6 Google Chrome OSの構成から見るセキュリティ対策

天野光隆（ミラクル・リナックス(株)）

Chrome OSの概要とセキュリティポイント

Chrome OSとはGoogle社が開発しているオペレーティングシステム（以下OSと記載）であり，Webブラウザを動作させることを主目的にしたネットブック向けのOSである．Chrome OSのオープンソース版が「Chromium OS」であり，2009年11月にソースコードが公開された．開発には，Acer，ASUS，HP，Lenovo，TOSHIBAといったパソコンメーカや，Adobe Systems，Freescale，Qualcomm，Texas Instruments，Intelなどさまざまな分野のベンダが開発に協力している．2010年後半にはChrome OSを搭載した端末が登場する予定であるが，ユーザが自由なハードウェアにインストールすることはできず，ハードウェアと一緒に出荷されることになっている．ただし，公開されている「Chromium OS」のソースコードをコンパイルすれば，Chrome OSを試すことができる．今回，公開されているソースコードと資料をもとに，Chrome OSのセキュリティ機能の調査を行った．

セキュリティ機能を説明する前に，まず，Chrome OSの概要について説明する．

Chrome OSはパッケージ管理に，Gentoo Portageという管理システムを使って，独自のLinuxディストリビューションとして組み上げている．一般的にソフトウェアのパッケージはバイナリで配布されることが多いが，Gentoo Portageは，ソースコードを開発環境上もしくはターゲットとなるマシン上でビルドしてインストールする．この方法を用いると，ソースコードをビルドするためにインストールに時間はかかるが，個別のハードウェア・アーキテクチャに最適化できるというメリットがある．Chrome OSは，1つのソースコードをいろいろなハードウェアに最適に適用したいため，その目的に最適な管理システムとして，このGentoo Portageを採用している．

Chrome OSをビルドすると，図-1に示すようなソフトウェア構成図になる．

Chrome OSは一見ハードウェアと独立しているように思えるが，Google社が提唱する「電源投入後すぐに使える」という仕組みを実現するために，ハードウェアが搭載する「ファームウェア」にも手を加えており，これもChrome OSの一部といえる．さらにこの部分はセキュリティに関する機能も含んでいる．

Chrome OSはLinuxカーネルやライブラリにはオープンソースソフトウェアを採用し，一部分にGoogle社がChrome OS用に独自の実装やカスタマイズを行っている．Fedora，Ubuntu等の一般的なLinuxディストリビューションのカーネルはデバイスドライバやファイルシステムなどをカーネルに含めずになるべくモジュール化された構成に対し，Chrome OSの場合は限定した機器を提供すればよいので，必要最小限のドライバのみを採用し，カーネルイメージに含めている．これにより起動の高速化を表現している．また，ネットワークの接続状態を管理するのは「Connection Manager（connman）」というマネージャプログラムを採用している．connmanはMoblin（現：MeeGo）プロジ

