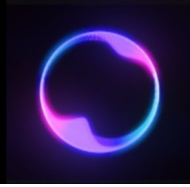
9. SGX攻撃編②

Ao Sakurai

2023年度セキュリティキャンプ全国大会
L5 - TEEの活用と攻撃実践ゼミ

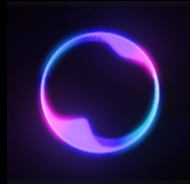
本セクションの目標



- Controlled-Channel攻撃とPlundervolt攻撃の解説を行う
- mprotectシステムコールを用いる事により、ごく簡単かつ擬似的なControlled-Channel攻撃を実践する
- SGX-Vaultの実装の一部に対し、Controlled-Channel攻撃に対する軽減策の導入を行う

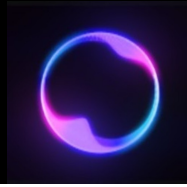
Controlled-Channel攻撃

Controlled-Channel攻撃

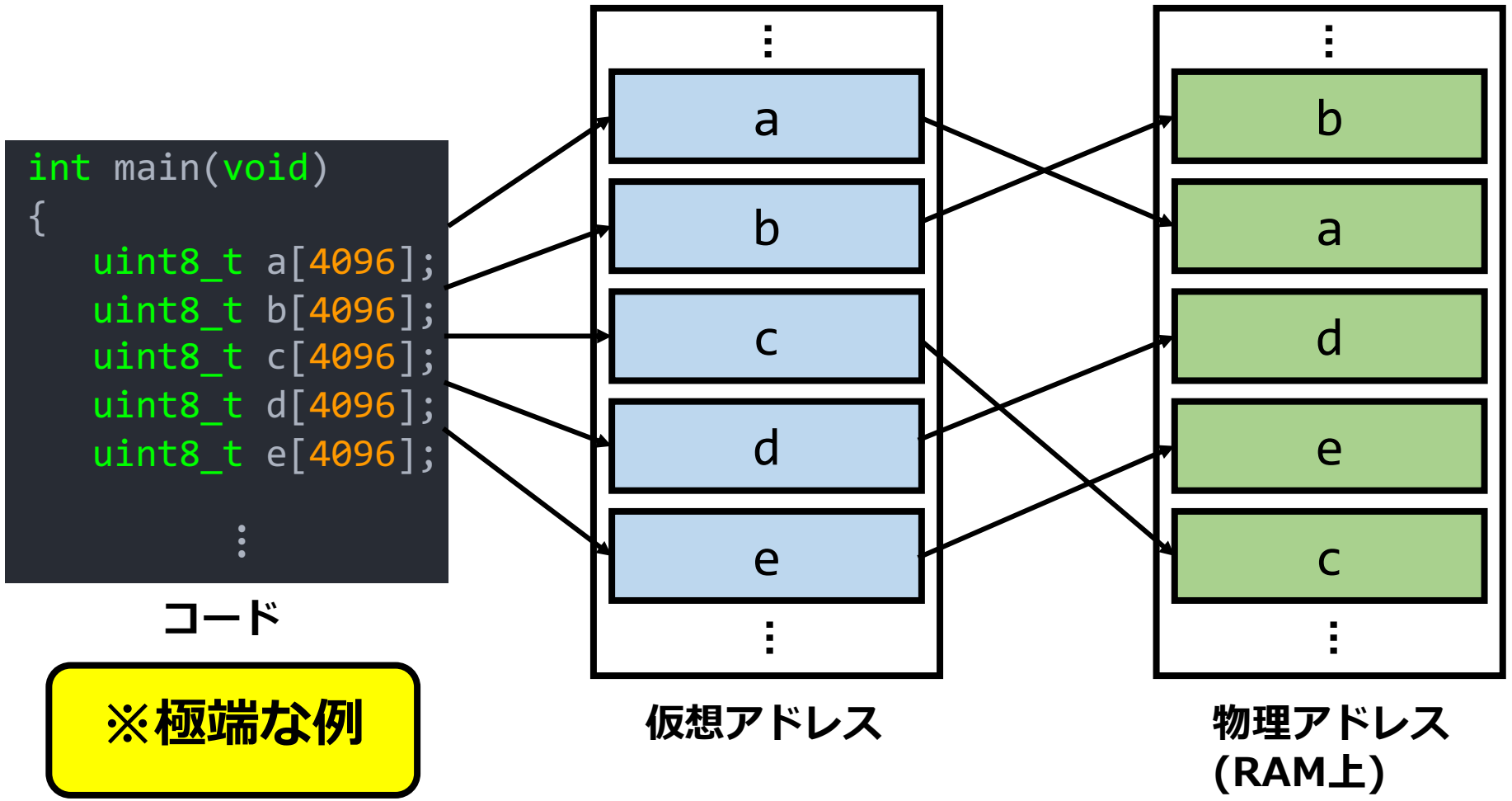


- **SGX**のような**防護システム**に対して**非常に有効**な**サイドチャネル攻撃**の一つ
- この攻撃では**ページフォルト**を悪用する
- 以下、OSは**攻撃者**により**制御権を掌握されている**と仮定する
- この攻撃では**3つ**のフェーズが存在する：
  1. コード・実行バイナリに対する**オフライン解析**
  2. **オンライン解析** (①実行バイナリを実行; ②ページフォルト起動; ③観測)
  3. 観測結果から**秘密を推定**

