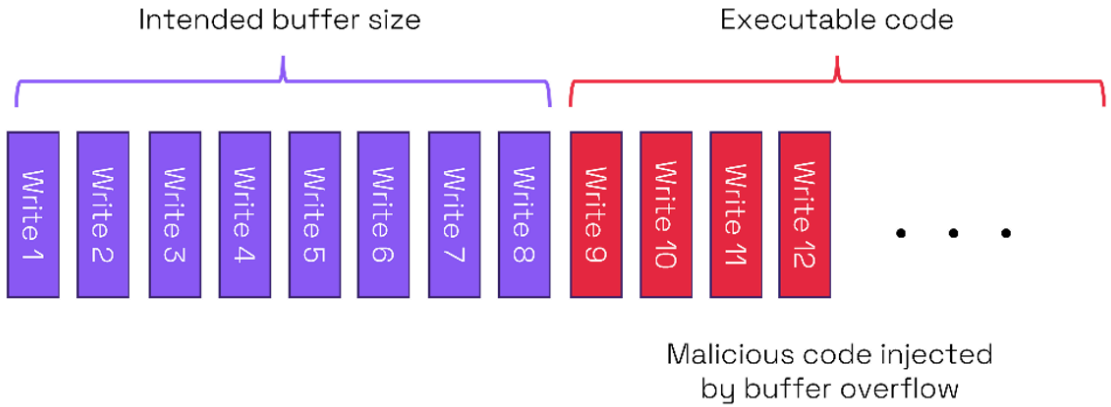
# → バッファ オーバーフロー…




オーバーフロー




オーバーリード




悪意コード実行

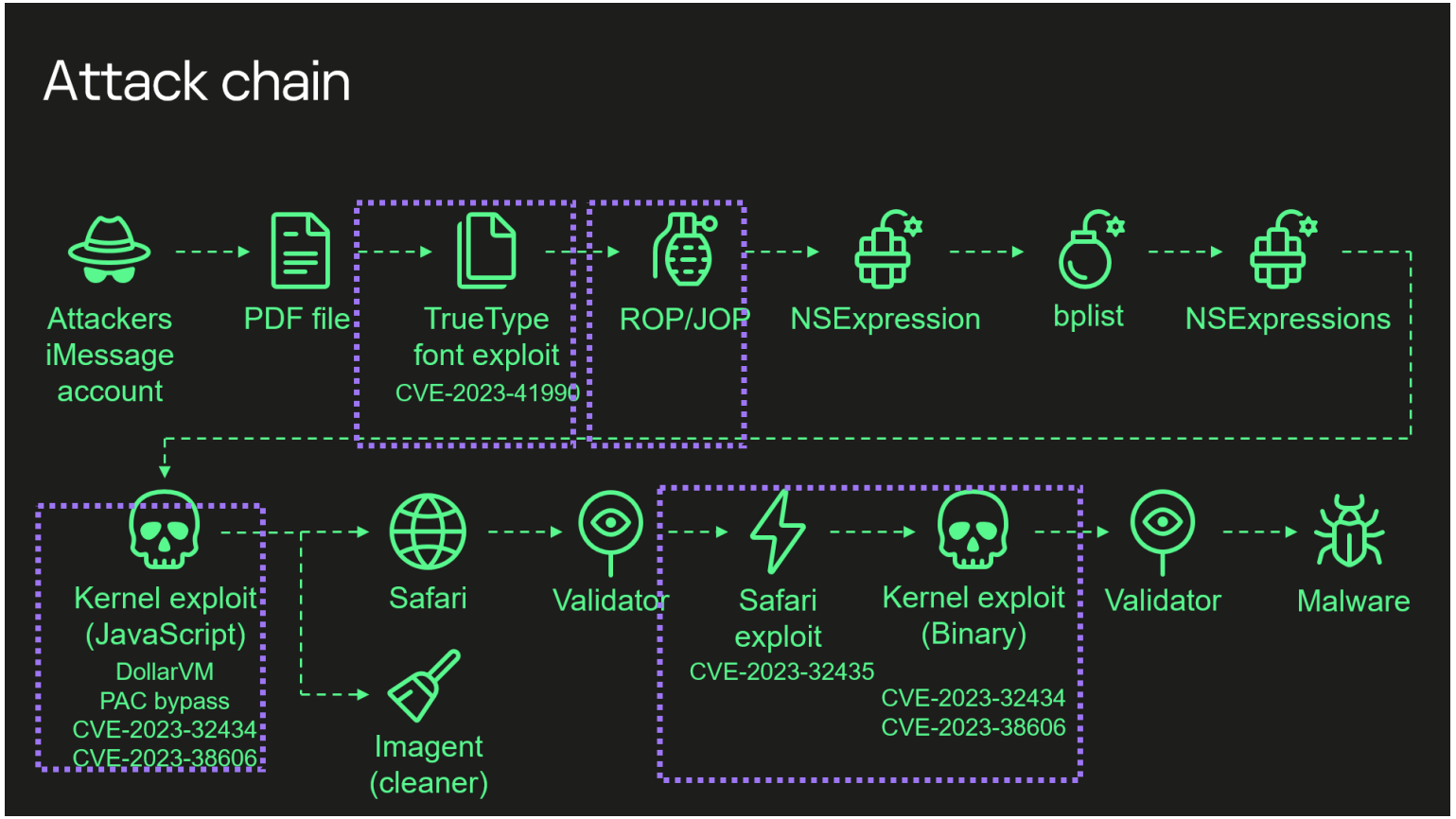
過去のサイバー攻撃:

- 1988: Morris Worm – sendmail, rsh/rexec
- 2003: SQL Slammer – MS SQL server
- **2014: Heartbleed - OpenSSL**
- 2018: Adobe Flash Player – Win, Mac, Linux and ChromeOS
- 2019: WhatsApp VOIP - smartphones

Source: Comparitech

# → 最近のPegasusスパイウェア

(Attack chain from Kaspersky's attack analysis https://securelist.com/operation-triangulation-the-last-hardware-mystery/111669/)




- 非常に複雑な攻撃チェーン
- CHERIなら1回目の攻撃を防げた (CVE-2023-41990)
- 次段のROP(Return Oriented Programming)/JOP(Jump Oriented Programming)も防げた
- CVE-2023-38606以外の他のCVEも防げた
  - 38696はKernelの特権昇格の脆弱性

出典: フリー百科事典『ウィキペディア（Wikipedia）』

**Pegasus**（ペガサス）とは、イスラエルの企業であるNSO Groupが開発したモバイル端末用のスパイウェアであり、同社の顧客が監視対象を秘密裏に監視するために使用している。人権侵害やプライバシー侵害などが指摘されている。このスパイウェアの存在は、Forbidden Stories、アムネスティ・インターナショナルのセキュリティラボを含む17の報道機関などによるThe Pegasus Projectの調査によって明らかになった[1]。

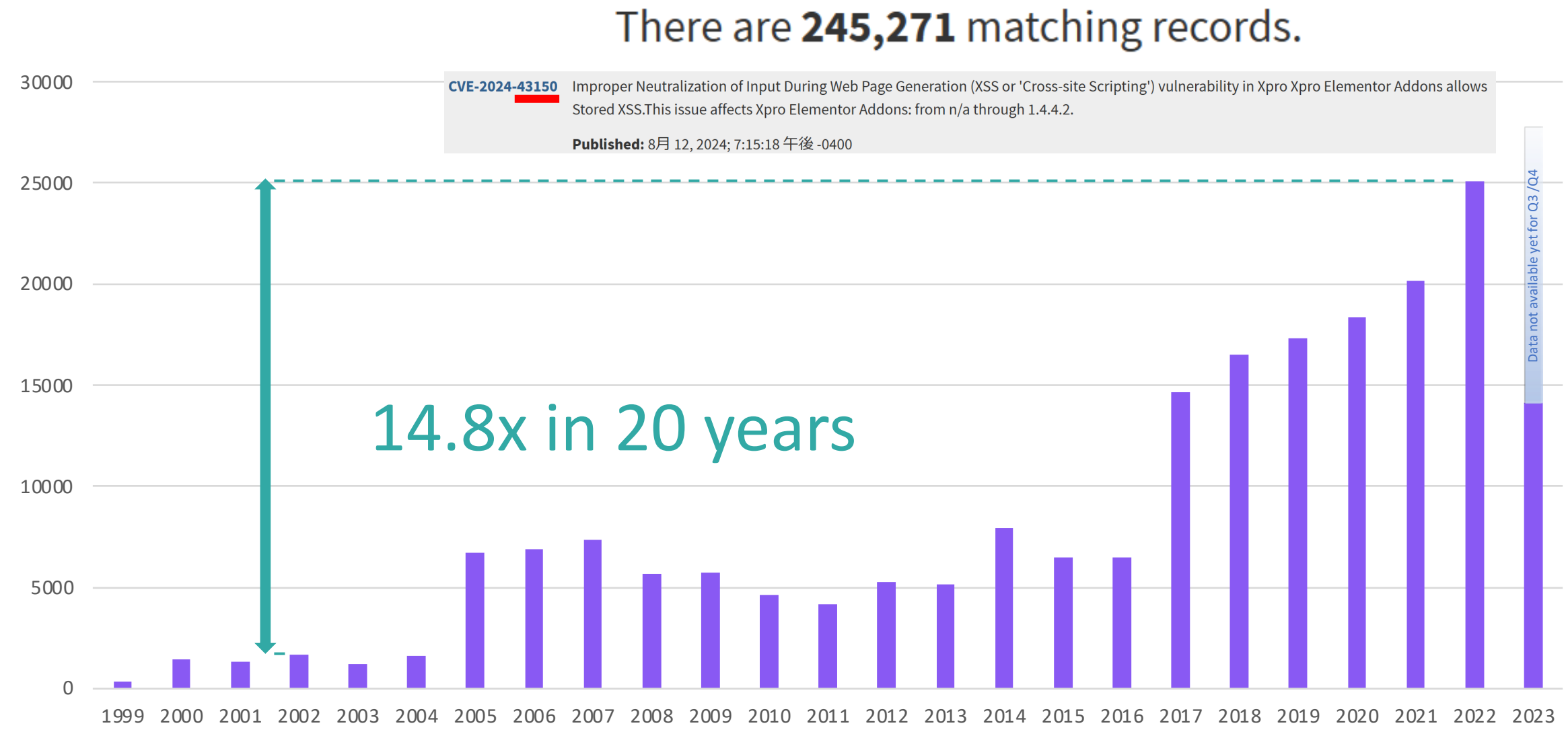
**感染後** [ 編集 ]

