スマートデバイスの堅牢化ガイド

β版

2012年6月5日

日本スマートフォンセキュリティ協会
技術部会 デバイスワークグループ デバイス堅牢化タスクフォース

本書では、OSの設計方針に反する不正なアプリケーションが実行された場合においても、OSやアプリケーションが設計意図通りに動作するために配慮すべき事項を提示する。これは、ユーザの故意や不注意によって引き起こされる事故による影響を抑止する狙いがあり、デバイスの堅牢化において注目すべきポイントを示すものである。

文書管理番号： JSSEC-TECD-DS-GD201205188

【製作】

技術部会 デバイスワークグループ デバイス堅牢化タスクフォース

リーダ	重田大助	シャープ株式会社
メンバ	竹森敬祐	KDDI 株式会社
	川上稔彦	ソニーモバイルコミニュケーションズ株式会社
	北島真理子	ソニーモバイルコミュニケーションズ株式会社
	倉林俊介	トヨタ自動車 株式会社
	八津川直伸	日本ユニシス株式会社
	二村廉太	株式会社ネクストジェン
	松崎なつめ	パナソニック株式会社

目次

1. はじめに

1.1. 本ガイドの利用にあたって

近年スマートフォンやタブレットといったスマートデバイスはその自由度の高さからさまざまな用途に利用されている。一方で自由度が高いために、デバイスの設計者、ソフトウエアの開発者が意図しない（悪意ある）動作をさせることが容易なデバイスでもある。Android OS で例えるならば、通常利用では書き換えられることのない/system 領域にマルウェアが組み込まれることで、システム権限による端末の踏み台化が懸念される。これに対して、/system 領域を保護するなど、スマートデバイス側で一定の配慮を施すことで、デバイスの悪用をある程度低減することが可能である。本ガイドではこうした考え方にもとづいて、デバイスの堅牢化に関するポイントを整理する。

なお、本ガイドは 2011 年 12 月 31 日現在のβ版であり、記載された内容は今後変更の可能性がある。また、本稿に記載の内容に従って対策を施した際に生じる事故に対して、いかなる責任も JSSEC が負うものではない。

1.2. 本ガイドの対象デバイス

Android などの OS を搭載したスマートフォン、および、タブレット

1.3. 本ガイドの目的

本ガイドは、情報がオープンで、汎用的であり、更新頻度の高い OS を搭載したデバイスを、いかに堅牢化するかについて、デバイスの開発者、運用管理者向けに、指針を提示することを目的とする。

このため、本ガイドに記載されている項目を全て実装することを求めるものではない。また、デバイスの堅牢度は増すものの、攻撃と対策の繰り返しで進む技術分野であることから、将来に渡って保証できるものでもない。

1.4. 本ガイドが対象とする読者

- スマートデバイスを開発するデバイスメーカ
- スマートデバイスの利用を検討している法人

1.5. 本ガイドの対象とする範囲

OS が設計通りに動作すれば、アプリケーションも設計通りに動作することにつながり、その上で実現されるさまざまなサービスが安全に実行できるようになる。こうしたデバイスを実装する上で考えるべき点について本ガイドに記載する。

尚、アプリケーションや Mobile Device Management（MDM）によるデバイスのセキュリティ確保に関する手法については本ガイドの範囲には含めない。