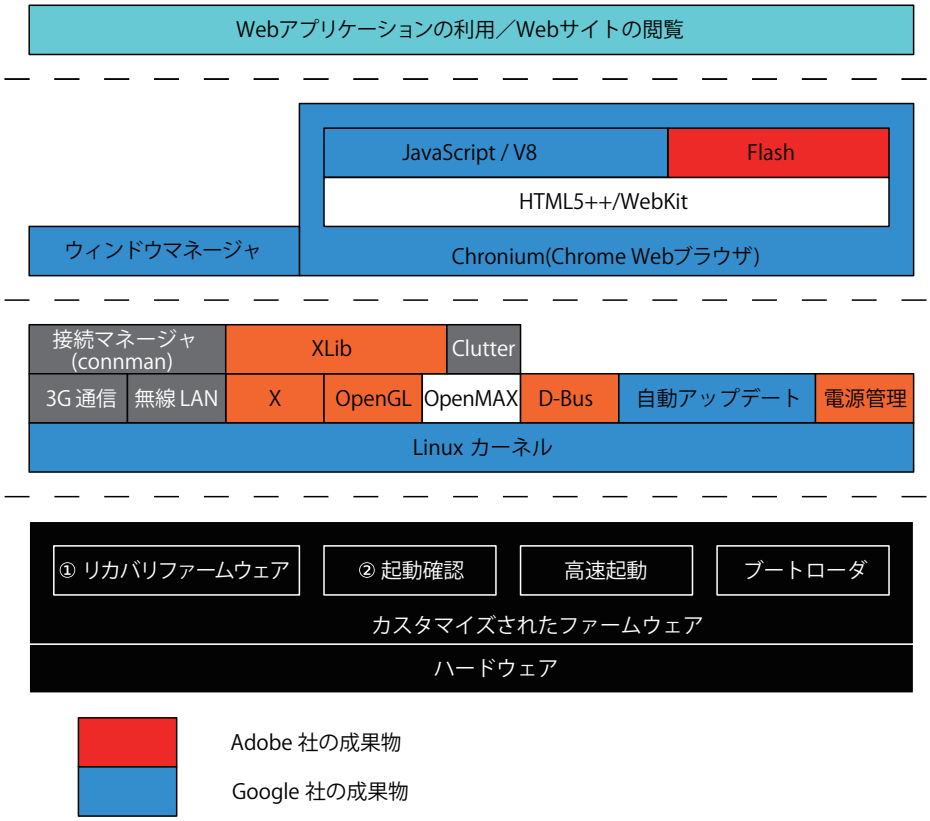

図 -1　Chrome OS のソフトウェア構成図

ェクトで開発された機能で，無線 LAN，Bluetooth，3G データ通信，WiMAX 通信などが管理可能である．これらの機能はすべてプラグイン形式になっており，必要な機能のみをロードするため，プロセスが使用するメモリを最小限に抑えられる特徴を持つ．

アプリケーションは Google 社が開発した Webkit ベースの Web ブラウザ「Chrome」のみが起動可能である．ブラウザ機能しかないが，全画面に表示できるように，独自開発のウィンドウマネージャを使用している．Chrome は V8 という高速な JavaScript エンジンと Adobe Flash Player が利用できる状態で提供される．基本的には，同社の Web アプリケーションをはじめとしたブラウザからの利用を想定している．それ以外のアプリケーションは接続マネージャのフロントエンド，入力メソッド，設定フロントエンドのみが動いている状態で提供される．そのため，アプリケーションはローカルには追加されず，ブラウザを通した Web アプリケーションという形で利用することになる．

この Chrome OS には，いくつかのポイントにセキュリティ機能が適用されている．特に OS 側だけでなくファームウェアにも実装されるなど，よりローレベルへのセキュリティ対策も適用されている．Chrome OS はブラウザ向けシステムであるため，Gumblar 型ウイルスのような「Web ページを閲覧しただけで感染する」攻撃には特に気をつけなければならない．Gumblar 型のようなウイルスに対応するには，ブラウザは他のプロセスへ影響が広がらないようにし，カーネルは呼び出し可能なシステムコールを制限して影響を抑える必要がある．さらに感染した場合にはファームウェアが復元できるような仕組みを持ち，ファイルシステムが改ざんされた場合には起動を抑止することが必要になる．このようにブラウザ，カーネル，ファイルシステム，ファームウェアすべてに対してセキュリティ対策が必要であり，Chrome OS もこれらに対する機能がある．

以下では，Chrome OS 適用システムの実行にかかわる 5 つのレイヤ，システム起動時，Linux のファイルシステム，login 時，アプリケーション実行時，Web ブラウザに関してセキュリティ機能を説明する．

システム起動時における セキュリティ機能

Chrome OSは，前述した通り，通常のLinuxディストリビューションとは異なり，特定の機器にプリインストールされるオペレーティングシステムであり，独自のファームウェアを機器ごとに持つ．このファームウェアにはセキュリティチェックやシステムリカバリの機能が搭載されている．

たとえば，カーネルが悪意のある第三者によって書き換えられていたらその使用を回避できるように，ファームウェアが起動前に書き換えを検査する．もし，Chrome OSが書き換えられていた場合，あらかじめ保存しておいた正常時のOSイメージを使って，ファームウェアのレベルでその正常時のOSに戻って起動することができる．図-1に描かれている「①リカバリファームウェア」「②起動確認」がこの処理を行う．次にこれらの各機能を説明する[1)]．

なお，通常のPCにChrome OSをインストールしてもこうした機能が有効になるわけではない．プリインストール対象のハードウェアにChrome OS用のファームウェアが含まれていることが条件となり，これはユーザ自身がインストールできるものではない．

Chrome OSに搭載されるファームウェアの構成を図-2に示す．2MBのEEPROM（電気的にデータを消去可能なメモリ）の内部は書き込み保護された領域と書き込み可能な領域の2つに分かれており，システム回復用のイメージは書き込み保護されたエリアにある．もしダメージを受けてもファームウェアからシステム回復機能を使ってリカバリできる．このイメージが書き込まれている書き込み保護のエリアはメーカが出荷されるときにすでに入っている．システム回復はユーザが任意で起動可能であるが，ブートチェック時に問題があった場合には自動起動する仕組みになっている．

