平成 25 年度 卒業論文概要			
所　　属	機械情報工学科	指導教員	光來 健一
学生番号	10237059	学生氏名	福田直人
論文題目	メモリ暗号化による Android 端末の盗難対策		

1　はじめに

近年、Android 端末が急速に普及している。スマートフォンやタブレットなどの Android 端末は従来の携帯電話より多くの情報を保持することができ、より重要な情報も格納されるようになってきている。その一方で、ノート PC より小型軽量であるため、盗難にあうリスクも高い。Android ではディスク暗号化の機能を提供しており、端末が盗まれた場合でもディスクからの情報漏洩を防いでいる。

しかし、近年、コールドブート攻撃と呼ばれる攻撃手法が報告され、メモリからの情報漏洩が問題となってきている。Android では高速化のために、ディスク上のデータをメモリ上にページキャッシュとして保持している。端末を盗まれてコールドブート攻撃が行われると、ページキャッシュ上のデータを盗み見られてしまう。その結果、暗号化されているディスク上の一部のデータが漏洩してしまう危険がある。

本研究では、メモリ上のページキャッシュを暗号化することで Android 端末の盗難時に情報漏洩を防ぐためのシステム Cache-Crypt を提案する。

2　ページキャッシュからの情報漏洩

Android ではセキュリティ対策としてフルディスク暗号化が提供されている。フルディスク暗号化とは、ディスクのパーティション全体を暗号化することでデータを保護する仕組みである。Android ではアプリのデータが置かれたパーティションが暗号化される。暗号化されたパーティションを復号するには、OS 起動時に PIN を入力する必要がある。そのため、PIN を知られない限り、ディスクを取り外して別のマシンに取り付けても情報を盗み出すことはできない。

一方、Android を含め多くの OS は高速化のためにディスク上のデータをメモリ上にキャッシュとして保持している。特に、ファイルの内容はページキャッシュと呼ばれるキャッシュに保持される。アプリがファイルを読み込む際には、まず、OS がディスク上のデータをメモリ上のページキャッシュに読み込む。そして、ページキャッシュ上のデータがアプリに返される。次にアプリが同じファイルにアクセスする時には、ディスクへのアクセスを行わず、ページキャッシュ上のデータを即座に返すことができる。メモリ上のデータは端末を盗んだとしても盗み見るのは困難であるため、従来はメモリからの情報漏洩への対策はあまり行われてこなかった。

しかし、近年、比較的容易にメモリから情報を盗み出せるコールドブート攻撃が報告されている [1]。コールドブート攻撃は、メモリを冷やした状態で端末をリセットし、別の OS で再起動してメモリ上に残されたデータを盗み見る攻撃である。端末のリセットにより電源供給が途絶えている間にメモリ上のデータは次第に破壊されていくが、メモリを冷やすことで破壊されるのを遅らせることができる。

Android 端末においてもコールドブート攻撃が報告されている [2]。端末をリセットして USB 経由でリカバリイメージをインストールし、リカバリイメージから起動してメモリ上のデータを抜き出すことができる。コールドブート攻撃を用いてメモリ上のデータを盗み見られると、ページキャッシュ経由でディスク上のデータの一部も漏洩する危険がある。

3　Cache-Crypt

本研究では、メモリ上のページキャッシュを暗号化することでコールドブート攻撃によるページキャッシュからの情報漏洩を防ぐシステム Cache-Crypt を提案する。Cache-Crypt はディスク上のファイルをメモリに読み込む際にページキャッシュの内容を暗号化する。これを暗号化ページキャッシュと呼ぶ。アプリがアクセスするときだけ暗号化ページキャッシュを復号し、アクセスが終わったら再び暗号化する。Cache-Crypt を用いることで情報漏洩の対象をアクセス中のページキャッシュのみに限定することができる。また、ページキャッシュの暗号化に用いる暗号鍵はメモリ上ではなく、使われていない CPU レジスタ上に置くことで保護する。CPU レジスタはシステムの起動時に初期化されるため、コールドブート攻撃により暗号鍵が漏洩する恐れはない。

3.1　ページキャッシュの暗号化フラグ

Cache-Crypt ではページキャッシュを 4KB のページ単位に分割し、暗号化フラグを用いてページキャッシュの状態を管理する。暗号化フラグが ENCRYPT であればページキャッシュの内容が暗号化されていることを表し、DECRYPT であれば暗号化されていないことを表す。また、暗号化フラグが MEMORY_MAP の時は、アプリがそのページキャッシュに対応するファイルをメモリマップしていることを表す。この時、ページキャッシュの内容は暗号化されていない。これら 3 つの状態の間の関係は図 1 のようになる。

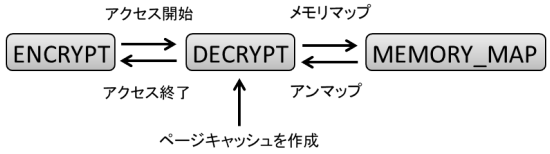

図 1　暗号化フラグの状態の間の関係

3.2　ファイルの読み込み

アプリがファイルを読み込む際には、図 2 のように、OS によってページキャッシュ上のデータがアプリのバッファにコピーされる。Cache-Crypt はページキャッシュの暗号化フラ