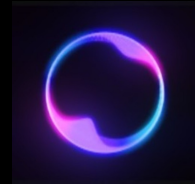
ページング



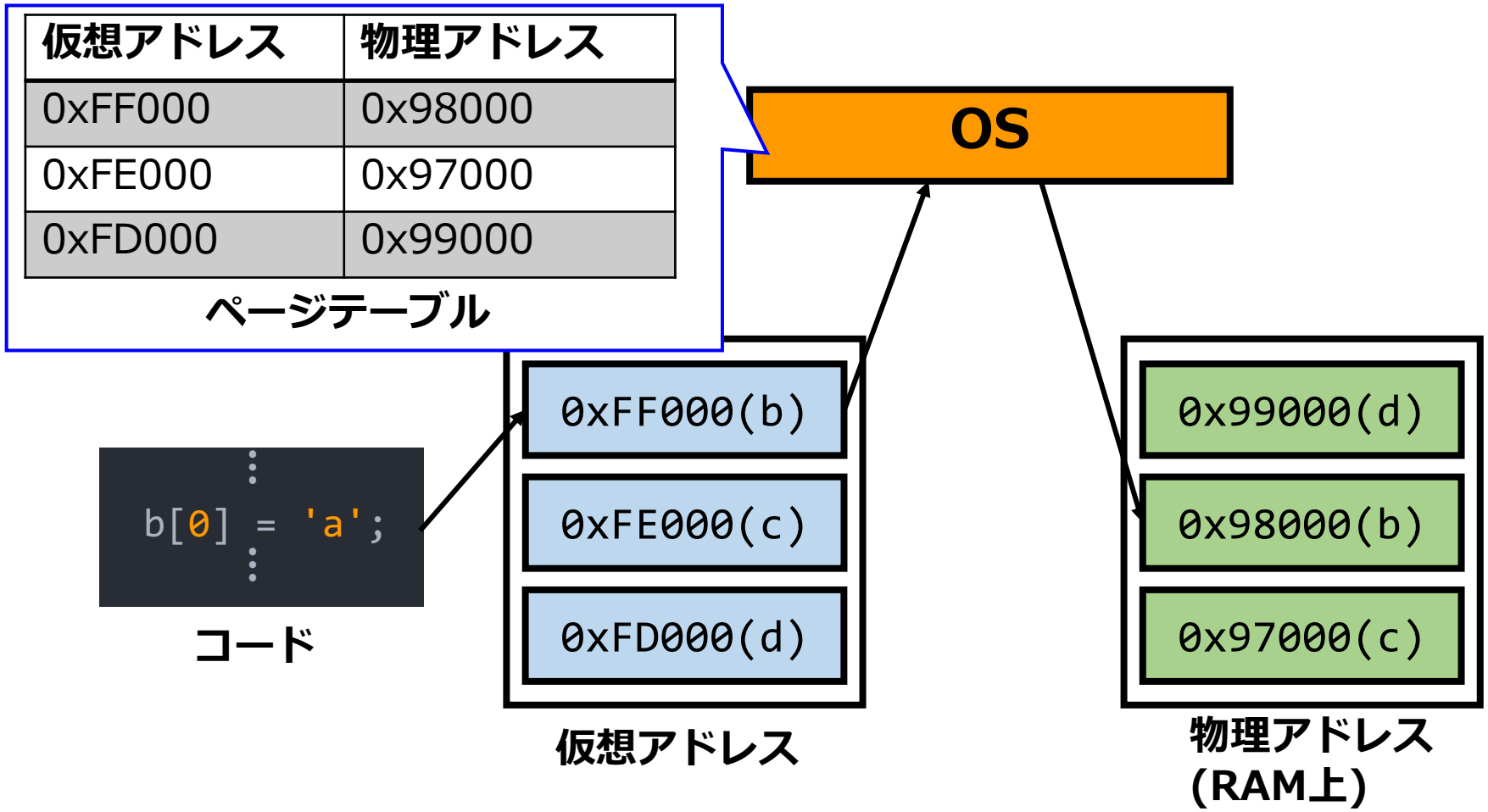
- 仮想記憶を実装するためのアルゴリズム



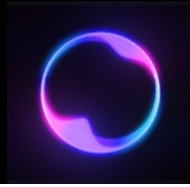
ページテーブル



- OSは**ページテーブル**を用いて**仮想アドレス**に対応する**物理アドレス**を取得する

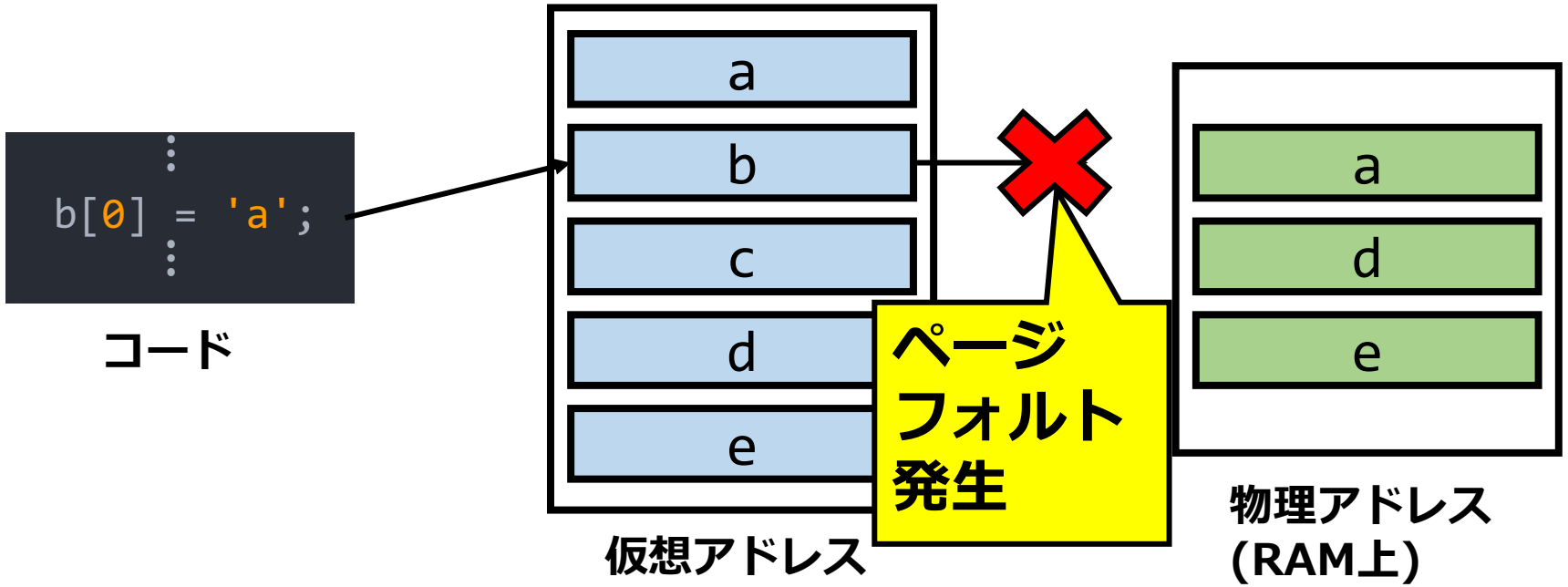


ページフォルト

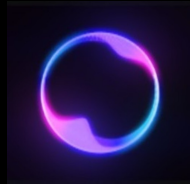


- **ページフォルト**: 以下の状況が発生した場合に発生する、**必ずしも致命的ではない**エラーの事

1. プログラムが**仮想ページ**にアクセスする
2. OSが何らかの原因 (ページアウト、**アクセス制限**等)により**物理アドレス**を**仮想アドレス**から**導出できない**



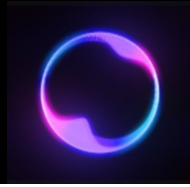
入力依存処理



- ある**処理実行**が**特定の入力データ**によって**条件的に決定される**時、それを**入力依存処理**と呼ぶ
  - 例: ifブロックに囲まれた処理

- 入力依存処理には**2種類**存在する：
  - 入力依存**制御転送**
  - 入力依存**データアクセス**

入力依存制御転送



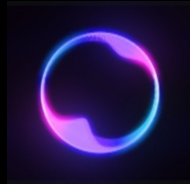
- **変数sによって<u>どちらの関数</u>にアクセスされるかが決定される時、それを「<u>入力依存制御転送</u>」と呼ぶ**

```
char* WelcomeMessage(GENDER s) {
  char *mesg;

  //GENDER is an enum of MALE and FEMALE
  if(s == MALE) {
    mesg = WelcomeMessageForMale();
  }
  else {
    mesg = WelcomeMessageForFemale();
  }

  return mesg;
}
```