システムを起動すると，起動チェックが動作する．システム起動時にファームウェア内の「ブートスタブ」に格納されているGoogle社提供の公開鍵を使って，これから起動するファームウェアAが，対応

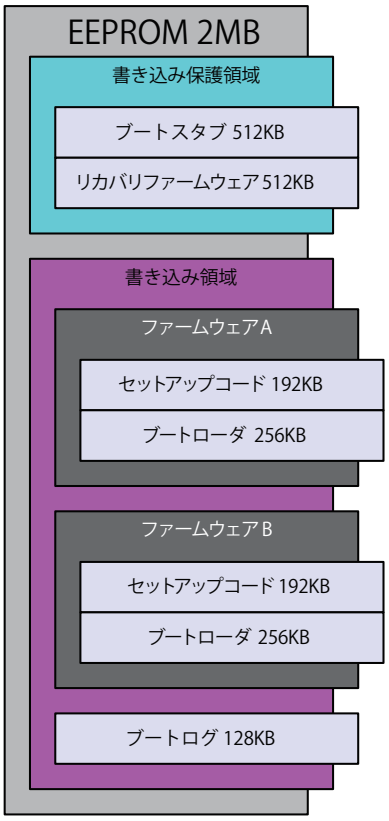

図-2　EEPROMの構成図

する秘密鍵で署名されているファームウェアであることを確認する．正当であると判断されれば次の処理へ進む．図-2でも分かるように，公開鍵は読み出し専用ファームウェア内のブートスタブ部分にあるため，ユーザによって更新されることはない．すなわち悪意ある書き換えが行われることがない．Chrome OSでは，公開鍵は各ハードウェアメーカがハードウェアに組み込み，ファームウェアはペアとなる秘密鍵を使って署名を暗号化してから出荷する．この暗号化形式は明確にされていないが，TPM（Trusted Platform Module）チップを経由した確認ができるようにする方法も考えられているため，RSA暗号演算などが用いられると推測される．

次に起動チェックであるが，大きく分けてファームウェアとカーネルの2段階の検査を行う．

まずファームウェアでは，ブートスタブに含まれている公開鍵を使い，書き込み可能なファームウェアの署名が対応する秘密鍵で署名されているかどうかをチェックする．正規のファームウェアと判断されたら，ファームウェアの処理へ移行する．正規と