端末がPegasusに感染すると、攻撃者は理論上端末のあらゆるデータを収集できる。攻撃者はメッセージ、通話、写真、電子メールを抽出したり、秘密裏にカメラやマイクを有効化したり、WhatsApp、Telegram、Signalなどの暗号化メッセージアプリのメッセージを閲覧できる[6]。

Source: https://ja.wikipedia.org/wiki/Pegasus_(%E3%82%B9%E3%83%91%E3%82%A4%E3%82%A6%E3%82%A7%E3%82%A2)

# → 脆弱性は日々増加している

Codasip




■ Number of reported CVE – https://www.cve.org/About/Metrics

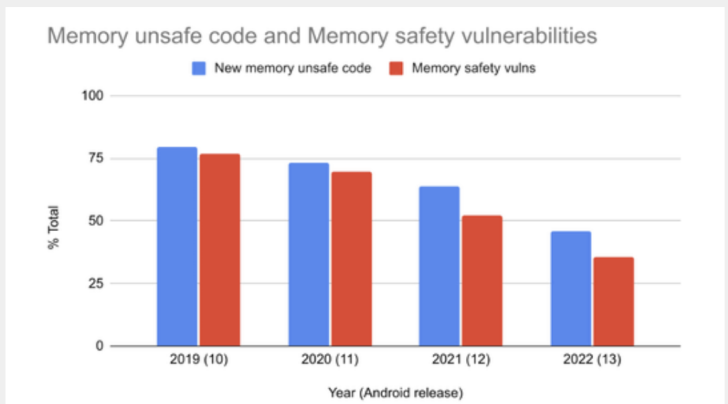
# → メモリセーフ vs メモリアンセーフ プログラミング言語

Source: 窓の杜
Google Security Blog

Shift to Memory-Safe Languages Gains Momentum

2022/12/07 DarkReading — 今週に、ソフトウェア・セキュリティの専門家たち発表したところによると、リモートからの悪用が可能で、野放し状態での攻撃の大部分に関与している、きわめて深刻な脆弱性のグループに対して、ソフトウェア業界は前進しているようだ。この脆弱性グループとは、いわゆるメモリ安全性の問題である。具体的には、バッファオーバーフロー／ユースアフターフリーなどを含むものであり、ソフトウェア会社が公表するアプリケーション・セキュリティ問題の大半を占めるものだ。今回の最新データでは、Java／C#／Rust などの、メモリ安全性の高い言語の使用が増加したことで、このクラスの脆弱性全体が急速に減少していることが示されている。

Java／Rust／Kotlin のメモリセーフ機能：C／C++ と安全性を比較してみる
Source: IoT Security News



メモリ安全性にかかわる脆弱性は漸減

そして重要なのは、現在までのところ、AndroidのRustコードで発見されたメモリ安全性の脆弱性はゼロだということだ。Rustコードにも非Rustコードと対話するために安全ではない部分があり、追加で精査しなければならないこともあるが、純粋なC/C ++実装よりもはるかに安全であるという。

ニュース

**Rustの採用が進んだ「Android 13」、メモリ破壊バグは減少、脆弱性の深刻度も低下傾向**

2022年はメモリ安全性の脆弱性がAndroidの脆弱性の大部分を占めない最初の年に

樽井 秀人　2022年12月9日 16:21

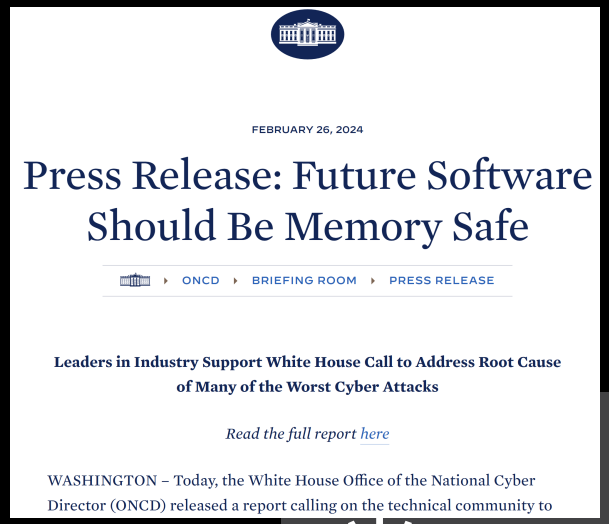
公式ブログ「Google Online Security Blog」

# → Android 5.0 – 14.0 CVE 累計 2024/01/16 時点

Codasip

| Year | Overflow | Memory Corruption | Sql Injection | XSS | Directory Traversal | File Inclusion | CSRF | XXE | SSRF | Open Redirect | Input Validation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2015 | 48 | 41 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 6 |
| 2016 | 175 | 84 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 96 |
| 2017 | 189 | 108 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 61 |
| 2018 | 41 | 141 | 5 | 0 | 6 | 0 | 0 | 2 | 0 | 0 | 40 |
| 2019 | 56 | 257 | 12 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 41 |
| 2020 | 169 | 338 | 18 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 89 |
| 2021 | 147 | 341 | 4 | 0 | 12 | 0 | 0 | 0 | 0 | 0 | 122 |
| 2022 | 196 | 560 | 7 | 0 | 23 | 0 | 0 | 0 | 0 | 0 | 180 |
| 2023 | 247 | 634 | 6 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | 130 |
| 2024 | 6 | 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Total | 1274 | 2523 | 52 | 0 | 62 | 0 | 0 | 2 | 0 | 0 | 765 |
| | 27% | 54% | 1% | 0% | 1% | 0% | 0% | 0% | 0% | 0% | 16% |

Source: https://www.cvedetails.com/version-list/1224/19997/1/Google-Android.html を集計



FEBRUARY 26, 2024

Press Release: Future Software Should Be Memory Safe

ONCD › BRIEFING ROOM › PRESS RELEASE

**Leaders in Industry Support White House Call to Address Root Cause of Many of the Worst Cyber Attacks**

*Read the full report here*

WASHINGTON – Today, the White House Office of the National Cyber Director (ONCD) released a report calling on the technical community to

