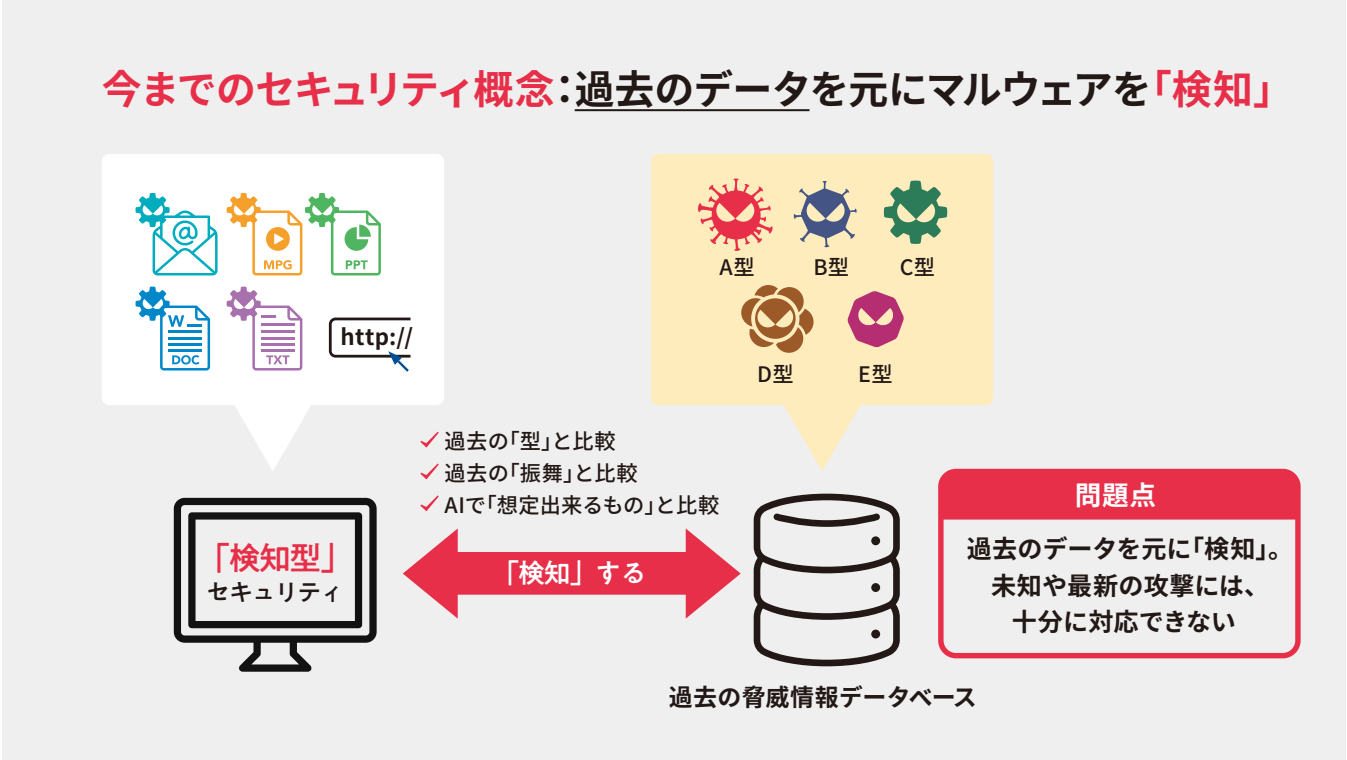
APPGUARD

Blue Planet-works
Safety for the Connected World

10分でわかるAppGuard：仕組み、セキュリティの「新概念」

従来のセキュリティ技術「検知型」の限界：終わりが見えない「いたちごっこ」

今ではAI機械学習を始め、次世代型エンドポイントと言った新しい名のセキュリティ製品が様々なメーカーから提供されています。次世代型、AI、振舞検知、サンドボックス等の技術で表されていますが、実際の所、従来のアンチウイルスとコンセプトは同じ延長線上にあり、マルウェアを「検知」する手法が改善されているにすぎません。そもそもの概念がマルウェアを「検知」する事が主目的になっているセキュリティ製品は限界に達しており、新種や未知のマルウェアに対する防御力が決して高くない実情があります。悪意のある攻撃者からシステムを守るには、新たな技術革新が必要であり、従来のセキュリティ技術概念を根本的に覆すコンセプトの必要性が急務です。従来の「検知型」セキュリティ製品の場合、攻撃から100%システムを防御する事は理論上不可能な為、マルウェアの侵入を前提にした提案をセキュリティベンダーは行います。構成としては多層防御や実際にマルウェアに感染した後のインシデント対応、例えばEDR (Endpoint Detection and Response：エンドポイントでの検知と対応)やインシデント・レスポンス・サービス（セキュリティ事故が発生した際に、事故の波及範囲の把握、侵入経路の解析、攻撃の遮断などの緊急対応を実施するサービス）が重要視されています。