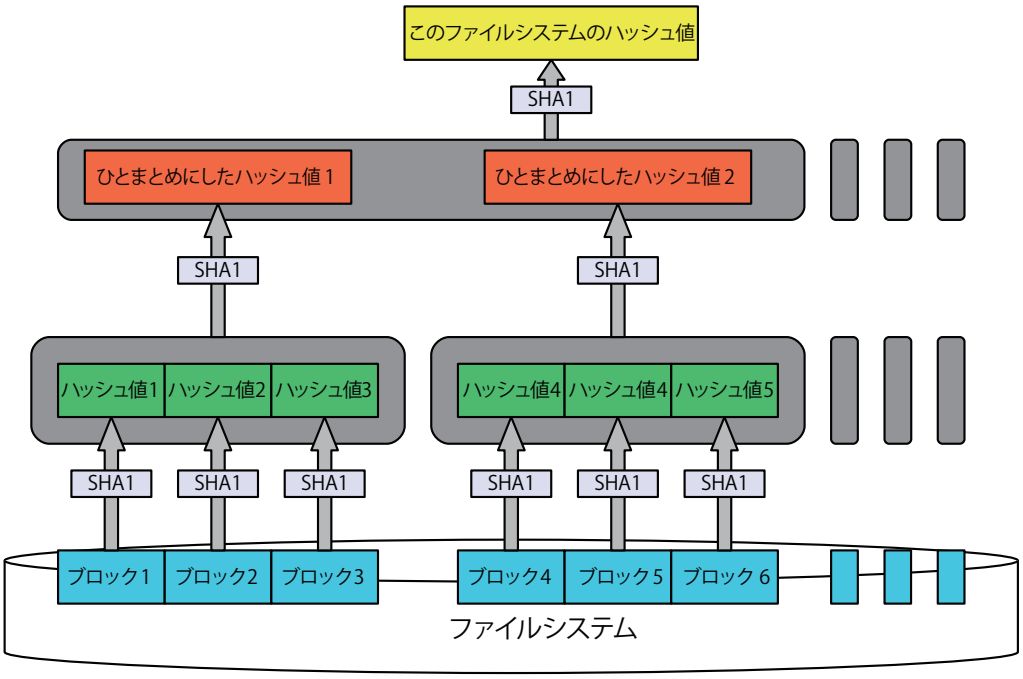

図 -3　ファイルシステムのハッシュ値の導き方

判断されなかった場合は，2 つ目のファームウェア B のチェックを行う．2 つのファームウェアの扱いについては，公開されている資料には明確に定義されていないが，ファームウェア B はバックアップ用としてファームウェア A からコピーされ，通常利用されるファームウェア A が改ざんされた場合に使うために組み込まれていると推測される．ファームウェア A，B ともに正規ではないと判断された場合は，リカバリファームウェアが起動し，外部ストレージより正規のファームウェア（A または B）と OS イメージをコピーする形で修復を行う．これにより EEPROM の書き込み領域上にあるファームウェアが置き換えられることになる．新しいファームウェアにも署名されているが，当然ブートスタブにある公開鍵で復号化できなければならないため，別の秘密鍵で署名されたファームウェアをコピーしても，復号化できないのでそこから起動することはできない．さらに，ファームウェアでは，ブートローダとカーネルイメージ，起動デバイス上のパーティション，ファイルシステムのチェックを行う．

ファームウェア側のファイルシステムのチェックは，ファイルシステムハッシュ値と対となるファームウェアが持つハッシュ値と照らし合わせる．一致すれば正しいファイルシステムと判断する．ファイルシステムのハッシュ値は図 -3 の通りで，ブロック（4KB）ごとのデータを SHA1 の暗号方式を使ってハッシュ値を生成し，ある程度の数（図では 3 つずつ）にまとめて，さらにハッシュ値を生成する．最終的にひとまとまりになったハッシュ値がファイルシステムのハッシュ値となる．つまり 1 つでもファイルシステムのブロックが変更されたら，それは信頼できるハッシュ値とは認められない．論理的にはファイルシステムのすべてのブロックを検査して完全に正しいと言えるようにするべきであるが，4KB の SHA-1 ハッシュ値の計算速度は 0.2ms（x86 アーキテクチャ）から 0.5ms（ARM アーキテクチャ）程度かかり，チェックのために数秒かかることになる．これは高速起動におけるボトルネックとなるため，Chrome OS では約 19,200 個の SHA-1 ハッシュ値だけ計算し，先頭から約 75MB までのチェックでファイルシステムのハッシュ値としている．

チェックの結果，正しくないと判断された場合，ユーザはチェック結果を無視する方法と，リカバリモード（再起動後にブートスタブからリカバリファームウェアを呼び出す）で起動する方法のいずれかを選択する．

一方，カーネル側のチェックは，ファームウェア側のチェックとは独立して実行される．カーネルの

チェックはまだ具体的には実装されていないが，以下の 2 つの方法が提案されている．

1. ファームウェア上にあるカーネル用の公開鍵を使ってカーネルが対応する秘密鍵で署名されたカーネルであるかどうかを判断する．
2. 読み書き可能なファームウェアが検証用に使ったものと同じ公開鍵を使用する．

1. の場合はファームウェア用の鍵とは別にカーネル用の鍵を用意する必要があるが，よりセキュアである．2. の場合は 1 種類の公開鍵があればよいが，1. と比べるとセキュリティは低い．Chrome OS 標準として 1 つの方法に固めるか，Chrome OS のデバイスを提供するメーカによって変えられるように自由度をつけるかは未定である．

以上の 2 つのチェック機能により，システム起動時において，ファイルが改ざんされているかを知ることができ，早めの対策を打つことが可能となる．また，これらの公開鍵は Google 社が生成して提供することになる．

ファイルシステム構成（ストレージ）のセキュリティ

Chrome OS では，Linux カーネルが持つセキュリティ機能，たとえば SELinux や TOMOYO Linux は利用していない．前述ではファイルシステムのチェックについて述べたが，ここではファイルシステムの構成から見るセキュリティ対策について紹介する．

Chrome OS 適用機器は，特定の専用機器となるため，不要な書き込みをさせないことが重要である．このセキュリティ対策のために，ファイルごとに配置される領域が異なっている．扱われるファイルを 3 つの種類に分けて説明する．