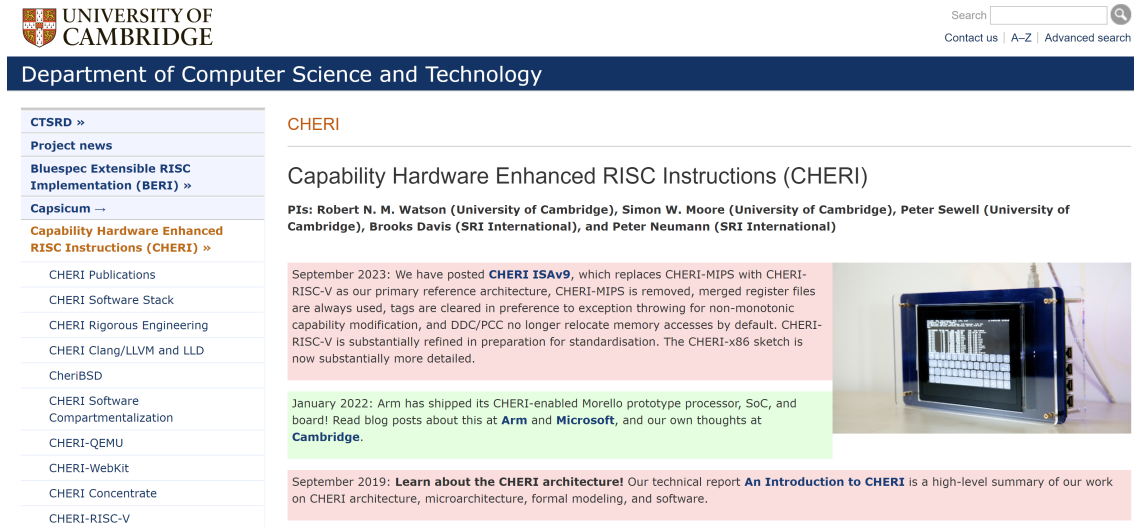
Codasip

米Whitehouse、米NSA、米DARPA、英政府も「メモリ**アンセーフ**プログラミング言語」に警鐘、対策へ

※ Appendixを参照

# → CHERI (Capability Hardware Enhanced RISC Instructions)

- 英DSIT下のDSbDや米DARPA等が約14年、￡300Mを投じて英ケンブリッジ大が中心となって開発した**セキュリティ強化型の命令セットアーキテクチャ**
  - https://www.cl.cam.ac.uk/research/security/ctsrd/cheri/
  - ハードウェア上で実装された能力(capability)に基づくメモリ保護を提供し、セキュリティの向上を目指す
  - **最も脆弱なC/C++ソースの救済**




DSIT: Department for Science, Innovation and Technology / 英国科学・イノベーション・技術省
DSbD: Digital Security by Design
DARPA: Defense Advanced Research Projects Agency