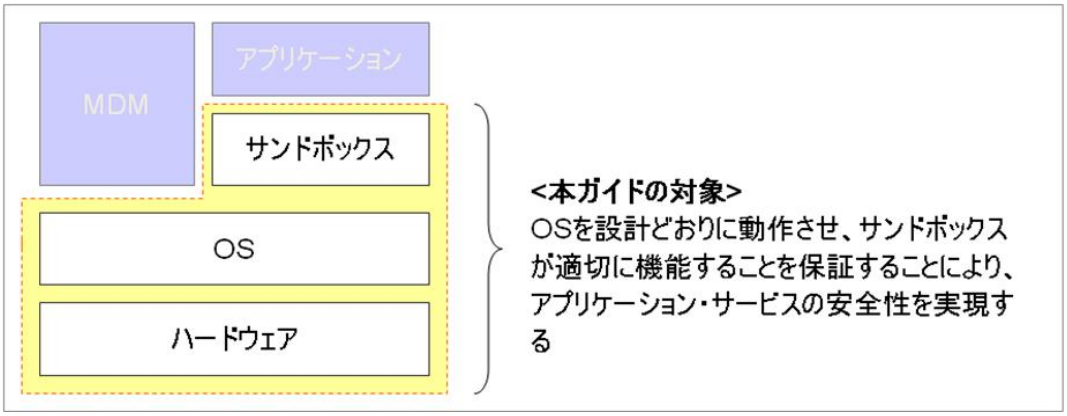

図 1本ガイドの対象とする範囲

2. デバイスの堅牢化とは

2.1. スマートデバイスの OS におけるセキュリティモデルとリスク

スマートデバイスに搭載された OS には、各アプリケーションに対して必要以上に実行権限を与えない仕組み(サンドボックスと呼ばれる)が実装されている。サンドボックスが機能し続ける限りにおいてはデバイス上で動作するアプリケーションは OS の設計通りに実行され安全であると言える。逆にこのサンドボックスの仕組みが崩されてしまえばデバイスの安全性は低下してしまうことになる。サンドボックスを無効化できるようなソフトウエアの不具合は脆弱性と呼ばれ、こうした不具合を少なくすることでデバイスを堅牢化することができる。脆弱性のないデバイスであれば OS やその上で動作するアプリケーションは安全に実行することが可能であり、脆弱性は発見次第速やかに対応するべきである。