1. 実行バイナリ，共有ライブラリ等

Chrome OS の変更する必要がないファイルとして実行バイナリや共有ライブラリがあるが，これらは読み込み専用の領域に配置する．これにより容易にユーザが実行プログラムの動作を変更させないようにすることができる．

2. 実行プログラムが作る一時ファイル，キャッシュ，ログファイル等

キャッシュや一時ファイルなどは書き込み可能な領域に保存するが，ディスク上には書き込まず，RAM 上に保存する．そのため電源が落ちるとこれらのファイルはすべて消去される．

3. ブラウザで保存された Cookie，認証情報，ページキャッシュなどのユーザ固有ファイル等

ユーザ固有のファイルは書き込み可能な領域に保存し，ディスク上に書き込む．ext3 などのファイルシステム上にそのまま書き込むと，簡単にデータを読まれてしまうためセキュアではない．そのため AES128 アルゴリズムによる暗号化を適用したファイルシステムイメージを作成し，これをマウントした先へ保存している．起動中（ファイルの使用中）はそのまま参照されるが，停止中はユーザデータは暗号化されたファイルシステムイメージしか見えないため，容易に解析することができないようになっている．

この仕組みはパーティションごとに書き込み属性を変えてマウントすることで実現している．具体的には，1 つのハードディスクを 3 つのパーティションに分割している（図 -4）．

1 番目のパーティションにはユーザデータが保存される．このパーティションのことを Chrome OS では「Stateful Partition」と呼んでいる．マウントポイントは 4 個所（/home, /var/log, /var/cache, /mnt/stateful_partition）あるが，すべて同じ 1 番目のパーティションにマウントされている．2 番目のパーティションは EFI フォーマットで作成されており，BIOS レベルでの高速起動「Boot Booster」を行うために必要なデータが入っている．3 番目のパーティションには実行ファイルやライブラリなどが含まれている Chrome OS のルートファイルシステムが置かれ，ext3 フォーマットのファイルシステムを「read only」属性でマウントする．

システムのログファイルやプロセスのロックファイルといった，キャッシュや一時ファイルは tmpfs（RAM ディスク）上にマウントする．RAM ディスクに保存さ

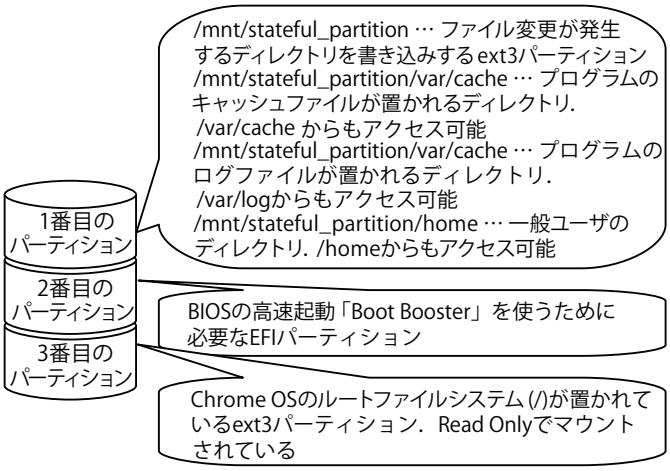

図 -4　Chrome OS のパーティション構成

れたファイルは電源が落ちるとすべて消去される.

逆に，ネットワーク設定などは，利便性を考えて Stateful Partition に保存し，電源を投入して再度使うときでもすぐに接続することができるようにしている．ドキュメントファイルといったユーザデータは，/home/.shadow ディレクトリ以下のイメージファイルに格納される.

ここで行われているセキュリティは，他のユーザによるアクセスを回避するための制御である．イメージファイルは Linux kernel の機能である dm-crypt を使って暗号化され，/dev/mapper/crypthome というデバイスを使い，ログイン時にマウントされる．暗号化 は SHA 256bit のハッシュ関数を使い AES 共通鍵暗号方式を使用する．この共通鍵は，ログインした Google アカウントごとに作成されるため，他のユーザとの棲み分けを実現できる.

login 認証について

Chrome OS では独自のアップデート機能がある．Fedora や Ubuntu などの Linux ディストリビューションで使用されているような yum, apt などは使用していない．緊急のセキュリティ更新があると，ユーザがシステムにログインした後，即時に自動更新され強制的に再起動が行われる．また，コンソールの切り替えや出力するメッセージを最低限に絞ることで第三者が解析をして取得できる情報を少なくし，成りすましができないようにしている.