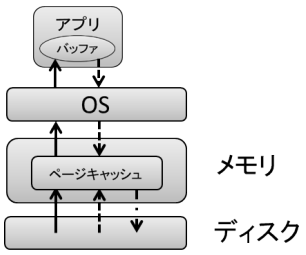

図 2　ファイルの読み書きの流れ

グが ENCRYPT であれば、まずその内容を復号し、暗号化フラグを DECRYPT に変更する。この時にまだ、ページキャッシュが存在していなければディスクからファイルを読み込み、ページキャッシュを作成する。ただし、暗号化フラグは DECRYPT とし、暗号化は行わない。そして、暗号化されていないページキャッシュ上のデータをアプリのバッファにコピーする。

その後、ページキャッシュを再び暗号化し、暗号化フラグを ENCRYPT とする。ただし、暗号化フラグが MEMORY_MAP となっており、ページキャッシュがメモリマップされている場合には暗号化を行わない。

3.3　ファイルへの書き込み

アプリがファイルに書き込む際には、図 2 のように、OS によってアプリのバッファ上のデータがページキャッシュにコピーされ、その後でディスクに書き込まれる。Cache-Crypt はファイルの読み込み時と同様に、ページキャッシュの暗号化フラグが ENCRYPT であれば復号し、ページキャッシュが存在しなければ必要に応じてディスクから読み込む。いずれの場合も暗号化フラグは DECRYPT とする。そして、アプリのバッファ上のデータをページキャッシュにコピーしてから、ページキャッシュを再び暗号化し、暗号化フラグを ENCRYPT にする。ただし、暗号化フラグが MEMORY_MAP の場合には暗号化を行わない。

OS は適切なタイミングで変更されたページキャッシュの内容をディスクに書き戻す。この際に、ページキャッシュの暗号化フラグが ENCRYPT なら、まずページキャッシュを復号し、暗号化フラグを DECRYPT にする。ページキャッシュ上のデータをディスクに書き戻した後、ページキャッシュを再び暗号化し、暗号化フラグを ENCRYPT にする。ただし、暗号化フラグが MEMORY_MAP の場合には暗号化を行わない。

現在のところ、ディスクへの書き戻しの際の復号化・暗号化は未実装であるため、ファイルへの書き込み後の暗号化を行わないようにしている。

3.4　ファイルのメモリマップ

アプリがファイルをメモリマップすると、図 3 のように、そのファイルに対応するページキャッシュについては OS を介さずに直接アクセスできるようになる。そのため、アプリがファイルにアクセスするたびに OS が復号化・暗号化を行うのは難しくなる。そこで、アプリがメモリマップしたファイルに最初にアクセスした時にページフォールトが発生することに着目する。ページフォールトが発生した時に、対応するページキャッシュの暗号化フラグが ENCRYPT であれば復号を行い、暗号化フラグを MEMORY_MAP にする。暗号化フラグが

表 1　ベンチマーク結果（MB/秒）

	標準カーネル	Cache-Crypt
Write	113.5	102.8
Read	562.1	503.8

MEMORY_MAP になったページキャッシュはファイルの読み書き後に暗号化されない。そして、アプリがファイルをアンマップしメモリマップを終了した時に、ページキャッシュを再び暗号化する。

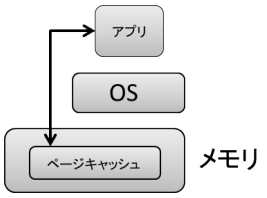

図 3　ファイルのメモリマップ

4　実験

Cache-Crypt がページキャッシュを暗号化できていることを確認する実験を Android エミュレータを用いて行った。まず、テキストエディタアプリで大量の”A”の文字からなるファイルを読み込み、メモリにキャッシュさせた。次に、アプリを完全に終了させてからエミュレータのメモリをダンプして、文字の部分を抜き出した。その結果、大量の”A”の文字が暗号化されていることが確認できた。

Cache-Crypt によるオーバヘッドを調べる実験を Nexus 7(2013) を用いて行った。この実験には Benchmark というアプリを用いてベンチマークを実行し、Google が提供する標準カーネルと Cache-Crypt を標準カーネルに実装したものとで比較を行った。ベンチマーク結果を表 1 に示す。読み書きともに Cache-Crypt を用いることによるオーバヘッドは 10% 程度であった。

5　まとめ

本研究では、メモリ上のページキャッシュを暗号化することでコールドブート攻撃による情報漏洩を防ぐシステム Cache-Crypt を提案した。Cache-Crypt はアプリがアクセスする時だけページキャッシュを復号し、それ以外の時は暗号化したままにする。Cache-Crypt を Android OS に実装し、ページキャッシュが正しく暗号化できていることを確認した。

今後の課題は、高度な暗号化を実装し、そのオーバヘッドを削減する手法を考案することである。

参考文献

[1] J. Halderman et al., Lest We Remember: Cold Boot Attacks on Encryption Keys, In *Proc. USENIX Security'08.*

[2] T. Müller et al., FROST – Forensic Recovery of Scrambled Telephones, In *Proc. ACNS'13.*