→ CHERIアライアンス 2024年6月17日 発足
https://cheri-alliance.net/about-cheri/

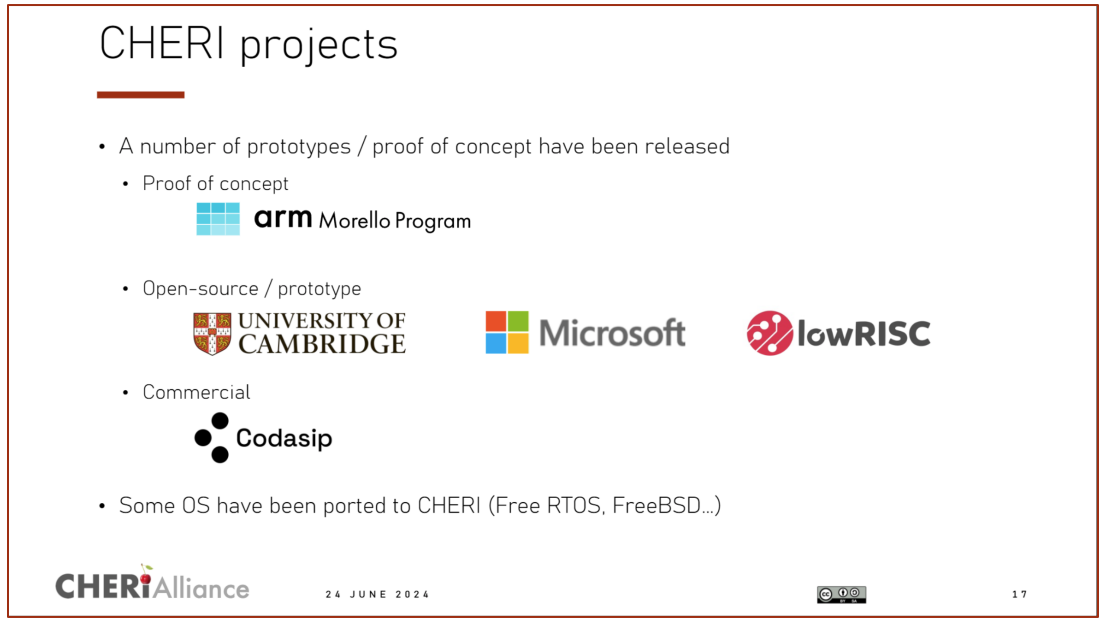
Codasip

CHERI Alliance

CHERI technology overview

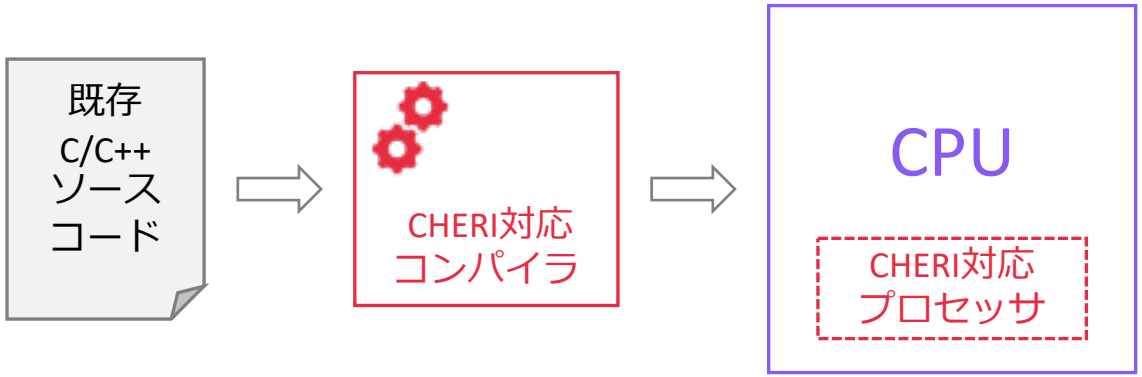
A QUICK INTRODUCTION

# Source: https://cheri-alliance.net/about-cheri/

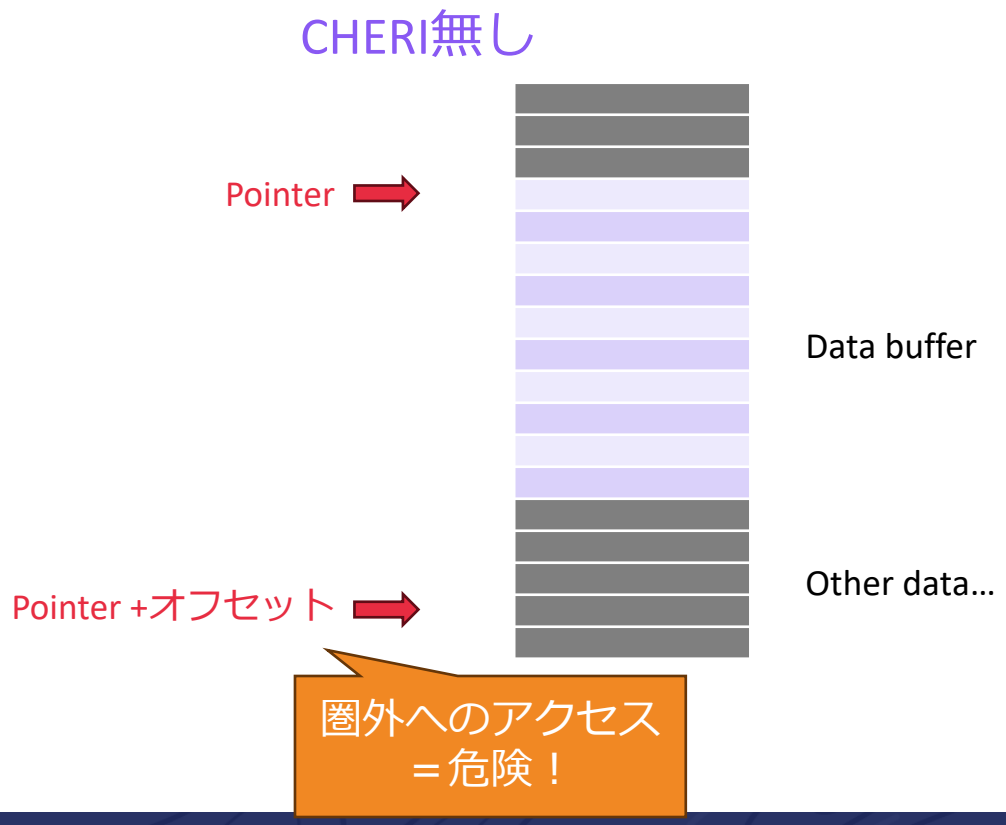
# → CHERIのコンセプト
## ハードウェアによる保護

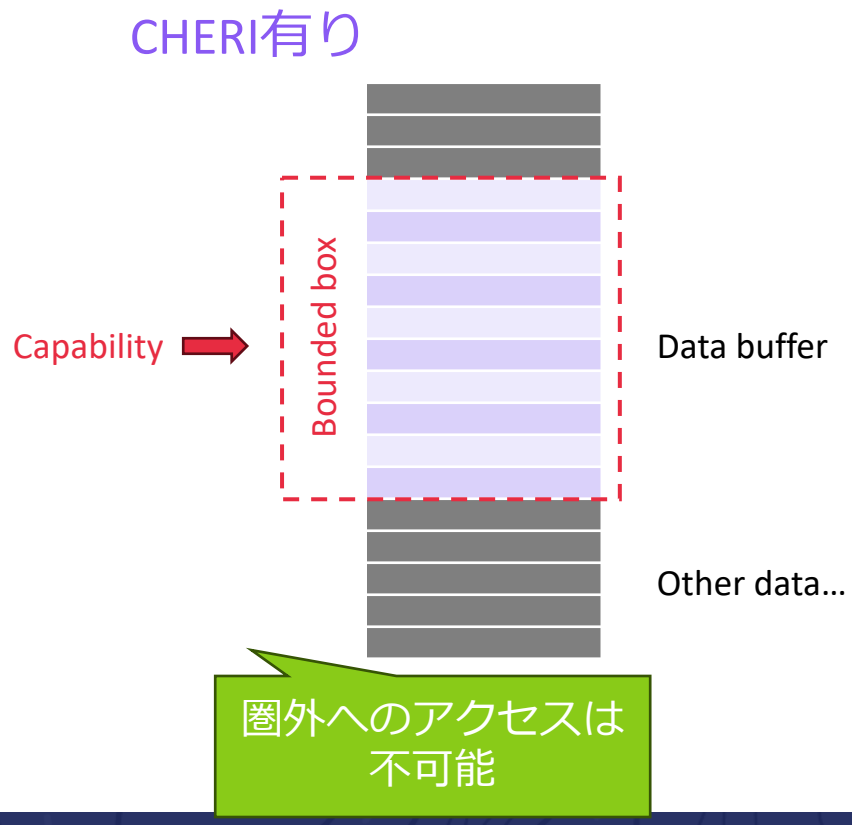
- CHERIはCHERI-ISAに適応したプロセッサを必要とする
  - 多様なコアを開発可能
  - ハードウェアによるセキュリティ
  - ソフトウェアによるバイパス不可能
  - 実証済み
- **既存コードの再利用**
  - **必要なコード変更はゼロか最小限**
  - **どの部分を保護するかを選択する**
- CHERIのメリット
  - 危険なコードは拒否
  - クリティカルなコード用にCHERIコンパートメントを作成する
  - Arm、~~MIPS~~、x86など、さまざまなアーキテクチャに対応



# → CHERIのコンセプト
## メモリ空間の安全性

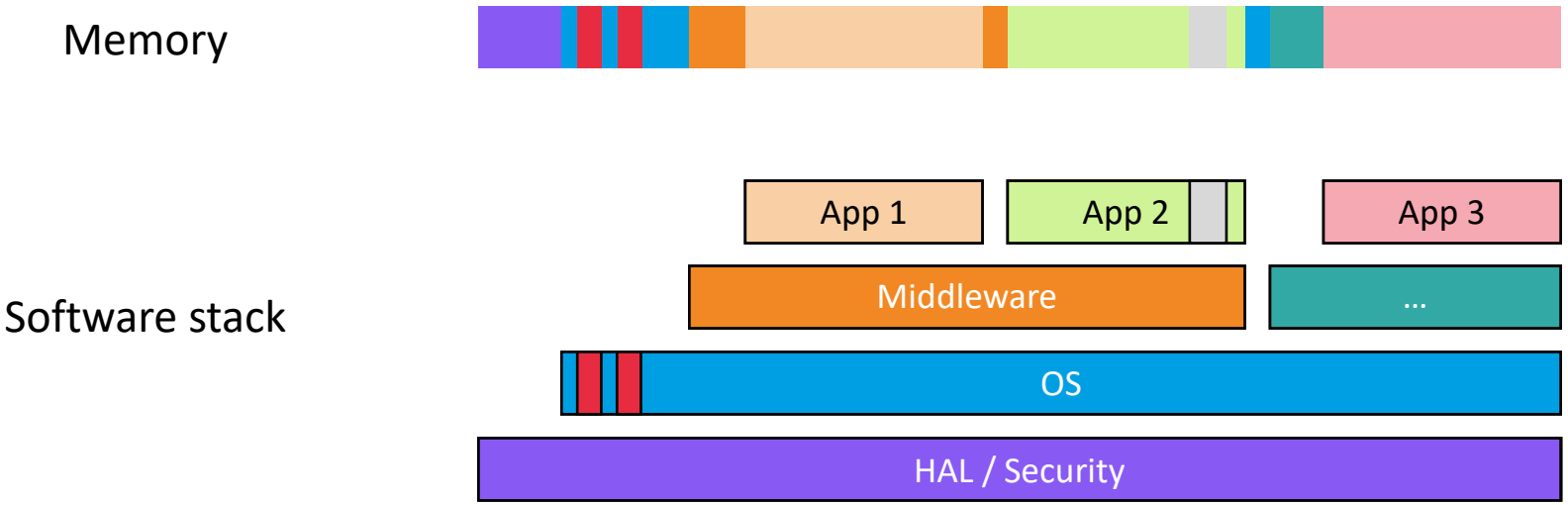
- PointerをCapabilityに置き換える - ハードウェア制御によって