Chrome OS のログインは，ユーザが Google アカウントのユーザ名（Gmail メールアドレス）とパスワードを入力すると，2 種類のログイン処理を行う．1 つ目は Google アカウントでのログイン，2 つ目は Chrome OS の一般ユーザとしてのログインである．Google アカウントのログイン処理はネットワーク通信が必須であるが，ログインキャッシュが実際のディスクに書き込まれるため，2 回目以降はオフラインでもログイン可能となっている．ただし，一部のキャッシュは電源切断時に削除されるため，最低限の情報のみ持たせることで，悪意のあるユーザによって解析される範囲を少なくしている．Chrome OS の一般ユーザログインは，Web ブラウザや入力メソッドなどログイン後に使うプログラムを一般ユーザとして起動する．これは管理者権限が必要なコマンドの実行を禁止するために行っている．オープンソースプロジェクトとして公開されている Chromium OS のソースコードでは，デフォルトで「chronos」というユーザ名を作成しており，これが一般ユーザとして利用される．このユーザ名はバイナリプログラム内に固定化されているが，ビルド時に変更することが可能である [2].

なお，ログイン画面ではネットワーク接続設定，言語設定が可能となっており，ネットワーク接続が完了してからログインを行う手順になる．ログイン画面ではネットワーク接続のアプリケーションが起動されており，有線 LAN または無線 LAN による接続が可能である．接続されたことをアイコン上で確認した後に，実際にログイン情報を入力して処理することになる.

具体的な処理の流れを説明する． Chrome OS のログイン処理は図 -5 のような流れになっている.

ログインマネージャには Google 社が独自で実装した「SLIM（Simple LogIn Manager）」を採用している．Chrome OS を起動し，ログイン画面が表示されたらユーザは「Google アカウント」を入力する．Google アカウントを持ってない場合，作成ボタン

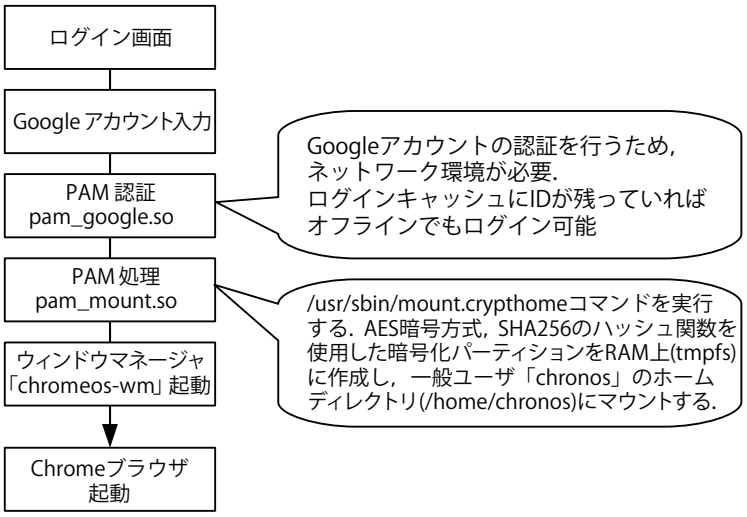

図 -5　Chrome OS のログインプロセス

が用意されているのでそこからアカウントの作成とログインを行うことになる．アカウントの作成は Google API が呼び出されてサービス上で作成を行うため，ネットワーク接続が必須となる．入力された情報は pam_google.so という PAM（Programmable Authentication Module）モジュールに渡される．pam_google.so は Google の ClientLogin API[☆1] を呼び出し，それが Google アカウントとして存在するかどうかの確認を行う．このアカウントは Gmail や Google Calendar などを使う際のアカウントである．Google 側の認証情報とサービスは一対多であるため，1 つ以上のサービスを利用していれば Google アカウントが入手できる．最低限 Gmail を使っていないとログインできないなどといった制約はない．アカウントが存在し，ユーザ，パスワードが正しければ，続いて pam_mount.so の処理へ移る．pam_mount.so は前述のとおりユーザデータのマウント（/dev/mapper/crypthome）を行い，最終的に一般ユーザ「chronos」のホームディレクトリ「/home/chronos」からアクセスできるようにする．そして，マウント後に一般ユーザとしてログインが行われる．pam_mount.so の処理が終了すると認証手続きがすべて完了し，この後のウィンドウマネージャ「chromeos-wm」や「Chrome ブラウザ」は chronos ユーザによって起動される．

