AES鍵の複製と残存実態調査のためのFPGAを用いた周期的ライブメモリフォレンジック手法

FPGA-Based Periodic Live Memory Forensics Method for Investigating AES Key Duplication and Residual Presence

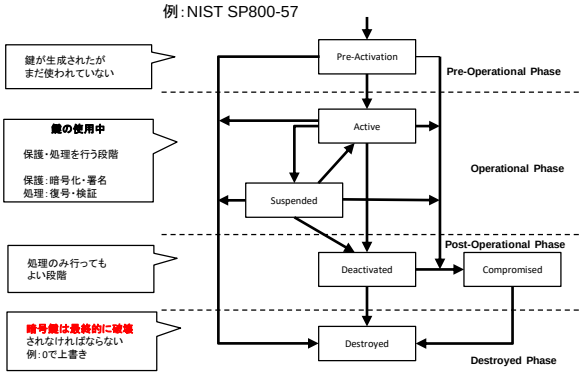
澤豊文・マネジメント分科会・情報セキュリティ大学院大学

Abstract: Cryptographic keys must not be unnecessarily duplicated and must be erased from memory after use, as required by standards like ISO/IEC 19790 and FIPS 140. Verifying key erasure is challenging due to interactions between applications and the operating system. While methods for key erasure exist, research confirming actual erasure remains insufficient. To address this gap, we propose a live memory forensic method using FPGA to periodically detect AES keys from systems running operating systems. Using this method, we investigated key erasure practices in AES-using applications on aarch64 and x86_64 platforms with Ubuntu Linux, including FIPS-compliant kernels, and on Windows 11 in an x86_64 environment. Our findings revealed several issues with key management. Keys are not erased during abnormal process termination. We also observed that keys may be included in core dumps, which could be sent externally. We reported the identified issues to application developers, along with clear and practical mitigation strategies.

背景

暗号鍵は使用後に復元できないように消去されなければならない

例：NIST SP800-57



ISO/IEC 19790やFIPS 140のような規格・ガイドラインで鍵の消去が要求される。
メモリ開示攻撃のAttack Surfaceを局限するために、暗号鍵の寿命が不必要に長期化しないようにするための知見を獲得したい。

課題

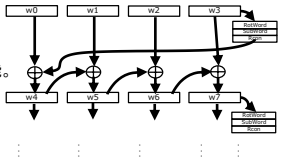
鍵の消去を実機で高速に検査したい

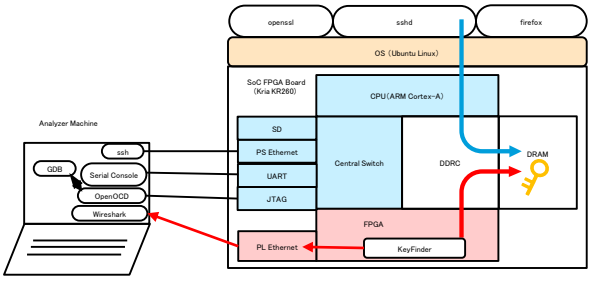
アプリは各コンポーネント（ライブラリ、OS、ファームウェア、ハードウェア）に依存して動作するが、アプリ開発者がこれらの相互作用を作用を理解しているとは限らない。そこで、鍵が消去されずに残存したり、複製される可能性がある。
鍵の消去の検証は難しい。既存研究はソースコードやバイナリレベルの静的解析が主で、システム全体の動作を観測する手法とその相互作用を調査した研究が不足している。
既存の動的解析手法は1つのプロセスのアドレス空間のみを対象としたり、高速に通信を行うにはオーバヘッドが大きい。また、暗号ライブラリ開発者は複数のアーキテクチャに対応して開発することもあるため、アーキテクチャに依存しない検証手法があると便利。
鍵の残存と複製の実態を調べるには、動作中のプロセスに影響を与えずに検査を行いたい。

提案手法

FPGAを活用したライブメモリフォレンジック手法

CPUと共有するメモリをFPGAが約3.96GB/秒の速度で周回してスキャンすることで鍵を検出する。
CPUを使わず、高速に動作中の鍵のライフタイムを計測可能。
高速な通信を行うプロセスにも影響を与えにくい。
AESの鍵スケジュールの特徴に注目している。
aarch64とx86_64の2種類の環境を対象に実装した。





調査結果

10種類の組み合わせ（アプリ、OS）と4種類のプロセス終了シナリオを実験
さらに、シグナルでコアダンプが生成されたときのコアダンプに含まれる鍵を調査

表：鍵の消去実態調査結果

正常終了時にはプロセス終了後に鍵が消去されていたが、異常終了時には鍵が残存していた。

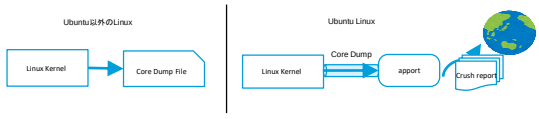
✔：鍵が消去される　✘：鍵が残存する　-：対象外

鍵の出現と消滅をタイムラインに表示

図：鍵の検出状況タイムライン

異常終了時にはプロセス終了後も鍵が残存し、しばらく後に消える
検査対象のプロセスのアドレス空間以外にも鍵が複製されていた

プロセス終了後も鍵が観測され続け、別のアドレスに複製された鍵も見られる。
これは鍵を含むコアダンプがクラッシュレポートとして外部送信されるときの挙動だった。



アプリ開発者に緩和策と共に報告し、対応したとの返答を得た。
例：https://github.com/openssh/openssh-portable/commit/cc048ca536d6

OSの再起動後にも鍵が残存するか？動作環境とカーネルの実装次第で異なる結果

表：OS再起動後の鍵の消去実態調査結果

FIPS対応カーネルは再起動時にメモリをゼロ化しており、よくできている。
Windowsは異常終了時に鍵を消すのに、再起動後、鍵が残存した。
シャットダウン処理が開始すると、ゼロ化したメモリが必要になる見込みがないため、プロセスを強制終了した後にメモリを消さないのではないか？

今後

①ツールのオーバヘッドの評価
②対応アルゴリズムの拡大
③Thunderboltを活用してMacに対応
④Sweet Spotの調査
⑤Windowsで大量に見られた鍵の由来調査