入力依存制御転送

入力依存制御転送

入力依存データアクセス



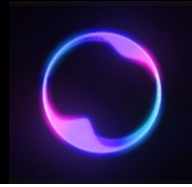
- **変数sによってどちらのデータにアクセスされるかが決定される時、それを「入力依存データアクセス」と呼ぶ**

```
void* CountLogin(GENDER s) {

  if(s == MALE) {
    gMaleCount++;
  }
  else {
    gFemaleCount++;
  }

  return;
}
```

入力依存データアクセス

入力依存データアクセス

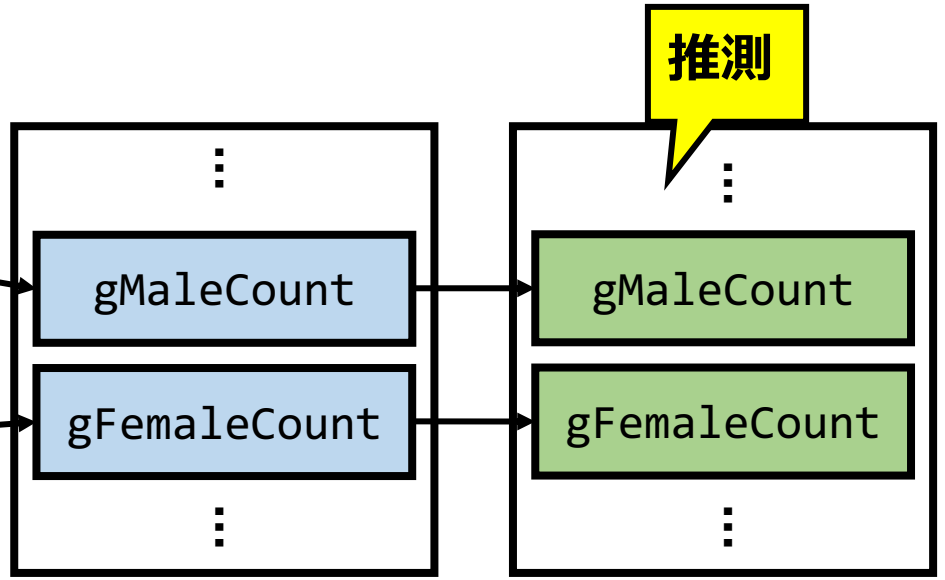
Phase 1. オフライン解析



- **実行バイナリ**や**ソースコード**を解析し、 gMaleCountや gFemaleCountがロードされる**アドレスを推測**する

```
void* CountLogin
(GENDER s) {

  if(s == MALE) {
    gMaleCount++;
  }
  else {
    gFemaleCount++;
  }

  return;
}
```

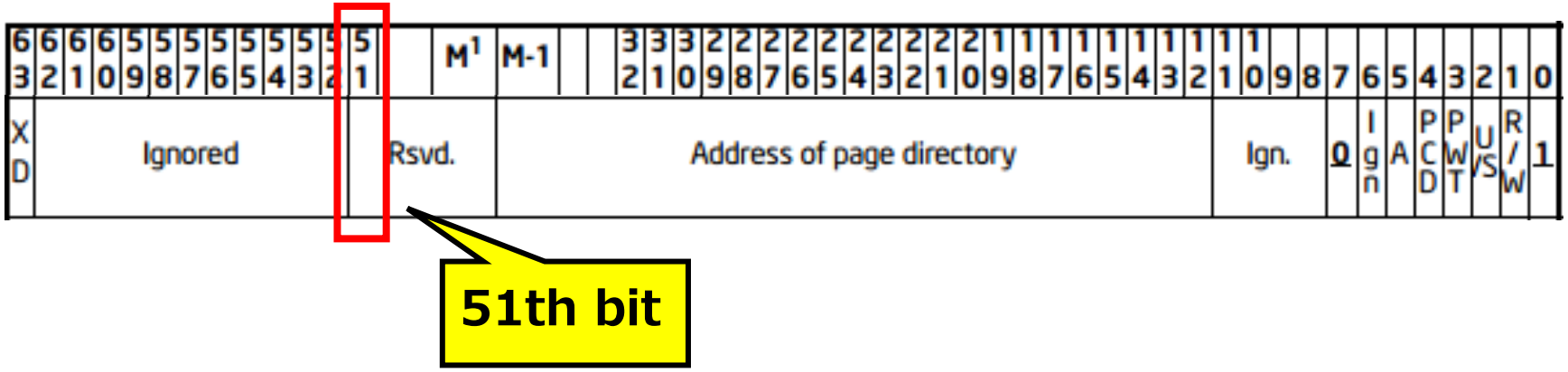
コード



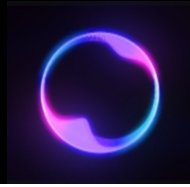
Phase 2. ページフォルト起動 (1/2)