ログイン情報はプレーンテキストで書かれたファイルをユーザデータのマウントで使用した AES 共通鍵暗号方式の鍵を使って，保存する．これにより，次回からのログインは，ネットワーク経由による Google アカウントの認証か，もしくはディスク上に保存されたログイン情報と照らし合わせた Google アカウントの認証を行うことになる．オフラインによる認証で再起動する場合は pam_google.so が行い，サスペンド，ハイバネーションから復帰時は pam_offline.so という PAM モジュールが行う．pam_offline.so はスクリーンセーバーのロックにおける認証で使われるが，認証手順は pam_google.so と同様である．すなわち pam_offline.so は pam_google.so のオフライン処理のみを持つモジュールである．ログイン情報はディスク上に保存されることになるが，暗号化されているため，ディスクを盗まれて解析された場合でも，容易に解析できないようになっている．

アプリケーションのセキュリティ（SECCOMP sandbox）について

Linux のファイルシステムは，基本的に Read Only でマウントされており，ログやキャッシュは RAM 上（tmpfs）に書き込まれる．そのためシャットダウンすると，キャッシュファイルは消えて，起動前の状態となる．また，カーネルに標準で実装されている seccomp 機能が有効になっており，特定のプロセスには基本的なシステムコールしか許可しないようにしている．Chrome OS はこれに修正を加え，ブラウザなど一部のアプリケーション・プロセスに対して色々な処理の実行を有効にしている．以下，具体的に述べる．

Chrome OS のカーネルは，セキュリティ機能の「seccomp」に独自のパッチを加えている．seccomp はカーネルに標準に実装されている，プロセスの行

☆1 http://code.google.com/intl/ja/apis/accounts/docs/AuthForInstalledApps.html

動を制限する仕組みである．一般にプロセスは，デバイスへのアクセス，および外部に対して何らかの要求をする場合，システムコールを用いてカーネルを呼び出す．Linux には 300 種類以上のシステムコールが用意されているが，カーネルの seccomp 機能を有効（CONFIG_SECCOMP=y）にすると，特定のプロセスには「read()」「write()」「exit()」「sigreturn( )」という基本的な 4 種類のシステムコールしか許可しない．これによって，特定のプロセスが脆弱性を持ち，攻撃されたときにほかに悪影響を及ぼすのを防ぐことができる．この設定があるプロセスからほかのシステムコールが発行されても，それらは無視される．この seccomp 機能自体は，Ubuntu などの Linux ディストリビューションでも有効化しているようだが，具体的にどの辺で利用されているかは明らかになっていない．

seccomp はセキュリティ面では強力であるが 4 種類のシステムコールだけでは処理できないプログラムもある．4 種類以外のシステムコールを使う必要があるプログラムもあるため，seccomp をそのまま適用すると不都合が生じる．たとえば Chrome ブラウザはプログラム内部のメモリ操作で mmap() を使用するが，seccomp 上で実行するとメモリの確保をできずに予測しないエラーが起こる可能性がある．そこで Google 社は，Chrome OS 向けに seccomp の機能を拡張するパッチを作成し，seccomp を有効にした環境下でもプロセスに特定のシステムコールの発行を許可する「seccomp モード 2」を追加している．このパッチは，発行できるシステムコールの種類を制限するのではなく，システムコールを発行できるメモリ領域に制限をかけており，システムコールに対しビットマスクを与えて，特定の命令のみを可能にする．たとえば write() が書き込み可能なファイルディスクリプタを 2（標準入力）のみにするなどができる．これはバッファオーバフローなどによって書き込まれた不正なプログラムの動作を制限するときに有効な方法である．

ただし，現時点では Chrome ブラウザをはじめとしたアプリケーションでは，このような seccomp sandbox に関連する実装はされていない．まだ開発段階のため，今後，仕様が変更される可能がある．

Web ブラウザのセキュリティ（Sandbox 型セキュリティによるリソースの隔離）について

Chrome OS のブラウザである Chrome そのものにもセキュリティ対策が行われている．これは Chrome OS のみではなく，今日利用されている Windows, Linux, Mac OS X 用の Chrome でも標準で適用されている．

Chrome ブラウザはタブ形式で 1 つのウィンドウに複数の Web ページを表示することができる．ブラウザのプロセス管理において，同じ機能を持つ Firefox では 1 プロセスですべてのタブを管理している．対して，Chrome では 1 つのタブ／1 プロセスで管理している．これによってメモリリークなどの影響を最小限に抑えることが可能になっている．さらにこのプロセス分割が Chrome のセキュリティ機能として役立てられている．各タブを制御する管理プロセス（親プロセス）と各タブ（子プロセス）との間では IPC（Inter-Processor Communication，プロセス間通信）を通して制御されている．Chrome は各タブ（子プロセス）を Sandbox 内で起動させて，以下のような制限を実現している．この機能は Chrome ブラウザそのものの機能であるため，前述したカーネルの seccomp sandbox とは別である．