脆弱性の対策ができていないデバイスでどのようなことが実現可能となるのか、具体的な例を以下に示す。

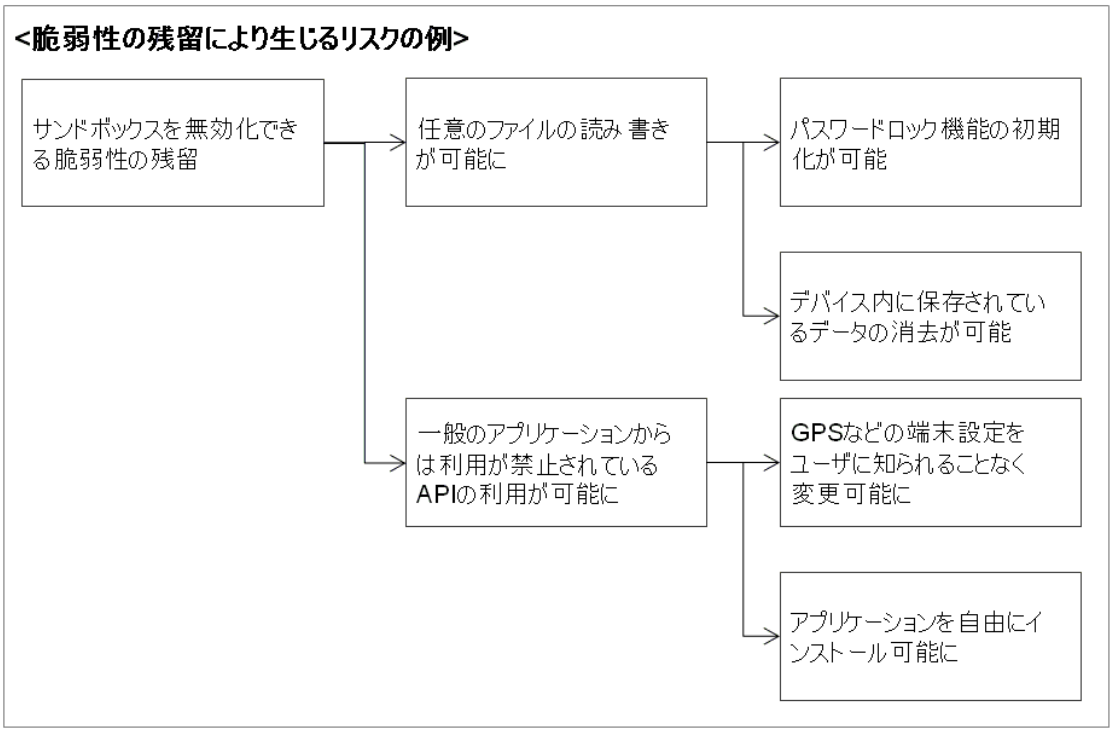

図 2 脆弱性の残留により生じるリスクの例

秘密にしたい情報が秘密ではなくなり、デバイスの動作を変更することができるようになってしまうため脆弱性をつぶすことは重要な対策であると考えられる。

2.2. デバイスの堅牢化のために対策すべきこと

現実問題として脆弱性が全く存在しないデバイスを実現することは極めて困難である。顕在化した脆弱性への対応を迅速に行うことはもちろん、潜在する脆弱性に対しても何らかの配慮が必要となる。ここで考えるべきは、仮にデバイスに脆弱性が存在したとしても、サンドボックスを無効化されてしまわないようにするための対策である。脆弱性を利用してサンドボックスを無効化する際には必ず OS そのもののコード、もしくは、データを書き換えるという手順を踏む必要がある。

では、OS が書き換えられるとするならそれはどこで行われるのか考えてみる。通常、デバイスに格納されているプログラムはデバイスの電源を落としたとしてもその内容を失われることのない不揮発メモリに保存される。不揮発メモリにはデバイスの出荷時から通常書き換えられることのない OS などを格納する ROM 領域と、ユーザの操作によって値を書き換えられる可能性のあるユーザデータ領域とが存在する。これらの不揮発メモリに保存されているプログラムはそのまま実行されることもあるが、一度 RAM に展開してから実行されることが多い。OS に限定して言うならば、書き換えられる可能性があるのは ROM 領域に保存されている OS、もしくは、RAM 上に展開された OS のいずれかであると言える。

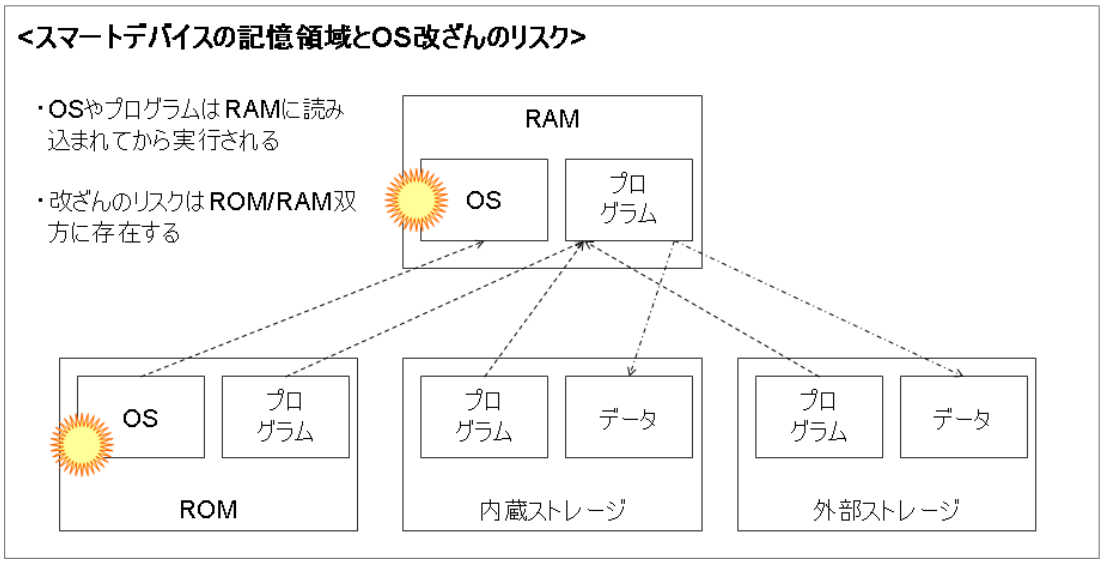

図 3 スマートデバイスの記憶領域と OS 改ざんのリスク

OS が動作する環境として堅牢なデバイスを実現するために考えるべきことは上記の中で説明したが、そもそもデバイスに搭載された OS 以外のものがデバイス上で動作

した場合のことを考える必要がある。デバイスに搭載された OS でいくら保護していたとしても、この OS を使用せずにプログラムが動作してしまえば、ROM 領域内のバイナリを書き換えることが可能になり OS そのものを書き換えてしまうことが可能となる。このため独自バイナリからデバイスが起動できるということは OS の動作を変更してしまう程のリスクを持つということになる。こうした機能はそもそも端末に安易に搭載すべきではない。