- SGXはページテーブルを**OSと共有**している為、ページテーブルは**容易に改竄できる**

- X86-64のページテーブルエントリの**51ビット目**を立てる事で、**対応するページ**へのアクセスを**全て制限できる**

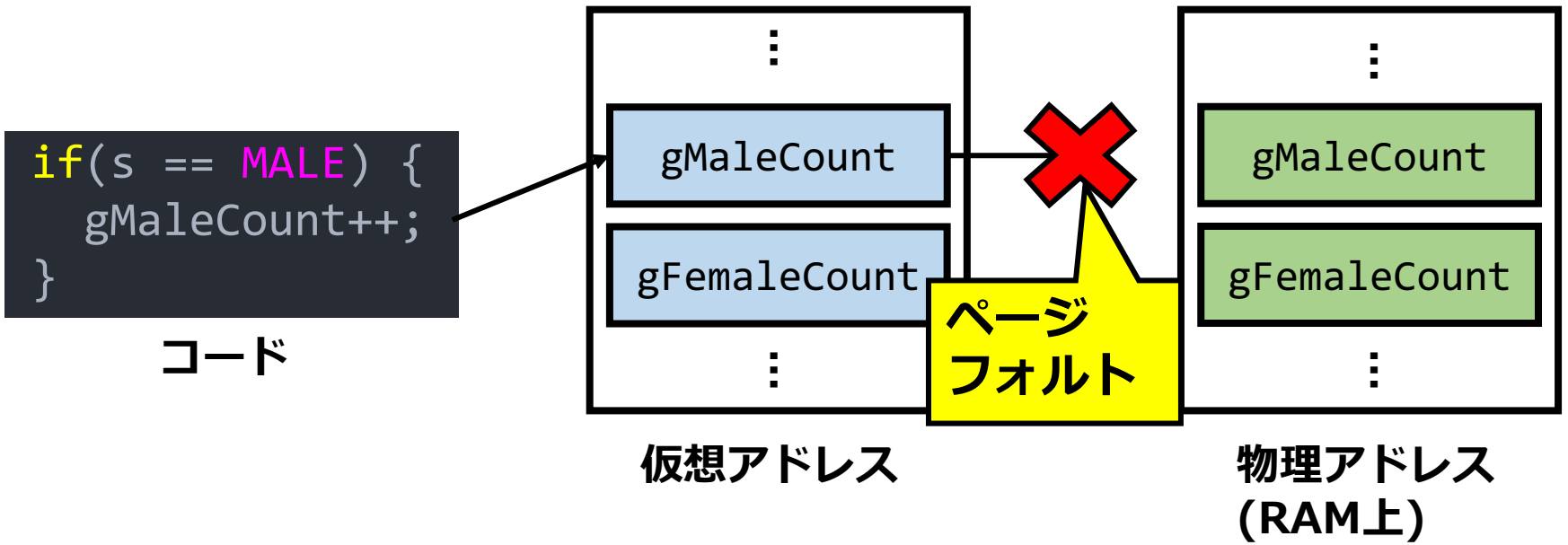


Phase 2. ページフォルト起動 (2/2)

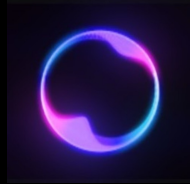


- ページテーブルエントリの**予約ビット**を立てる等により、gMaleCount及びgFemaleCountの載る**ページへのアクセス**を**禁止**する

- **禁止したページ**に載る**データへのアクセス**が発生すると、**ページフォルト**が**発生**する

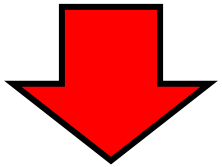


Phase 3. 秘密情報の推定



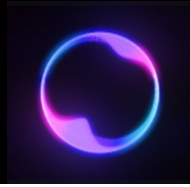
- **ページフォルト**が発生すると、**OS**の**ページフォルトハンドラ**に**ページフォルトアドレス**が伝達される

- 攻撃者は、**攻撃対象の秘密情報**が載っている**アドレス**(あるいは**ページ**)を**オフライン解析**にて取得済みである



ページフォルトアドレスをOSから取得する事により、gMaleCountへのアクセスが発生した事、ひいては“s”がMALEだった事も漏洩してしまう

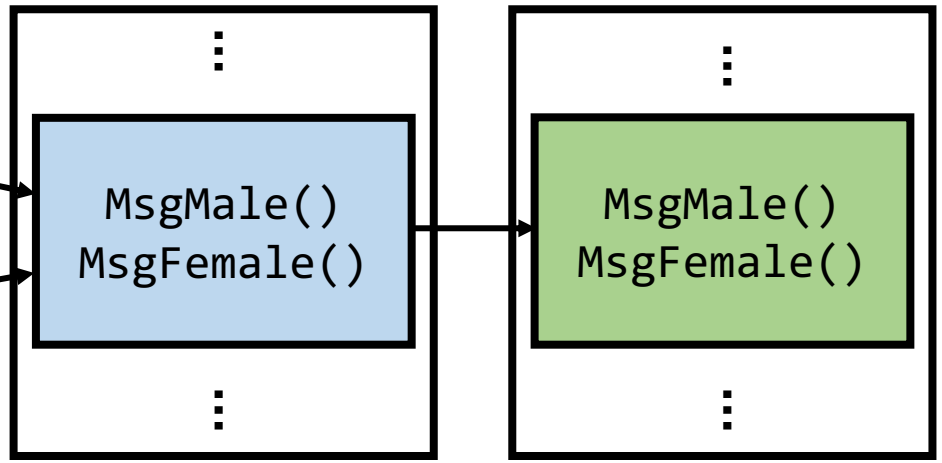
この攻撃を実行する上での困難



- **複数のコードやデータ**が**同一ページ上に載る**可能性が**多分に存在する**ため、その場合単にフォールトの発生したページだけを見るだけではどちらのコードやデータなのか**識別できない**

```
char* WelcomeMessage
 (GENDER s) {
  char *mesg;

  if(s == MALE) {
    mesg = MsgMale();
  }
  else {
    mesg = MsgFemale();
  }

  return mesg;
}
```