# → CHERIのコンセプト
## コンパートメント化

- ケイパビリティは、特定された実行コンテキストに属する



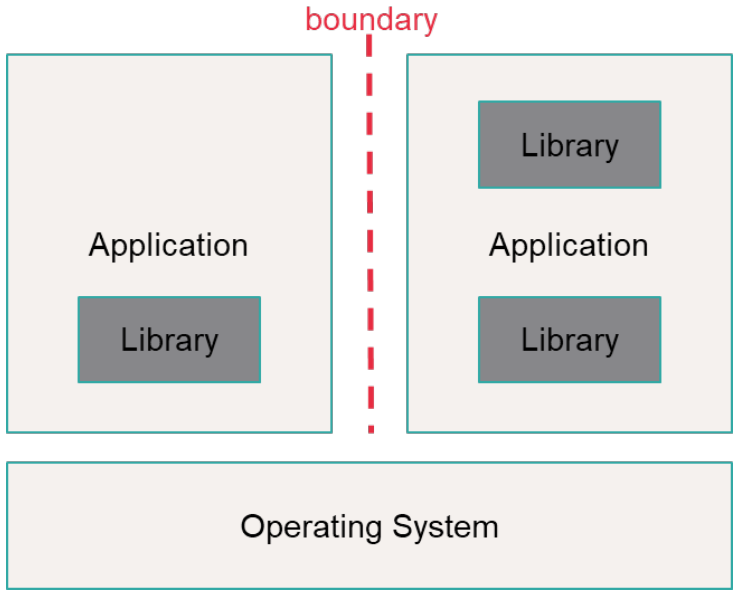

- Or driver
- Or library

Vertical and Horizontal fine-grained isolation

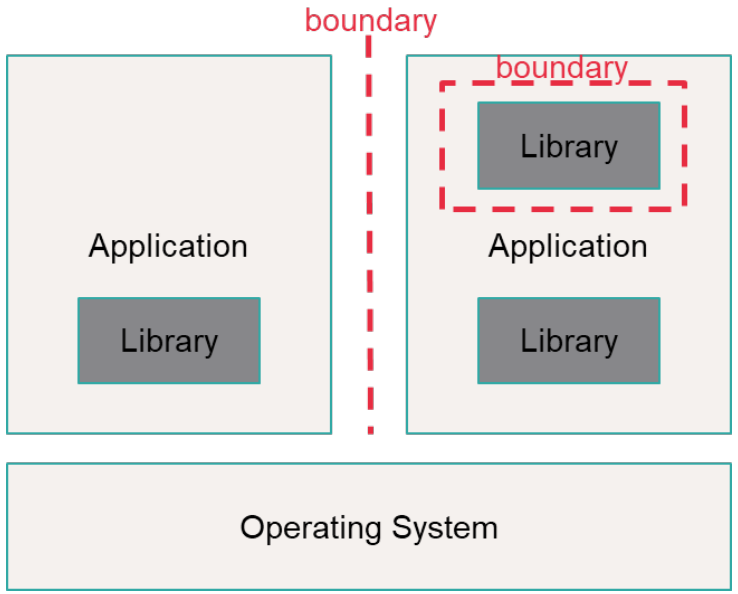
# → コンパートメント化

Codasip

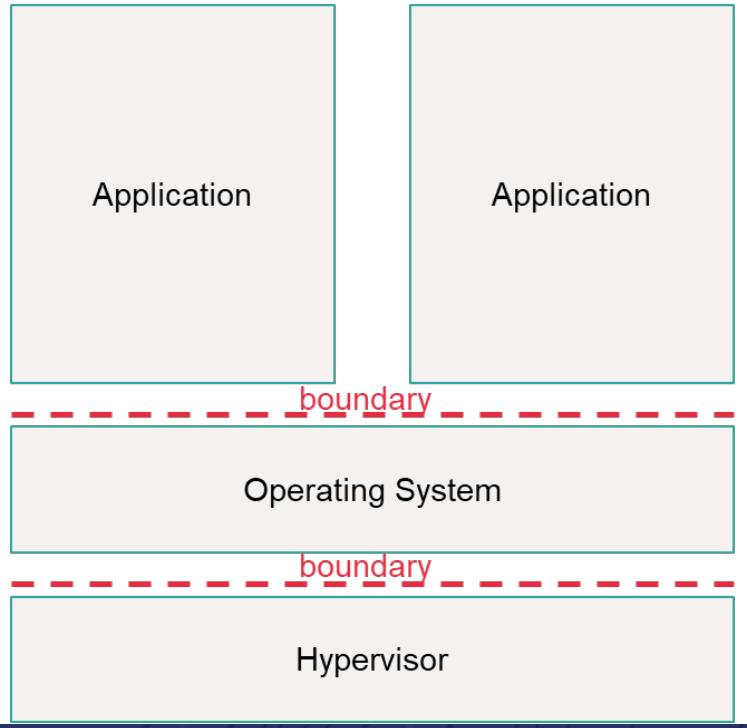
垂直




垂直 + 特定部分




水平

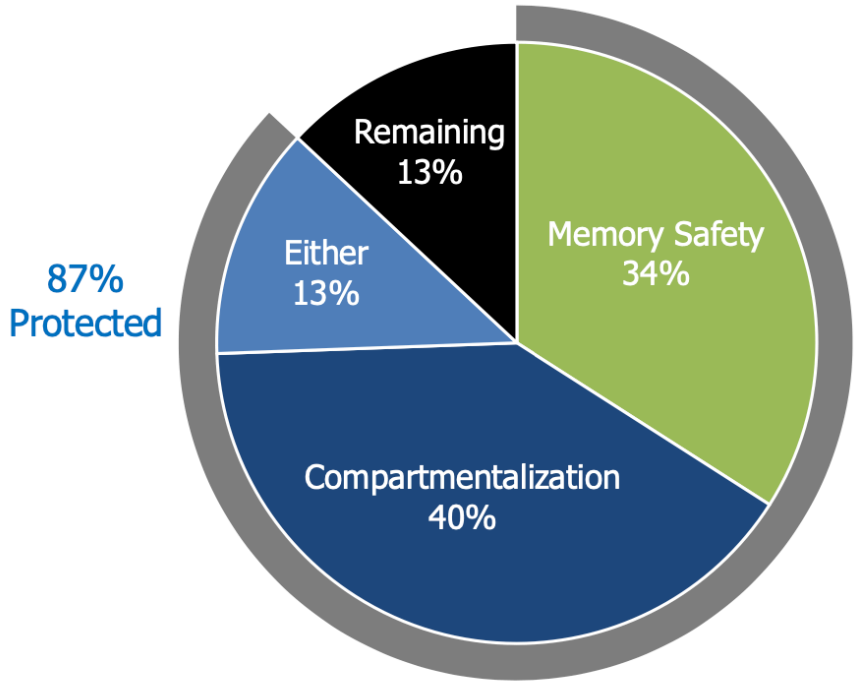


# → Linuxカーネルの深刻なCVEの87%に有効

Codasip

## Memory safety and compartmentalization would stop most campaigns

Mitigation analysis of Linux kernel high severity CVEs




- Either: if memory safety or compartmentalization were present, the bug would not be exploitable
- Remaining: memory safety or compartmentalization would not mitigate the CVE

CVE: Common Vulnerabilities and Exposures

McKee, Derrick, et al. "Preventing kernel hacks with HAKC." Proceedings 2022 Network and Distributed System Security Symposium. NDSS. Vol. 22. 2022.





accellera SYSTEMS INITIATIVE

2024 DVCON JAPAN

Codasip

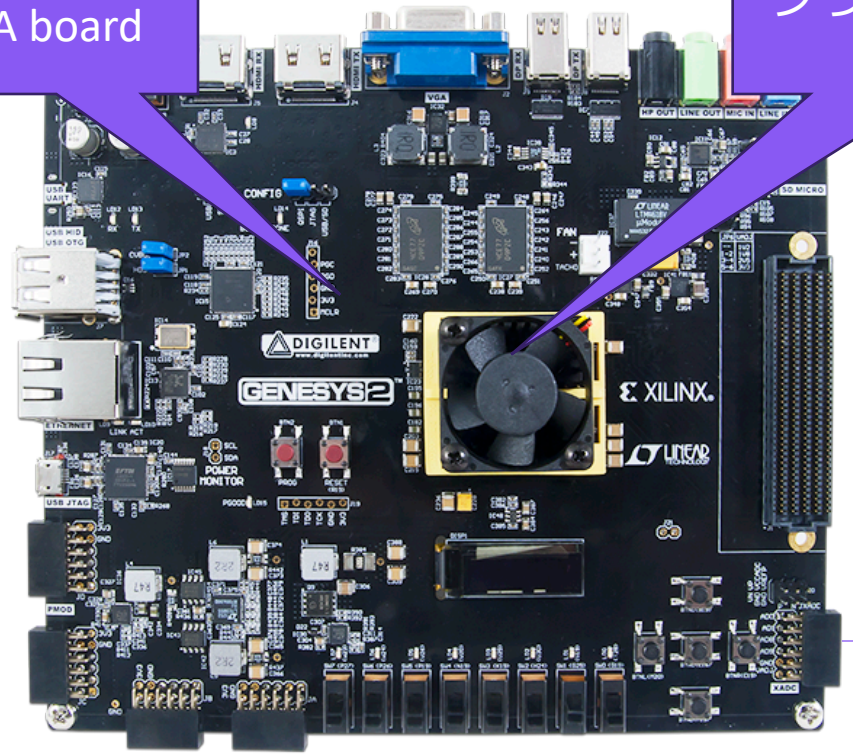
↘

# CHERIデモ

# → Codasip CHERIデモ: 環境

Digilent
Genesys 2
FPGA board

Codasip
CHERI対応IPコアと
プラットフォーム

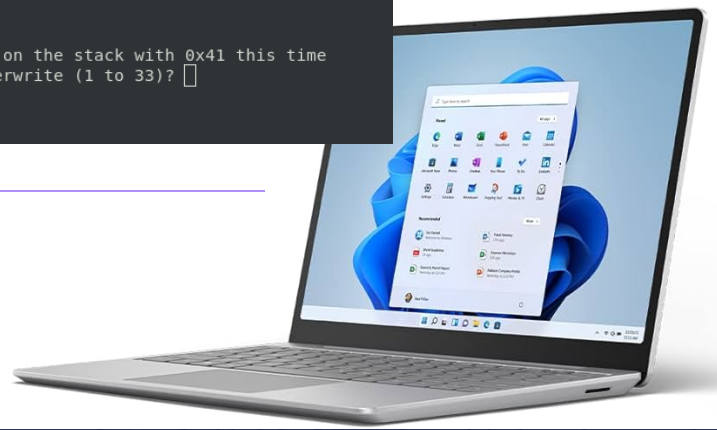



```
Now let's try a buffer overflow with CHERI ENABLED...
We will reset the stack arrays back to their previous contents
(but still keep the corrupted canary value).

Stack contents:
20017e80:b0 f2 00 20 00 00 00 00 84 b1 45 05 00 00 7d a5
20017e90:a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 50
20017ea0:41 53 53 57 4f 52 44 00 21 21 21 21 21 21 21 21
20017eb0:48 ea 00 20 00 00 00 00 4c aa 95 06 00 00 7d ff
20017ec0:3f 7f 01 20 00 00 00 00 3b 7f d2 07 00 20 fd ff
20017ed0:01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20017ee0:03 84 00 20 00 00 00 00 07 04 05 05 00 00 15 ff
20017ef0:54 0b 00 20 00 00 00 00 00 00 00 00 08 00 20 ff ff
20017f00:4f 7f 01 20 00 00 00 00 4b bf d7 07 00 20 fd ff

Secret password on the stack: PASSWORD
Canary value: 0x2121212121212121


We will overwrite the byte array on the stack with 0x41 this time
How many bytes do you want to overwrite (1 to 33)?
```



# → CHERIデモのゴール

Codasip

- デモの着目ポイント:
  - 攻撃者がスタックを破損し、重要なデータを変更
  - スタック破壊保護 (スタック・カナリア) が無力であることを確認
  - **CHERIの優れたアクティブなランタイム・メモリ保護を確認**