このうち最も優先して対策すべきは脆弱性を解消することそのものである。脆弱性が存在しなければ OS の動作を保証することが可能である。次に対応すべきは ROM 領域内の OS の書き換えを禁止することである。ROM 領域内の OS の書き換えができてしまえばサンドボックスを継続して無効化することが可能になる。一方で RAM に展開した OS の書き換えができてしまったとしても毎回脆弱性を利用しなければサンドボックスを無効化することができず、事後的であっても脆弱性を解消すればサンドボックスを機能させることが可能となる。このため RAM に展開した OS の書き換えを禁止する対応は ROM 領域内の OS の書き換えの禁止よりも重要度が低いと言える。

デバイス設計において想定していない方法により、デバイスに搭載された OS とは別の OS を起動することが可能である場合、ROM 領域内の OS の書き換えのリスクが発生する。このため、想定している OS 以外の独自バイナリからの起動を禁止する対応の重要度は、RAM に展開した OS の書き換えよりも高いと考えられる。とはいえ、OS の一部の書き換えではなく、デバイス上で動作する独自バイナリを一部ではなく一式準備するという必要があるため、ROM 領域内の OS の書き換えを禁止するという対策よりは重要度は低いと考えられる。

これらを整理するとデバイス堅牢化のための対策としては以下の順序で重きを置くべきである。

1. 脆弱性を解消する
2. ROM 領域内の OS の書き換えを禁止する
3. 想定している OS 以外の独自バイナリからの起動を禁止する
4. RAM に展開した OS の書き換えを禁止する

3. デバイス堅牢化のための対策

先に挙げた合計 4 点の対策について重要度順に以下に詳しく説明する。

3.1. 脆弱性を解消する

一般的に OS の管理者権限(root 権限と呼ぶ)を利用することで、本来 OS がアプリケーションに利用を認めている範囲の権限を超えてソフトウエアを動作させることができる。通常、管理者権限は特定の処理でしか利用できないように設計されているが、脆弱性を利用することで一般のアプリケーションからも root 権限を利用できるようになる場合が存在する。この脆弱性をデバイスの出荷前に既知のものについては事前に対策を行い、出荷後のデバイスであったとしてもソフトウエアのアップデートを行って悪意のあるアプリケーションが root 権限を利用できないようにするべきである。

脆弱性対策を考える上で注意すべきポイントとしては以下の 4 つが考えられる。

- **最新 OS の搭載**
  最新 OS では既知の脆弱性に対する対策が入っているが古い OS を利用する場合にはその限りではない。古い OS を利用する場合にはセキュリティパッチを独自で反映する必要があるケースが存在する。
- **脆弱性情報の入手**
  CVE(http://cve.mitre.org/)や JVN(http://jvn.jp/)などの脆弱性情報や一般のユーザの動向などからスマートデバイスの脆弱性情報を入手し各デバイスの対応状況を確認することでいち早く脆弱性への対策を行うことが可能となる。
- **セキュリティパッチの反映**
  OS ベンダからのセキュリティパッチのリリースや脆弱性情報とともに配布されるセキュリティパッチのリリースをデバイスの上で動作するソフトウエアに反映させる、もしくは、まだパッチの用意されていない脆弱性に対するセキュリティパッチを自前で用意するなどして、セキュリティパッチをあてたソフトウエアを作成することで、対策ソフトウエアを市場のデバイスに配布することが可能となる。
  また、セキュリティパッチは標準状態の OS に対して適用可能な形でリリースされる事にも注意が必要である。デバイスに対して独自の作り込みを行っている場合には、セキュリティパッチと作り込みの内容が競合する可能性がある。競合が発生した場合には、それを解決するためにセキュリティパッチを変更する必要がある。このように独自の作り込みは時としてセキュリティパッチの反映に時間を要する可能性があるため注意が必要である。