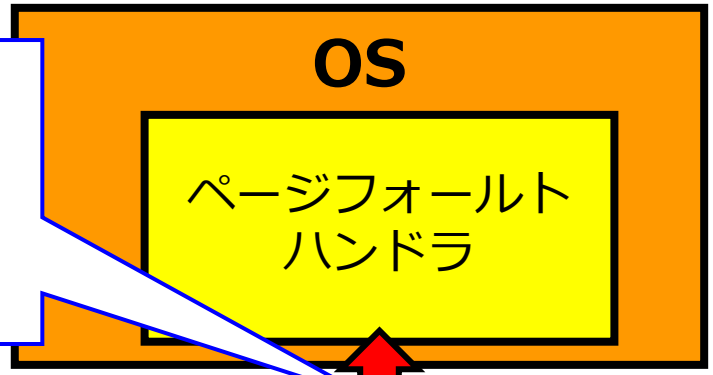
コード



SGXから返されるページフォールトアドレス

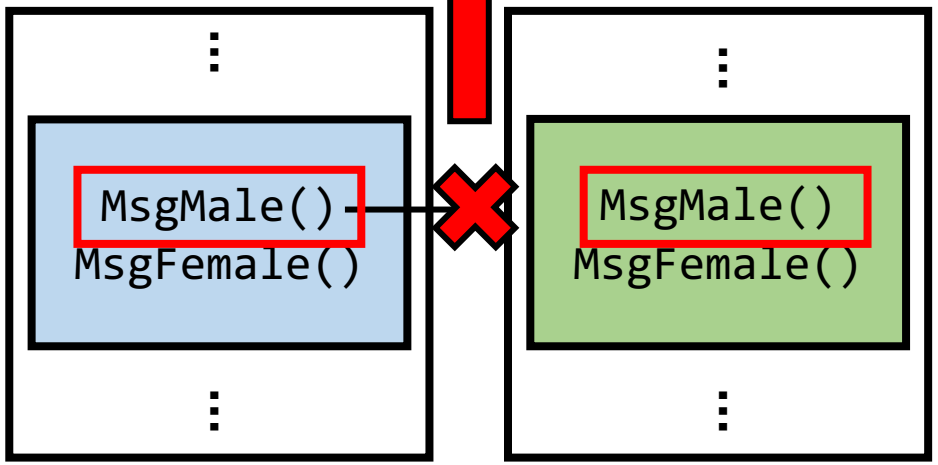
OSが制御可能

通知されるフォールトアドレスは
MsgMale()の完全なアドレスではなく、下位12ビットがゼロ化されている

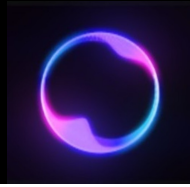


OSはページフォールトの発生したページ番号の粒度でしかアドレスを取得できない

OSは制御不能（Enclave）



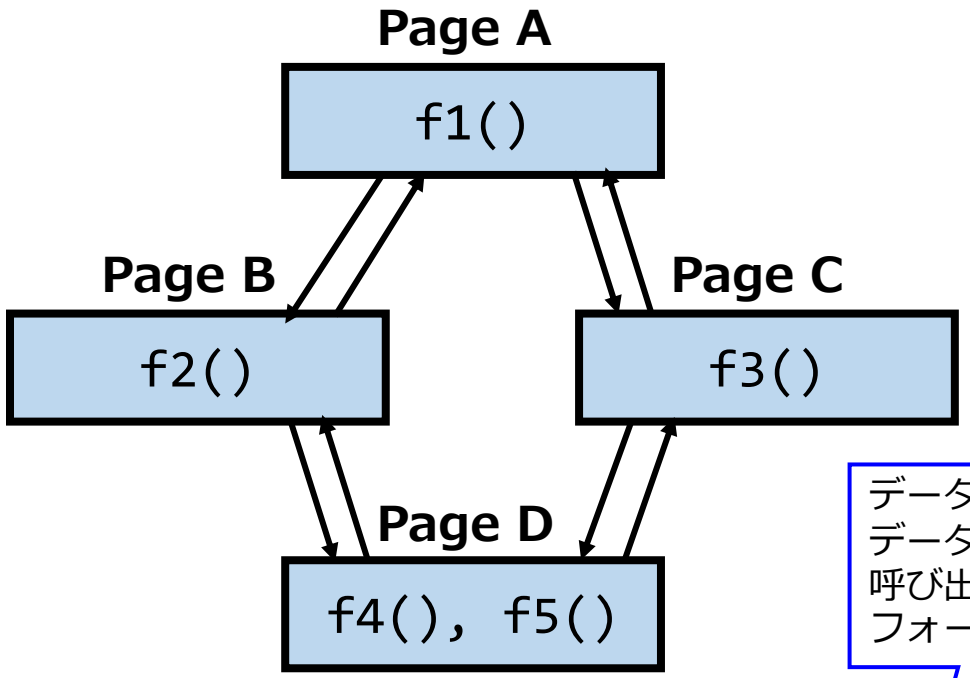
この攻撃を実行する上での困難



- **ページレベルの粒度**でしかページフォールトアドレスからは情報を得られないため、**複数の秘密情報**が**同一ページ上**に**載っている**と、このままでは**識別が出来ない**

- この問題を解決するには、次のページで説明する**ページフォールトシーケンス**という概念を用いる

ページフォールトシーケンス



ページレベルで見た場合の制御転送

データを攻撃対象とする場合は、
データアクセスの直前の関数
呼び出しに対応するコードページ
フォールトシーケンスを頼りにする

A, B, D, B, A, C, D, C, A

f4()　　f5()

コードページフォールトシーケンス

開始点

```
f1 {
   …
   f2();
   …
   f3();
   …
}
```

```
f2 {
   …
   f4();
   …
}
```

```
f3 {
   …
   f5();
   …
}
```

ソースコード

攻撃実践例 – Hunspell（スペルチェッカ）

- Hunspellにより参照される辞書ハッシュテーブルに対して攻撃し、秘密情報である文章を抽出

Folklore, legends, myths and fairy tales have followed childhood through the ages, for every healthy youngster has a wholesome and instinctive love for stories fantastic, marvelous and manifestly unreal. The winged fairies of Grimm and Andersen have brought more happiness to childish hearts than all other human creations.

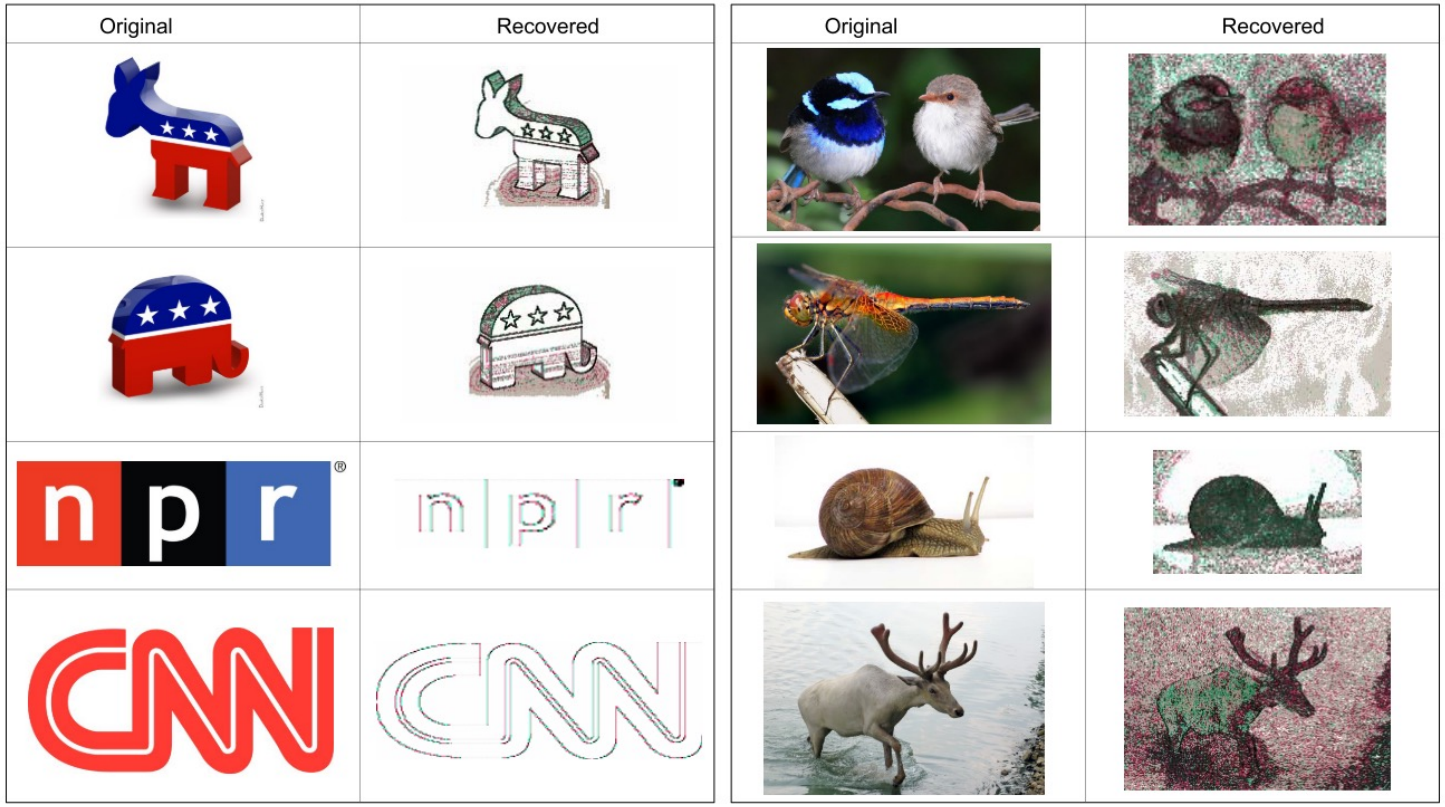
元の文章

folklore *legend* myths and fairy *tale* have *follow* childhood through the *age* for every healthy youngster has a wholesome and instinctive love for [store] fantastic marvelous and *manifest* unreal the [wine] *fairy* of [grill] and Andersen have brought more happiness to childish *heart* than all other human *create*