# → CHERIデモ: ソフトウエア

Codasip

- 右に示すCコードを使用して、demo()関数内でスタック上に2つのローカル配列を作成する

- 注：このデモは、一般的なソフトウェア・セキュリティの慣行に沿った「強力なスタック保護」オプションでコンパイルされている

```
int demo(void)
{
   printf("CHERI Demonstration¥n¥n");

   // Add secret password to stack
   char password[9];

   // Create byte array on stack
   char array[16];
   ...
```

# → CHERIデモ: (デモ画面1)

Codasip

```
CHERI Demonstration

This demonstration adds two things onto the stack:
  1. A character array containing a secret password ("PASSWORD")
  2. A byte array of 16 bytes before it (filled with 0xa5)


We assume the attacker can overflow the byte array.

NOTE: the stack protector software protection IS ENABLED
      The code is compiled with -fstack-protector-strong
      You can see the "canary" value 0xadbaadbaadbaadba

Stack contents:
20017e80:b0 f2 00 20 00 00 00 00 84 b1 45 05 00 00 7d a5
20017e90:a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 50
20017ea0:41 53 53 57 4f 52 44 00 ba ad ba ad ba ad ba ad
20017eb0:48 ea 00 20 00 00 00 00 4c aa 95 06 00 00 7d ff
20017ec0:3f 7f 01 20 00 00 00 00 3b 7f d2 07 00 20 fd ff
20017ed0:01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20017ee0:03 84 00 20 00 00 00 00 07 04 05 05 00 00 15 ff
20017ef0:54 0b 00 20 00 00 00 00 00 00 00 08 00 20 ff ff
20017f00:4f 7f 01 20 00 00 00 00 4b bf d7 07 00 20 fd ff

Secret password on the stack: PASSWORD
Canary value: 0xadbaadbaadbaadba

Now we will do a buffer overflow attack with CHERI DISABLED.
We will overwrite the byte array on the stack with 0x21 bytes
This is the ASCII character '!'
How many bytes do you want to overwrite (1 to 33)?
```

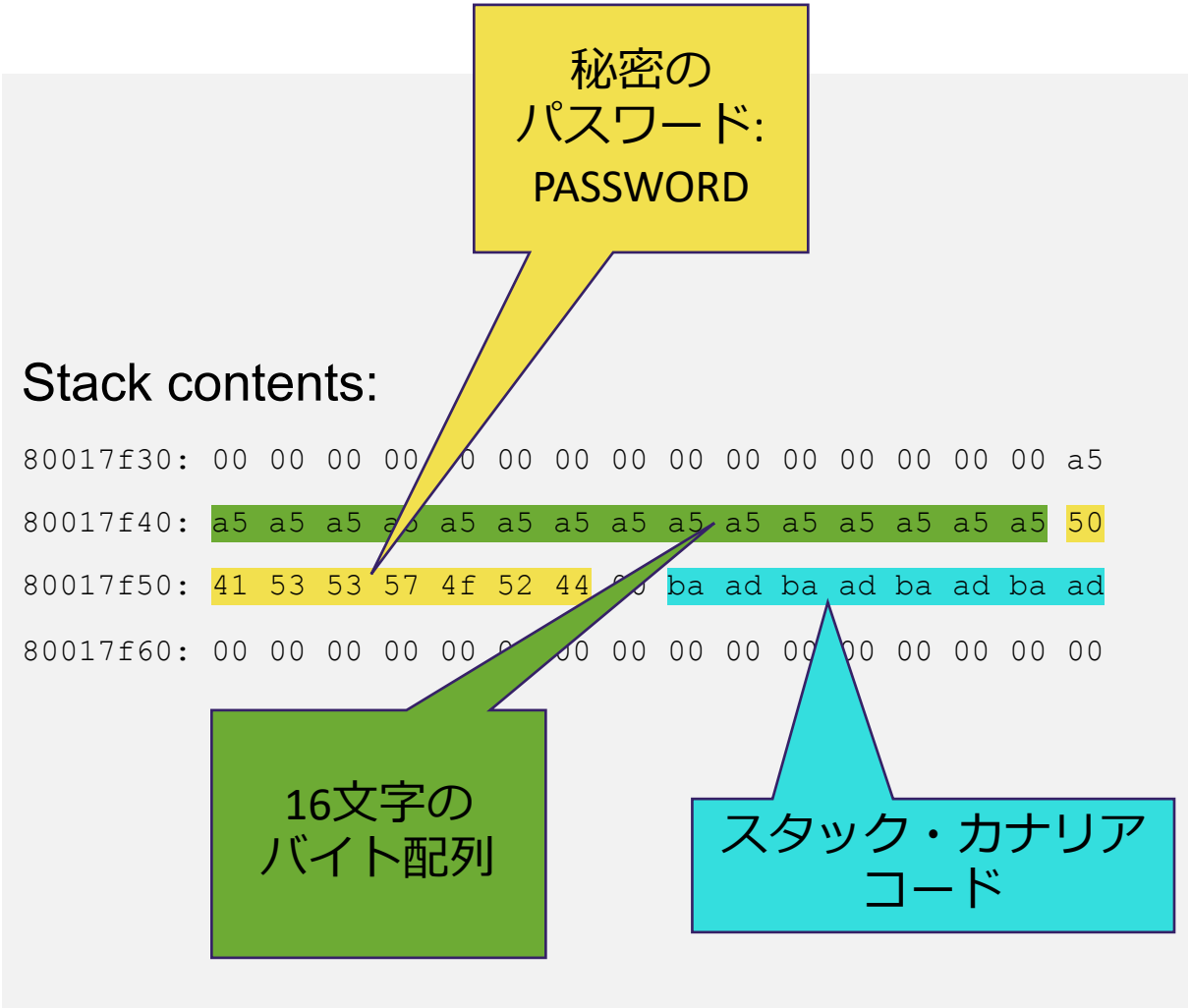
16文字の
バイト配列

秘密の
パスワード：
**PASSWORD**

スタック・カナリア
コード

# → CHERIデモ: ソフトウエア – 起動時

- コンパイラはスタック上に2つのメモリ領域を確保する
  - 9文字のパスワード
  - 16文字のバイト配列
- 16進数の50 41 53 53 57 4f 52 44は、文字のASCII値である
  - ‘P', ‘A’, ‘S’, ‘S’, ‘W’, ‘O’, ‘R’, ‘D’
- “スタック・スマッシング保護”を有効にしているため、コンパイラはパスワードの後に64ビットのカナリア値を自動挿入
  - 0xadbaadbaadbaadba



# → CHERIデモ: CHERIをDISABLE

- バッファオーバーフロー攻撃のように、バイト配列の末尾を越えて書込む
  - 16バイト以下の書き込みは、バイト配列を上書きするだけである
  - これを超えて書き込むと、パスワードとカナリア値を破損する可能性がある

Now we will do a buffer overflow attack with CHERI DISABLED.

We will overwrite the byte array on the stack with 0x21 bytes This is the ASCII character '!'

How many bytes do you want to overwrite (1 to 33)?

---

CHERIを無効にしてバッファオーバーフロー攻撃を行う

スタック上のバイト配列を0x21バイトで上書き
0x21はASCII文字の「!」

何バイト上書きしますか (1～33) ？

ここでは33バイト書き込み

# → CHERIデモ: CHERIをDISABLE (デモ画面2)

Codasip

```
How many bytes do you want to overwrite (1 to 33)? 33

Stack contents:
20017e80:b0 f2 00 20 00 00 00 00 84 b1 45 05 00 00 7d 21
20017e90:21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
20017ea0:21 21 21 21 21 21 21 21 21 21 21 21 21 21 21 21
20017eb0:48 ea 00 20 00 00 00 00 4c aa 95 06 00 00 7d ff
20017ec0:3f 7f 01 20 00 00 00 00 3b 7f d2 07 00 20 fd ff
20017ed0:01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20017ee0:03 84 00 20 00 00 00 00 07 04 05 05 00 00 15 ff
20017ef0:54 0b 00 20 00 00 00 00 00 00 00 08 00 20 ff ff
20017f00:4f 7f 01 20 00 00 00 00 4b bf d7 07 00 20 fd ff

Secret password on the stack: !!!!!!!!
** Attacker now controls password **

Canary value: 0x2121212121212121



Note that stack protector didn't detect the attack.
Even though the stack protector canary is corrupted.
It only checks the canary when we return from a function.
Canary's current value is  : 0x2121212121212121.
Canary's expected value is : 0xadbaadbaadbaadba.
An attacker can use the corrupt values up to that point
and still do damage to the system.

Press a key to continue...
```