・ **対策ソフトウエアの配布**
脆弱性への対策はそのソフトウエアがユーザの手元で動作する状態になって初めて完了したと言える。そのためには対策ソフトウエアを配布できる状態にし、配布できるようになったことをユーザに告知するところまで対応するべきである。

3.2. ROM 領域内の OS の書き換えを禁止する

脆弱性の利用を抑止するために ROM 領域内の OS の書き換えを禁止する方法が効果的であるということは既に説明済であるが、ROM 領域内の OS の書き換えを禁止しない場合には禁止している場合と比較して、さらなる被害が出る可能性が存在する。これも例を挙げて説明する。

・ ブラウザが利用するルート証明書が書き換えられる
・ デバイスのソフトウエアをアップデートするための仕組みを無効化できる

OS が書き換えられないことを前提にしている仕組みが意味をなさなくなってしまうという意味で ROM 領域内の OS の書き換えは禁止するべきであると考えられる。

3.3. 想定している OS 以外の独自バイナリからの起動を禁止する

想定している OS 以外の独自バイナリからの起動ができたとしても、動作することそのものは問題となるわけではない。あくまで本来デバイス上で動作する OS を書き換えられる可能性があるという観点で配慮すべき項目である。そもそもユーザが用意した独自バイナリの起動を行うためには、ブートローダが独自バイナリからの起動に対応していることが必要となるため、そうした起動を機能として搭載している場合にのみ問題となる。

また、デバイス上で動作する独自バイナリを別途構築する必要があるために、この方法を実現することは、単純に脆弱性を利用したり、ROM 領域内の OS の一部を書き換えたりするよりは困難であると考えられる。

また、ブートローダのアンロックという機能が搭載されているスマートデバイスも存在する。これは OS の起動前に実行されるブートローダ内に存在する機能であり、この機能を利用すれば ROM 領域内の OS を独自バイナリに変更して起動することが可能となるため、一般ユーザから容易に利用されないように保護するべきである。

3.4. RAM に展開した OS の書き換えを禁止する

脆弱性対策が必ずしも完璧にできるわけではないという前提に立った場合に、RAM に展開した OS の書き換えが可能であった時に存在するリスクは、脆弱性対策の次善の策が講じられない以上のリスクが存在するわけではない。また RAM に書き込まれた内容はデバイスを再起動することでリセットすることが可能であるため、ユーザが意図してデバイスの電源を一度 OFF することで、脆弱性を利用されていない状態に RAM を復元することが可能である。このため、OS をアップデートすることでデバイスに存在する脆弱性対策を行い、ユーザにデバイスの電源の OFF を促すことで、脆弱性を利用できない状態にすることが可能である。

このため、RAM に展開した OS の書き換えの禁止を行うことは、デバイスの脆弱性対策を補う用途にのみ貢献できる対策項目であると言える。

4. おわりに

これらの対策をデバイス上で実現するためにはさまざまな手法が考えられるが、どの手法が優れているということを単純に比較することは非常に難しい。本来守るべきものはデバイスの動作そのものであり、個別でいくら強固な対策を講じていたとしても横にその対策をすり抜ける問題が残っていればデバイスそのものが堅牢であると言えるわけではないからである。このためデバイスとして全体でどういった対策が行われているべきであるかというバランスを考えることは重要である。

また、デバイスを堅牢化するためにはソフトウエアの変更のコストであったり、動作速度への影響であったり、対策のない状態よりも多くメモリを消費したりすることがある。セキュリティ対策を行うためにはそうした犠牲が伴うものであり、どのような対策であっても無闇に行うことが必要なのではなく、過剰な対策については行わないという判断も実運用上では必要になってくる。

こうした判断をするための情報として本ガイドが貢献できれば幸いである。