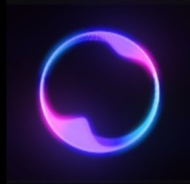
抽出した文章

攻撃実践例 – libjpeg

- JPEGファイルのロード時（デコード時）に呼び出される、逆離散コサイン変換を行う関数に対して攻撃し、秘密情報である画像データを抽出



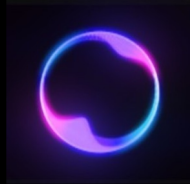
Controlled-Channel攻撃実践（1/6）



■**Controlled-Channel攻撃実践課題**
入力依存データアクセスを行うEnclaveにおいて、分岐先データの載るページのアクセス権を剥奪し、フォールトを発生させて秘密情報の断定を行う実験コードを実装せよ。

但し、OSのページフォールトハンドラを改竄したり中身を見たりするのは難易度が高いため、あくまでも**実験に最適化したEnclave**に対して攻撃を行う。

Controlled-Channel攻撃実践（2/6）



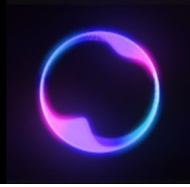
■Controlled-Channel攻撃実践の要件

- 攻撃対象Enclaveは、**uint8_t*型のポインタ2つをグローバル空間**に有している

- また、攻撃対象Enclaveは同じく**グローバル空間**に**1バイト**の**uint8_t型変数**である**秘密情報**を有する

- 攻撃対象Enclaveは、**初期化処理**を行うECALLと、**本処理**を行うECALLの**2つのECALL**を提供している

Controlled-Channel攻撃実践（3/6）



■ **Controlled-Channel攻撃実践の要件**

- 攻撃対象Enclaveの**初期化処理**では、**2つのグローバルポインタ**に対しそれぞれ**ページサイズ**分のバッファを割り当て、バッファを（非ゼロの）**適当な値**（'a'など）で**埋め尽くす**

- また、**sgx_read_rand**関数を用いて**uint8_t型**の**乱数**を生成する。uint8_tの性質上、この値は**0～255**のいずれかになる

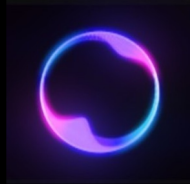
Controlled-Channel攻撃実践（4/6）

■Controlled-Channel攻撃実践の要件

- **乱数**が**128未満**である場合、**秘密情報**に**0を代入**する。**128以上**である場合は**秘密情報**に**1を代入**する

- 初期化処理ECALL時には、引数として**ページサイズ**を渡しても良い。同時に、Enclave外に**グローバルバッファの仮想アドレス**を**リターン**しても良い

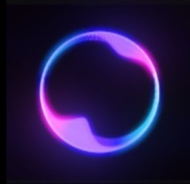
Controlled-Channel攻撃実践（5/6）



■Controlled-Channel攻撃実践の要件

- 攻撃対象Enclaveの**本処理**では、**秘密情報**が**0**であれば**一方のバッファ**に、**1**であれば**もう一方のバッファ**に**アクセス**する（＝**入力依存データアクセス**）

Controlled-Channel攻撃実践（6/6）

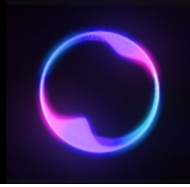


■Controlled-Channel攻撃実践のヒント

- ページアクセスの禁止は、**mprotect関数**をEnclave外で用いる事によって比較的容易に実現できる
  - mprotectに渡すアドレスは、**ページの境界のアドレス**に**一致**していなければならない点に注意

- mprotectでアクセスを禁じたページにアクセスすると**セグフォ**が発生する。この時Enclaveから返ってくるsgx_status_tの値は**SGX_ERROR_ENCLAVE_CRASHED**である
  - つまり、ページフォールトアドレスを見る必要はなく、**クラッシュしたか否か**で判定できるため、**片方の秘密バッファ**の載るページだけ**アクセスを禁じれば十分**

Controlled-Channel攻撃対策の導入実践

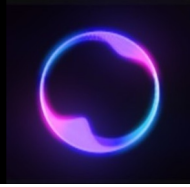


- Controlled-Channel攻撃は、攻撃対象コードに**条件分岐さえ存在しなければ無力**である
  - 条件分岐の排除を含め、タイミング攻撃を含むサイドチャネル攻撃の余地を排する実装方法を**定数時間実装**（Constant-time Implementation）と呼ぶ

- SGX-VaultのEnclave内コードにおいて、どれでも良いので**条件分岐を1つ選び、制御フローが単一**になるように**修正**せよ。

Plundervolt攻撃

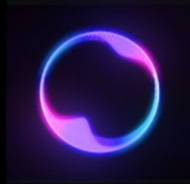
Plundervolt攻撃（1/6）



- 最近のCPUには、通常時は**クロック周波数**と**電源電圧**を**可能な限り低い状態**で保ち、必要時にはそれらを**動的にスケールアップ**する、電圧や周波数の**動的スケーリング機能**が備わっている
  - 常にフル稼働だと**過度な発熱**等の問題が発生する

- CPU（CMOS回路）の動的消費電圧は、クロック周波数と電源電圧の2乗との積に比例する（$P_{dyn} \propto f \cdot V^2$）
  - 周波数の増減に応じて電圧も増減するため、周波数や電圧を**独立で変更**すると**不具合が起きる**可能性がある

Plundervolt攻撃（2/6）

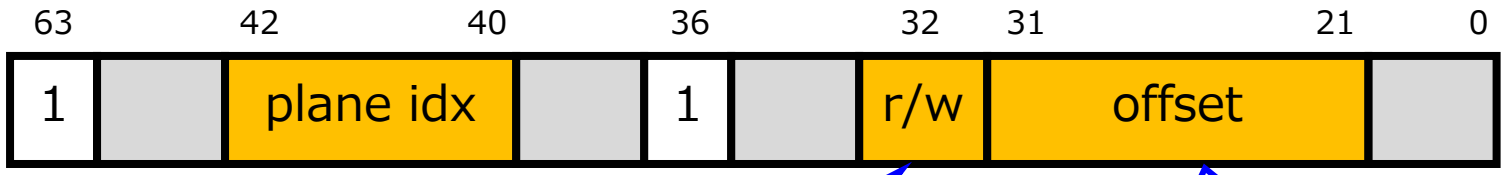


- もし動作電圧を急に下げると、下げた量に応じて**CPU内部の値**に**故障**（値の改変）が発生したり、酷い場合には**フリーズや クラッシュ**が発生する
  - 乗算をはじめ、**比較的複雑な命令**で**特に故障が発生しやすい**と論文で報告されている[3]