- 利用範囲が制限されたトークン（例：Windows の場合，CreateRestrictedToken 関数）を使うことで，各種カーネルオブジェクト，クリップボードやファイルへのアクセスを制限する
- ウィンドウアクセスやウィンドウメッセージの送受信を制限する

これにより疑わしい動作を行うプログラムからのアクセスを抑止し，ブラウザプログラムそのものの保護をする．たとえば脆弱性のある入力メソッドからの入力の抑止や，SQL インジェクションなどの文字列を破棄するといった応用が可能となる．

この布石とも言える動きが2010年3月30日に起きている．Google社はChromeブラウザにAdobe Systems社の「Flash Player」を統合した．これはFlashコンテンツを含むWebページをChromeのSandboxに対応させることで，さらにユーザを脆弱性から保護できるようにするためのアプローチである．

Sandboxによるアクセス制限は，OSが持つアクセス制御をそのまま使用しているので，使う環境によって制限対象が異なっている．たとえばWindowsでは，Chrome OSと違い，キーボードやマウスなどのリソースをアクセス制限の対象としていない．

ケーススタディ

Chrome OSのファームウェアからカーネル，アプリケーションまでセキュリティ対策を説明してきたが，実際の攻撃に対しどのように使われるのか．いくつかのケースをもとに説明する．

- ケース1：攻撃者がUSBメモリ／SDカードからシステムを起動し，Chrome OS内にあるファイルの変更を行った．
  【Chrome OSの対策】ファイルシステムのブロックが変更されたため，起動時の検査で改ざんが検出される．また，ログイン時に使用されるユーザデータは暗号化されているため，容易に解析できない．ただし，このままでは起動できないので，システムの復元が必要になる．
- ケース2：攻撃者はChrome OSのストレージをファイルシステムごと抜き取り，別のマシンで起動しようとする．
  【Chrome OSの対策】ユーザデータは暗号化されるため，簡単には解析されない．また，別のマシンではこのファイルシステムを起動するためのファームウェアがインストールされていないため，起動することはできない．
- ケース3：攻撃者がChromeおよびChromeのプラグインを別なアプリケーションに置き換えようとする．
  【Chrome OSの対策】アプリケーションの変更はスーパーユーザの権限が必要になるので，権限が奪取されない限りは変更できない．
- ケース4：攻撃者は自分で作成したChrome OSで起動し，ログインする．ログイン時に作成されたユーザデータが保存されている暗号化ファイルを攻撃対象のマシンにコピーして起動する．
  【Chrome OSの対策】SHA256bitのハッシュ値を生成するシードはマシンによって異なるので，そのままコピーしてもハッシュ値が一致しないため，ログインできない．
- ケース5：攻撃者はChrome OSにインストールされている入力メソッドでブラウザ上に入力する機構に脆弱性があり，そこを突いて管理者権限を取得する．
  【Chrome OSの対策】Chromeブラウザの Sandboxによって疑わしいトークンを検知すれば入力ができない．ただし入力メソッドはすでに正常に動作していない可能性があるので，最終的にはアップデートが必要である．

まとめ

Chrome OSで実現されている，もしくは実現を予定しているセキュリティ強化のための機能を紹介した．既存の方法を適用するだけでなく，Chrome OSが専用端末で使うことを利用し，専用用途以外の機能を削除するなどの独自の方法も開発して，セキュリティを強化している．必要としない機能を削ることこそがセキュリティ強化につながるという考えを，Chrome OSのセキュリティコンセプトの一端から見て取ることができる．

参考文献
1) Chromium Projects Security Overview，http://www.chromium.org/chromium-os/chromiumos-design-docs/security-overview
2) Getting the Chromium OS Source Code（ソースコードの入手先），http://dev.chromium.org/chromium-os/building-chromium-os/getting-the-chromium-os-source-code

（平成22年6月15日受付）

■天野光隆　mamano@miraclelinux.com
ミラクル・リナックスにて組込み機器向けLinuxOSの開発，移植を行っている．Moblinコミュニティに創設時から参加し，メンテナとして貢献．現在はMeeGoプロジェクトや技術雑誌への寄稿を中心に活動している．