33バイトの上書き

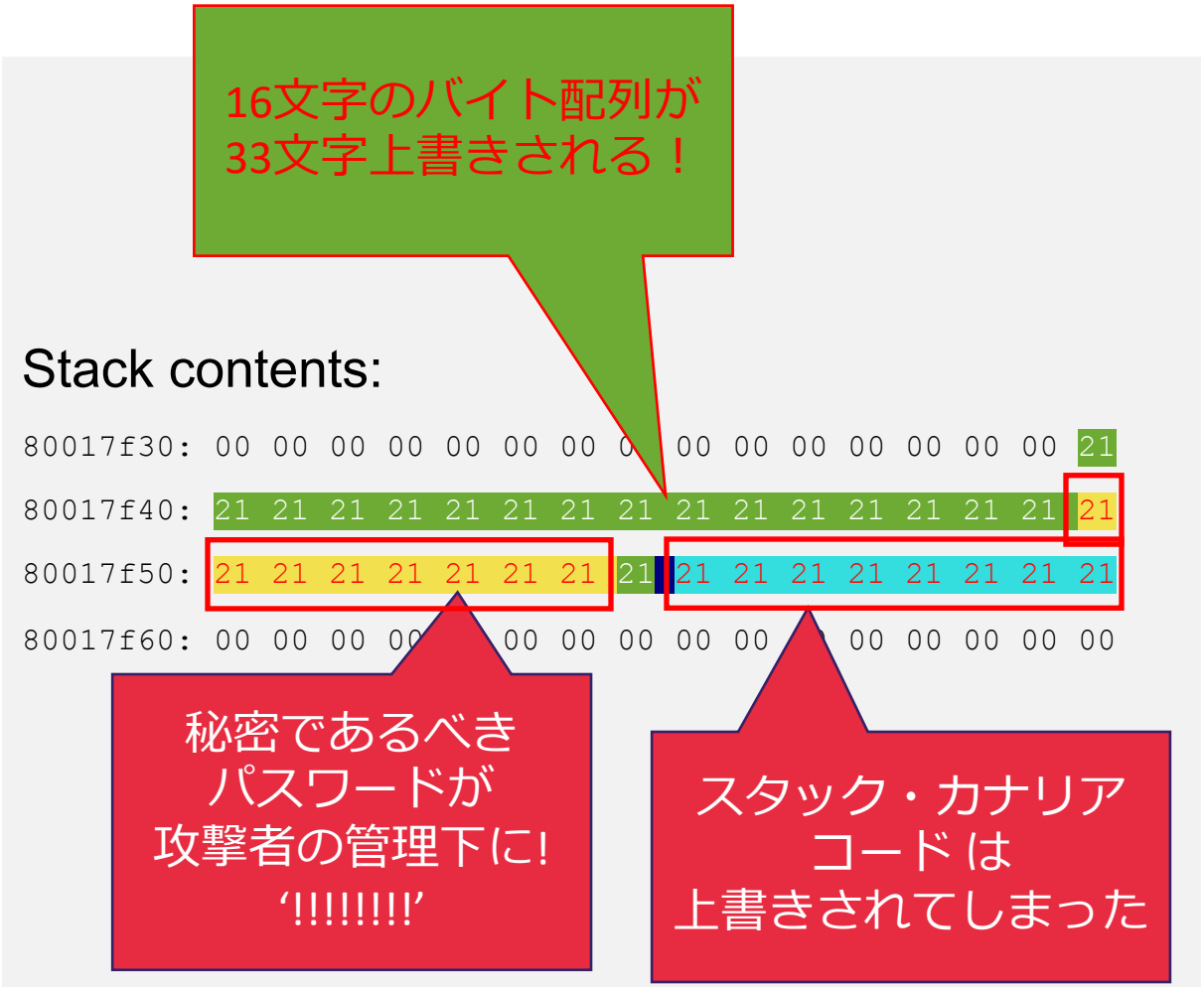
パスワードは攻撃者の手に渡った

スタック・カナリアコードは上書きされた

# → CHERIデモ: バッファーオーバーフローした状態

Codasip

- 0x21を33バイト入力した結果
- スタック・カナリアがカナリア値が破損しているにもかかわらず、攻撃を検知していないことに着目
  - カナリアをチェックするのは、関数からリターンしたときだけ
    - 期待値: 0xadbaadbaadbaadba
    - 現在値: 0x2121212121212121
  - 攻撃者はその時点までの破損した値を使用して、システムを攻撃することができる



# → CHERIデモ: CHERIをENABLE

- CHERIを**有効**にして
  再度バッファオーバーフローを試す
- スタック配列をリセットして、
  以前の内容に戻す
  - しかし、破損したカナリアはそのまま
- 上書きするバイトの値は0x41

We will overwrite the byte array on the stack with 0x41 this time

How many bytes do you want to overwrite (1 to 33)?

---

スタック上のバイト配列を0x41で上書きする

何バイト上書きしますか (1～33) ？

ここでは**20**バイト書き込み

# → CHERIデモ: CHERIをENABLE (デモ画面3)

Codasip

```
Now let's try a buffer overflow with CHERI ENABLED...
We will reset the stack arrays back to their previous contents
(but still keep the corrupted canary value).

Stack contents:
20017e80:b0 f2 00 20 00 00 00 00 84 b1 45 05 00 00 7d a5
20017e90:a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 a5 50
20017ea0:41 53 53 57 4f 52 44 00 21 21 21 21 21 21 21 21
20017eb0:48 ea 00 20 00 00 00 00 4c aa 95 06 [illegible] 00 7d ff
[illegible]17ec0:3f 7f 01 20 00 00 00 00 3b 7f d2 07 00 20 [illegible]
20017ed0:01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20017ee0:03 84 00 20 00 00 00 00 07 04 05 05 00 00 15 ff
20017ef0:54 0b 00 20 00 00 00 00 00 00 00 08 00 20 ff ff
20017f00:4f 7f 01 20 00 00 00 00 4b bf d7 07 00 20 fd ff

Secret password on the stack: PASSWORD
Canary value: 0x2121212121212121


We will overwrite the byte array on the stack with 0x41 this time
How many bytes do you want to overwrite (1 to 33)?
```

秘密の
パスワード：
PASSWORD

16文字の
バイト配列

スタック・カナリア
コードは
上書きされたまま

# → CHERIデモ: CHERIをENABLE (デモ画面4)

Codasip

```
Try again (y/n)? y
How many bytes do you want to overwrite (1 to 33)? 20

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++


++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++


Stack contents:
20017e80:b0 f2 00 20 00 00 00 00 84 b1 45 05 00 00 7d 41
20017e90:41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 50
20017ea0:41 53 53 57 4f 52 44 00 21 21 21 21 21 21 21 21
20017eb0:48 ea 00 20 00 00 00 00 4c aa 95 06 00 00 7d ff
20017ec0:3f 7f 01 20 00 00 00 00 3b 7f d2 07 00 20 fd ff
20017ed0:01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
20017ee0:03 84 00 20 00 00 00 00 07 04 05 05 00 00 15 ff
20017ef0:54 0b 00 20 00 00 00 00 00 00 00 08 00 20 ff ff
20017f00:4f 7f 01 20 00 00 00 00 4b bf d7 07 00 20 fd ff

Secret password on the stack: PASSWORD
Canary value: 0x2121212121212121

You will have just seen that CHERI stops any writes beyond the array.


Press a key to continue...
```

CHERI が
バッファオーバーフロー
を4回検出

これにより、攻撃者は
パスワードを知ることが
できない

スタック・カナリア
コードは
上書きされたまま

# → CHERIデモ: CHERIが配列境界を守る

Codasip

- 0x41を20バイト入力した結果

Program output:

```
++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++
```

16文字のバイト配列
16文字のみ上書きされる！

Stack contents:

```
80017f30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
80017f40: 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 50
80017f50: 41 53 53 57 4f 52 44 21 21 21 21 21 21 21 21 21
80017f60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

秘密のパスワード
異常無し

上書きされたままの
スタック・カナリア
コード

# → CHERIデモ: CHERIがメモリセーフを実現

Codasip

- CHERIが配列境界を越えての書き込みを阻止
  - バイト配列を超えてアクセスしようとする毎に例外が発生するため、4つのCHERI例外が発生

- スタックトレースで下記を確認
  - データは書き込まれなかった
  - パスワードはPASSWORDのまま保護された

Program output:

```
++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++

++++++++++++++++++++++++++++++++++++++++
++ EXCEPTION HANDLER: CHERI EXCEPTION ++
++++++++++++++++++++++++++++++++++++++++
```

# → CHERIデモ: スタック・カナリアが遅れて動作 (デモ画面5)

```
CHERI provides a much more robust runtime protection

Exiting the function now...
It is only at this point that -fstack-protector will work