- ところで、最近のIntel製CPUでは、**モデル固有レジスタ**（MSR）のアドレス**0x150**を通して、**CPUの動作電圧**に**変更を加える事が出来る**
  - MSR：CPU内部の特殊な制御を行うためのCPU固有のレジスタ
  - このMSR自体はIntelによって明文化されていない「**隠し要素**」

Plundervolt攻撃（3/6）

- MSR 0x150の各ビットの役割は以下の通り：



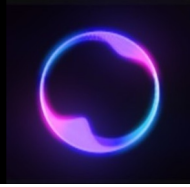
電圧を操作するハードウェア（plane）のインデックスを指定。

0: CPUコア
1: GPU
2: キャッシュ（コア内）
3: アンコア
4: アナログ入出力

書き込み可能ビット

11ビットの符号付き電圧オフセット。1/1024V単位で、-1Vから+1Vの範囲で動作電圧に加える変更値を設定できる

Plundervolt攻撃（4/6）



- この仕様を悪用し、**MSR 0x150**を通して**瞬間的**に**動作電圧を下げ**、Enclave内の**秘密情報**（に関連する値）に**故障を発生させる**攻撃が**Plundervolt**攻撃である
  - Plunder（略奪）+Undervolt（低電圧化）から成る鞄語
  - 一般に**故障注入攻撃**（Fault Injection Attack）と呼ばれる分類に属する攻撃
  - 攻撃者がOS含め攻撃対象のTCB外の全てを掌握している、**標準的なSGXの脅威モデル**に準じている

Plundervolt攻撃（5/6）

- CPU内の値に対して**直接故障注入を行う**ため、MEEのメモリ完全性検証により**改竄検知**を**行う事が出来ない**
  - 同じ物理寄りの攻撃でも、完全性検証に失敗させる事に絞っている SGX-Bombと比較した場合の大きな相違点でもある

- AzureのDCsv2ではマシンが既に**Plundervolt対策**されている上、そもそも**明文化されていないMSRに対する操作**をVMに実行させるのを**拒否している**はずなので、**本攻撃は通用しない**

Plundervolt攻撃（6/6）

- 当然、秘密情報に故障を発生させるだけでは秘密情報の抽出はできない
  - また、攻撃を行うにしても、タイミングの問題の対処など様々な工夫の上で成立する攻撃である

- **差分故障解析**という、正しい値と故障した値との差分を悪用し秘密情報を推定する攻撃に繋げたり、**配列インデックス等を故障させて不正な値を参照させる**ようにする事で**真価を発揮する攻撃**

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（1/10）

- RSA暗号では、p, qを相異なる素数、nを$n := pq$であるような値、ある整数eを選んだ上で$de \equiv 1 \pmod{(p-1)(q-1)}$となる整数$d$を用意する

- この上で、RSA暗号では(n, e)を**公開鍵**とし、d, p, qを**秘密鍵**とする
  - 公開鍵のnからはp, qの値は分からず、あくまでも1つの巨大数の形である

- 平文mの暗号化の際は$\boldsymbol{Enc(m) := m^e \bmod n}$とし、暗号文$c$の復号は$\boldsymbol{Dec(c) := c^d \bmod n}$のようにする
  - dを知らずにn, e, cから平文を計算するには、事実上**巨大数n**についての**素因数分解**を行わなければならない（$(p-1)(q-1)$の導出のためにp, qを導出するため）ので、その**困難さを安全性の根拠としている**

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（2/10）

- RSA-CRTは、**中国剰余定理**（**C**hinese **R**emainder **T**heorem）を使用して、復号処理である$\boldsymbol{Dec(c) := c^d \bmod n}$を高速化する、RSAの復号高速化手法である

- **中国剰余定理**：与えられた2つの素数p, qが互いに素であれば、任意の整数a, bに対し

$$x \equiv a \pmod{p}$$
$$x \equiv b \pmod{q}$$

を満たす、$0 \leq x < pq$であるような整数xが一意に存在する。

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（3/10）

- これをRSAの文脈に落とし込むためにa, bを暗号文関係の値に置き換えると

$$x \equiv c_p^d \bmod p$$
$$x \equiv c_q^d \bmod q$$

となる（但し$c_p :\equiv c \bmod p$, $c_q :\equiv c \bmod q$）

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（4/10）

- ここで、フェルマーの小定理について説明する。

■フェルマーの小定理

pを素数とし、aをpの倍数でない整数（aとpは互いに素）とすると、

$$a^{p-1} \equiv 1 \pmod{p}$$

が成立する

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（5/10）

- フェルマーの小定理を適用すると、$d_p :\equiv d \bmod (p-1)$, $d_q :\equiv d \bmod (q-1)$とした場合、前述の$c_p^d, c_q^d$は

$$c_p^d = c_p^{k_{dp}(p-1)+d_p} \equiv c_p^{d_p} \bmod p$$
$$c_q^d = c_q^{k_{dq}(q-1)+d_q} \equiv c_q^{d_q} \bmod q$$

となり、解きたい合同式は

$$x \equiv c_p^{d_p} \bmod p$$
$$x \equiv c_q^{d_q} \bmod q$$

となる

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（6/10）

- これにより、より小さい秘密鍵d_p, d_qを用いる事で、復号速度を4倍ほど高速化するのがRSA-CRTである

- 復号の際、実際にはRSA-CRTは以下の二項式を計算している：

$$x = \left[q \cdot s_p\right] \cdot c_p^{d_p} + \left[p \cdot s_q\right] \cdot c_q^{d_q} \pmod{n}$$

ただし、xは平文、s_p, s_qは予め算出した定数

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃（7/10）

- この二項式の片側に故障を発生させると、**Bellcore-Lenstra攻撃**という**差分故障解析**により**秘密鍵を抽出できてしまう**事が知られている

- よって、**Plundervolt**によって**片側の項に故障注入を行う**事で、**Enclave内で使用されているRSA-CRT**から**RSAの秘密鍵**を**抽出**する事が出来てしまう

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃 (8/10)

- この二項式の片側の項に故障が発生した、**誤った平文x'**が得られた時、以下の計算で**公開鍵n**を**秘密鍵p, q**に**因数分解できる**という手法が**Bellcore攻撃**である：

$$q = \gcd(x - x', n), p = \frac{n}{q}$$

ただしgcd()はユークリッド互除法等、最大公約数を算出する関数

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃 (9/10)

- さらに、このBellcore攻撃におけるqの計算式を

$$q = \gcd\left(x'^{e} - c, n\right)$$

とする事で、x'とx双方を計算する必要性を排除するのが**Lenstra法**である
  - Plundervoltの論文中だとこのLenstra法の式が**間違っている**ので注意。正しくはPlundervoltのPoCコード[8]を参照

