Design and Analysis of Memory Access Pattern Protection

仲野, 有登

（別紙様式２）

氏　　名　：仲野有登

論文題名　：Design and Analysis of Memory Access Pattern Protection
（メモリ・アクセスパターン保護の設計と評価）

区　　分　：　㊙　乙

論　文　内　容　の　要　旨

ソフトウェアの実行に必要なデータはメモリ上に一時的に格納されるため、これを狙った攻撃がいくつか知られており、ソフトウェアに対する大きな脅威となっている。また、ソフトウェアの脆弱性によって重要なデータが漏洩することも懸念される。ソフトウェアが扱う機密情報を保護するためには、これらの脅威への対策が必要であり、その一つとしてOblivious Random Access Machine(ORAM)が提案されている。ORAMを利用することで、攻撃者がメモリを監視できたとしても、メモリアクセスのパターンを秘匿することが可能であり、ソフトウェアが扱う情報を保護することが可能となる。しかし、既存のORAMはオーバーヘッドが大きいという課題があった。そこで、オーバーヘッドを削減した手法を提案するとともに、従来よりも強い攻撃者に対するORAMの安全性評価を行う。

第1章では、背景について述べ、成果および構成について説明する。

第2章では、ソフトウェアに対する脅威について述べ、対策として提案されているORAMの研究動向を整理する。

第3章では、メモリダンプを用いた攻撃の効率化手法を検討する。メモリダンプによってソフトウェアが利用するデータを取得することが可能であるが、メモリ上のすべてのデータの検索が必要であるという課題があった。そこで、ソフトウェアのアクセスを観測することでデータの検索を不要とする攻撃を示す。具体的にはソフトウェアが行うメモリアクセスについて、各アドレスへのアクセス回数を数え上げることで検索を不要とする。例として、RSAを対象に、鍵を固定し乱数の暗号化と復号を繰り返し実行する。この時、鍵が格納されているアドレスに対してアクセスが集中することを示す。実験では暗号化・復号の処理を10回繰り返した場合に、鍵の特定が可能である。また、AESを対象とした実験では鍵を固定し、乱数の暗号化を行った場合に、秘密鍵の特定が可能である。

第4章では、従来のORAMに比べて高速なメモリ・アクセスパターン保護手法の提案を行う。既存手法はORAMストレージ内のデータを一定周期でシャッフルする必要があり、処理負荷が高いという課題があった。そこで、一定周期のシャッフルを省略することで高速化を実現する。シャッフルなしの場合でも、ソフトウェア実行中に1

度しかアクセスされないデータについては、安全性を確保可能である。しかし、ソフトウェア実行中に複数回アクセスされるデータは、ソフトウェアにとって必要なアクセスである可能性が高く、保護する必要がある。そこで、提案手法では、アクセスの履歴を記録しておき、各アクセスにおいて、履歴として登録されているデータに対してダミーのアクセスを実行する。これによって、定期的なシャッフルを行わずに、アクセスパターンを秘匿することを可能にし、高速化を実現する。さらに、提案手法を実装する際の課題となる、データの効率的な管理手法、安全な領域の構築手法、ストレージの効率的な利用手法、について解決策を示す。さらに、既存手法に比べて最大約5倍高速であることを実験的に示す。

第5章では、より強い攻撃者に対してORAMの安全性評価を実施する。一般的にORAMの安全性は、アクセスパターンを監視するだけで、データの変更は行わない攻撃者(受動的な攻撃者)を想定して評価されている。しかし、攻撃者はデータの変更も可能であることが多く、データを変更可能な攻撃者(能動的な攻撃者)に対する安全性評価が課題であった。そこで、能動的な攻撃に対するORAMの安全性を評価する。攻撃者は、ソフトウェアの初期状態を記録し、さらに内部状態と出力を記録しながら1ステップずつ処理を実行する。その後、ソフトウェアを初期状態に戻し、データを1つ選択し、それを変更する。その後、ソフトウェアの挙動に変化が生じるかどうかを確認しながら、1ステップずつ実行する。挙動に変化が生じた場合、どのデータを変更した場合にどのステップで変化が生じたかを記録し、初期状態に戻す。この操作をすべてのデータに対して繰り返すことで、各データがどのステップで利用されるかが特定可能である。PathORAMを適用したAESに対して本手法を適用し、鍵の格納先を特定可能であることを示す。

第6章では、ORAMの適用先としてPC等のストレージの暗号化を挙げ、研究の動向をまとめる。単にストレージ全体を暗号化しただけでは、暗号化領域の存在を秘匿することはできず、何らかの情報が保管されている可能性を攻撃者に対して秘匿できない。そこで暗号化領域の存在そのものを秘匿する機能を実現したソフトウェアが公開されているが、アクセスのパターンを攻撃者が観測可能な場合は、暗号化領域の存在を検知可能であることが指摘されている。暗号化領域の存在を攻撃者から秘匿するためには、暗号化領域へのアクセスパターンを秘匿する必要があり、このためにORAMが活用されている。

最後に、第7章で得られた成果に関するまとめを行う。
(1989字)