**************************************************
** Stack-protector: Stack corruption detected **
**************************************************
```

その後
スタック・カナリアコードが
メモリ破壊を検出

# → このデモでCHERIは何を達成できたか？

Codasip

- **CHERI無しの場合:**
  - ハッカーはパスワードを発見し、変更することができる
  - システムは攻撃に気づかず、動作を継続する
  - 製品が危険にさらされる可能性がある
  - スタック・カナリアはバイパスされ、効果がない
- **CHERI有りの場合:**
  - 攻撃者がパスワードを盗んだり変更することを防ぐ
  - 攻撃を検知し、対応策を講じることができる
  - 製品は無傷のまま
  - スタック・カナリアのような統計的メカニズムは必要ない
- **CHERIは、すべてのメモリアクセスを100%チェックする**

Codasip

CHERIは、業界の高度なメモリ保護の新たなセキュリティコンセプトである

Codasipは、RISC-V IPポートフォリオ全体をCHERI対応します

Codasip

# 「プログラマ依存のメモリ安全」からの脱却

Codasip

→

# Thank you!

Codasip

↘

# Appendix:
## 世界の動向
## ご参考までに

# → 2023-12-6: NSA (National Security Agency) recommend Memory safety


**PRESS RELEASE** | Dec. 6, 2023

## U.S. and International Partners Issue Recommendations to Secure Software Products Through Memory Safety

FORT MEADE, Md. - The National Security Agency (NSA) joins Cybersecurity and Infrastructure Security Agency (CISA) and U.S. and international partners in releasing "The Case for Memory Safe Roadmaps" Cybersecurity Information Sheet (CSI). Expanding on the "Software Memory Safety" CSI published by NSA in April 2023, the report provides guidance for software manufacturers and technology providers to create roadmaps tailored to eliminate memory safety vulnerabilities from their products.

- Source: https://www.nsa.gov/Press-Room/Press-Releases-Statements/Press-Release-View/Article/3608324/us-and-international-partners-issue-recommendations-to-secure-software-products/

# → 2024-2-26: Whitehouse

https://www.whitehouse.gov/oncd/briefing-room/2024/02/26/memory-safety-statements-of-support/
https://www.whitehouse.gov/oncd/briefing-room/2024/02/26/press-release-technical-report/
https://www.whitehouse.gov/oncd/briefing-room/2024/02/26/memory-safety-fact-sheet/

THE WHITE HOUSE

FEBRUARY 26, 2024

## Statements of Support for Software Measurability and Memory Safety

ONCD › BRIEFING ROOM › PRESS RELEASE

*Read the full report here*

*Read the fact sheet here*

Today, the Office of the National Cyber Director released a new Technical Report titled *"Back to the Building Blocks: A Path Toward Secure and Measurable Software."* This report builds upon the President's National Cybersecurity Strategy, addressing the technical community to tackle undiscovered vulnerabilities that malicious actors can exploit.

Leading technology companies, academics, and civil society organizations applauded the Biden-Harris Administration's efforts and underscored the importance of software measurability and memory safety:

**Industry**

THE WHITE HOUSE

FEBRUARY 26, 2024

## Press Release: Future Software Should Be Memory Safe

ONCD › BRIEFING ROOM › PRESS RELEASE

**Leaders in Industry Support White House Call to Address Root Cause of Many of the Worst Cyber Attacks**

*Read the full report here*

WASHINGTON – Today, the White House Office of the National Cyber Director (ONCD) released a report calling on the technical community to proactively reduce the attack surface in cyberspace. ONCD makes the case that technology manufacturers can prevent entire classes of vulnerabilities from entering the digital ecosystem by adopting memory safe programming languages. ONCD is also encouraging the research community to address the problem of software measurability to enable the development of better diagnostics that measure cybersecurity quality.

The report is titled *"Back to the Building Blocks: A Path Toward Secure and Measurable Software."*

THE WHITE HOUSE

FEBRUARY 26, 2024

## Fact Sheet: ONCD Report Calls for Adoption of Memory Safe Programming Languages and Addressing the Hard Research Problem of Software Measurability

ONCD › BRIEFING ROOM › PRESS RELEASE

*ONCD Rallies Industry, Academia, and Civil Society to Join Effort*

February 26, 2024

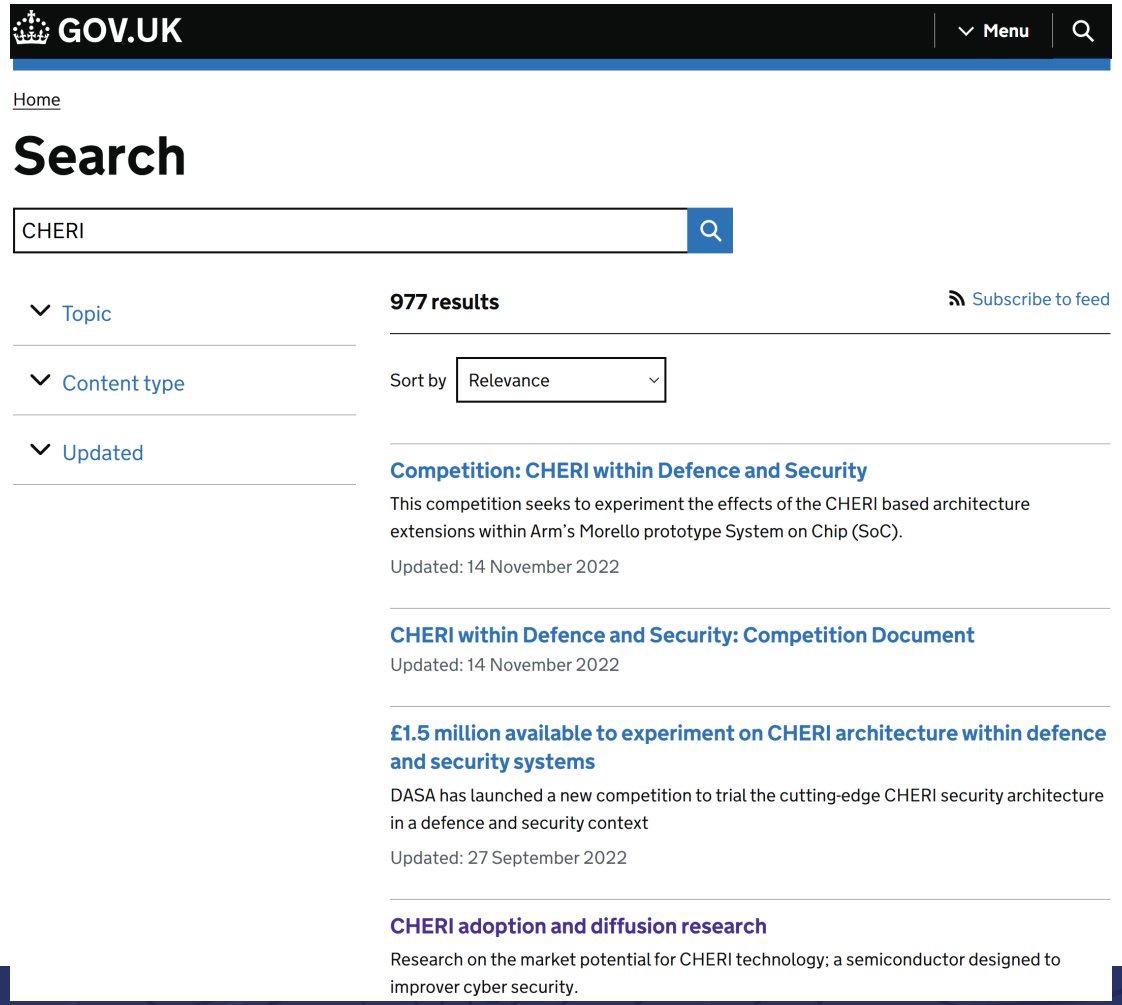
*Read the full report here*

*Watch the video address here*

Today, the Office of the National Cyber Director (ONCD) published a technical report entitled *"Back to the Building Blocks: A Path Toward Secure and Measurable Software."* The report builds upon the President's National Cybersecurity Strategy in describing the urgent need to address undiscovered vulnerabilities that malicious actors can exploit. The report outlines two strategic approaches to achieve this goal:

# → 英政府の状況

https://www.gov.uk/search/all?keywords=CHERI



Codasip

Blog

# An open letter regarding Cyber Resilience of the UK's Critical National Infrastructure

**8 May, 2024**

*by Ron Black*

**Codasip announced a commercially available RISC-V processor with CHERI for license in October of 2023 and is demonstrating technology for IP provenance.**

Dear Members of the Science, Innovation and Technology Committee,

Let me start by applauding your hearing on 24 April 2024, and in particular the evidence of Professor John Goodacre, Challenge Director of Digital Security by Design at Innovate UK, and Mr Richard Grisenthwaite, Executive Vice President and Chief Architect at Arm. During this hearing, the witnesses discussed two extremely important cybersecurity issues: **memory safety** and **IP provenance**. In this letter, I would like to