- **Enclave内**で**RSA秘密鍵が保護**されており、**暗号文を送る**と復号して**平文を返してくれる**ようなシステムに対して有効な攻撃

Plundervolt攻撃実践例 – RSA-CRTへの鍵抽出攻撃 (10/10)

- Plundervoltでは、RSA-CRT復号時に適切なタイミングで**-225mV**の**アンダーボルト**を行い、その後元に戻す事で、Bellcore-Lenstra攻撃による**RSA 2048bit秘密鍵**の**完全な抽出に成功**している

- ちなみに、RSA-CRTはデフォルトではEnclave内APIとしては提供されていないが、Intel IPPで用意されているものを使えるよう設定する事で使用可能になる

ref-LEに対する攻撃（1/4）

- FLCにおいて使用されるLaunch Enclaveである**ref-LE**（SGX基礎編セクション参照）では、**起動許可対象**を列挙した**ホワイトリスト**を参照し起動許可判定を行う処理を行う

- この**ホワイトリスト**では、**起動許可対象**の**MRSIGNER**と、そのEnclaveが**PK**等への**アクセス権を持つかのフラグ**が列挙されている

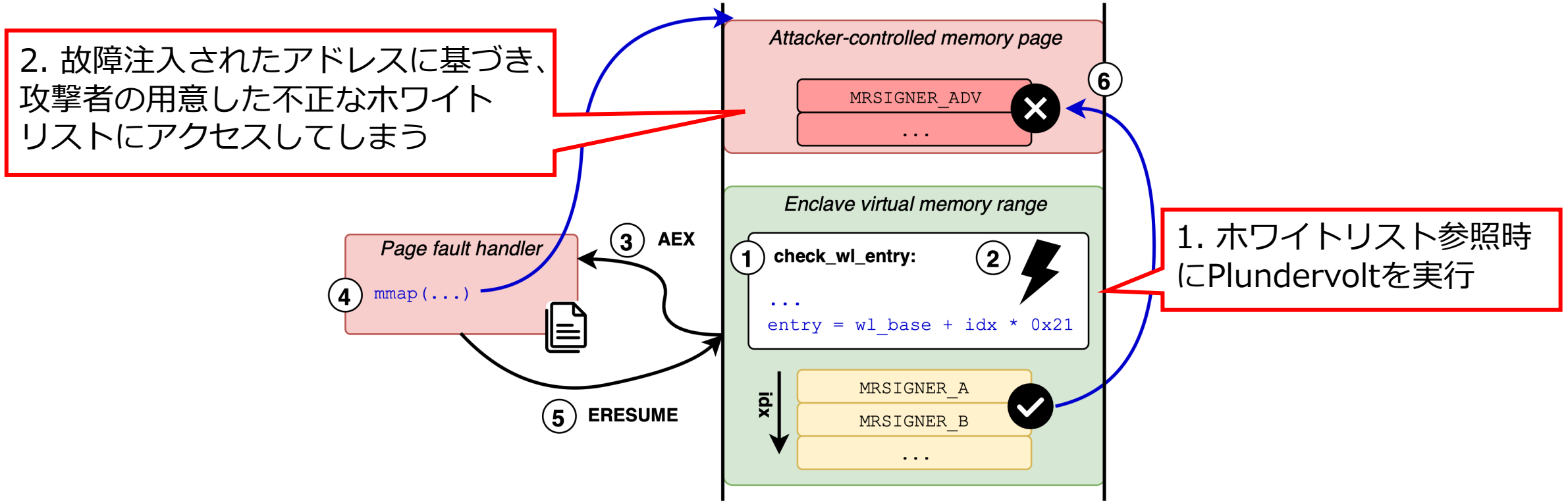
ref-LEに対する攻撃（2/4）

- ここで、Plundervoltにより**ホワイトリスト配列参照時**に見る**インデックスに故障注入**し、**攻撃者の用意**する**Untrusted領域上のバッファ**に**リダイレクト**させる事を考える

- これが実現すると、**攻撃者が用意した不正なホワイトリスト**を**誤って読み取ってしまい**、**不正な起動許可**どころか**不正にPKへアクセス**する事も許してしまう

ref-LEに対する攻撃（3/4）

- ホワイトリストのエントリが**53万件**ある場合に、**-118mV**の**アンダーボルト**をかけ、様々な工夫の末に**参照先リダイレクトに成功**している

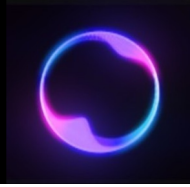


Plundervoltの論文[8]より引用

ref-LEに対する攻撃（4/4）

- ただし、**PKの導出**には**MRSIGNER**が含まれ、このMRSIGNERに対応するのは**IntelのプライベートなEnclave署名鍵**であるため、これにアクセスしなければ**PKの導出は不可能**であり、実質的には**PKへのアクセスは不可能に近い**という事実はある
  - 恐らくこの場合のMRSIGNERは**SECS**を参照するため、仮にAEからMRSIGNERだけを抽出していても、それを使い回す事は出来ないと想定される
  - 万一Intelの鍵が漏洩した場合、その鍵で署名を行う事でPKにアクセスする事が出来てしまう

本セクションのまとめ



- Controlled-Channel攻撃について解説を行い、実際にEnclaveのページアクセス権の拒絶に伴う挙動からサイドチャネル的に秘密情報を推測するPoCコードの実装を行った

- 電圧操作による故障注入攻撃であるPlundervoltについて解説し、ケーススタディとしてRSA-CRTに対する攻撃とref-LEに対する攻撃について説明を行った

参考文献（1/2）

[1]”Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems”, Yuanzhong Xu et al., https://ieeexplore.ieee.org/document/7163052

[2]”Intel SGX - Controlled-Channel Attacks解説”, 自己引用, https://qiita.com/Cliffford/items/f527fd210e3f7866e803

[3]”Plundervolt: Software-based Fault Injection Attacks against Intel SGX”, Kit Murdock et al., https://plundervolt.com/doc/plundervolt.pdf

[4]”CPU/MSR - SyncHack”, 2023/7/18閲覧, http://mcn.oops.jp/wiki/index.php?CPU%2FMSR

[5]”クラウドを支えるこれからの暗号技術”, 光成滋生, https://herumi.github.io/ango/

[6]”CRT-RSAとfault attack”, 2023/7/19閲覧, https://hackmd.io/@Xornet/ryoP8VxUw

[7]”中国の剰余定理 - wikipedia”, 2023/7/19閲覧, https://ja.wikipedia.org/wiki/%E4%B8%AD%E5%9B%BD%E3%81%AE%E5%89%B0%E4%BD%99%E5%AE%9A%E7%90%86

参考文献（2/2）

[8]”plundervolt/sgx_crt_rsa/Evaluation/eval.py”, GitHub,
https://github.com/KitMurdock/plundervolt/blob/master/sgx_crt_rsa/Evaluation/eval.py