AI機械学習、振舞検知、サンドボックス、アンチウイルス等の従来の「検知型」セキュリティ製品は、マルウェアを「検知」する事が主目的になっています。「検知型」の技術が効果を表すには、先ずは脅威情報データの精度（採取されたマルウェアデータの量）が重要であり、セキュリティ企業の多くは自社の脅威情報のデータの豊富さを売りにしています。脅威情報データベースは、各セキュリティベンダー製品（エンドポイント、ネットワーク、メールセキュリティ製品等）が収集するマルウェアデータ、各社が世界中に張り巡らすハニーポットに集まった脅威データ、Virus Total (https://www.virustotal.com/)等のコミュニティーに投稿されたデータにより構成されています。これらの脅威情報データベースを基に、マル

ウェアを「検知」するのが現在のセキュリティ技術の仕組みです。アンチウイルス製品は、脅威情報データベースに含まれる過去のマルウェアのハッシュ値情報などを基に比較対照を行いマルウェアを検知します。この脅威情報データベースに含まれる情報は、言うなれば過去のマルウェアの指紋、DNA、筆跡情報のようなものです。このデータをシグネチャ（英語で言う、自筆の“サイン”（筆跡は個々ユニーク）を意味しています）、又は日本語では定義ファイルと呼びます。AI機械学習は、この仕組みを応用したもので、過去の脅威情報を学習させ、数学的モデルを作り、新種のマルウェアを予測する仕組みです。今まで学習したものと全く異なる攻撃をするマルウェアであれば、数学的モデルが上手く当てはまらず、AI機械学習でも予測する事が難しくなります。このように、AI機械学習もアンチウイルスと同じように、脅威情報データベースを基にウイルスを「検知」する仕組みです。

サイバーセキュリティの課題：増大するマルウェア数と攻撃者の組織化

「イタチごっこ」

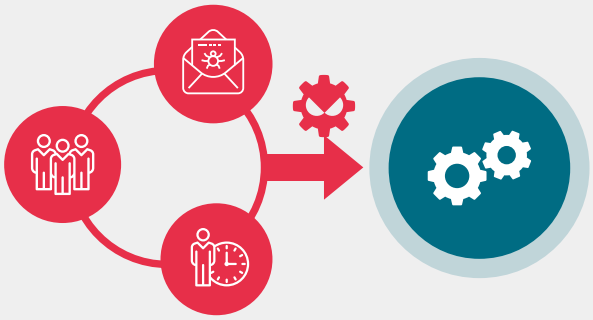
2009年　作成された新種のマルウェア数

236万個

2015年　作成された新種のマルウェア数

4億3,000万個

（一日当たり117万個）



Source:Symantec Internet Security Threat Report Volume 21, 2016

組織化された専門集団が一日中攻撃を仕掛ける

世の中が静的な環境であれば「検知型」も有効ですが、残念ながらマルウェアの脅威は日々増しています。今では、一日当たり100万個以上の新しいマルウェアが生成されていると言われており、これは一秒あたり11.6個の新しいマルウェアの出現を意味します。これらの新しいマルウェアを「検知」するには、脅威情報データの頻繁なアップデートが必須になります。アンチウイルスのように静的な脅威データを基にした比較対照では、新しいマルウェアを「検知」出来ない可能性が高く、AI機械学習の場合でも、過去の数学的モデルと合致するものであれば検知しますが、そうでないものはすり抜けます。また、より多くのマルウェアを検知するように設定すると、必要なファイルまで有害であると誤検知し、削除したり隔離したりします。つまり、「検知型」の場合は、脅威データを基準にした技術概念である為、一日あたり100万個もの新しいマルウェアが生成される世の中では、脅威情報データの更新が追い付かず、「いたちごっこ」状態になります。脅威情報データの更新が追付かなければ、その分、新種のマルウエアからの攻撃を受けて侵入されるリスクは高くなります。また、これらの脅威情報データベースは、大抵PC内に保存されています。従来の「検知型」のセキュリティソフトをお使いの方は、定期的にPCの定義ファイルのアップデートが走り、新しいデータをダウンロードしている際、PCの動作が急に遅くなったり、他のアプリケーションが動作しなくなるような場面を経験された事があると思います。AI機械学習の場合も、AIエンジンのアップデートが定期的に必要になります。AIエンジンは学習を続けると頭が良くなり、予測する数学モデルの精度が上がりますが、PCにインストールされているAIエンジンは学習を続けていないので、日々検知能力が低下します。その為、最新のマルウェアへの対応を考えると、より精度の高くなった（より賢くなった）新たなAIエンジンを定期的にダウンロードする必要があります。定義ファイルの更新やAIエンジンのダウンロードが定期的に行われるという事は、ユーザの業務にも影響がでますし、運用面でも更新やダウンロードの管理、例えば更新ファイルの事前検証、と言ったコストアップにつながる課題が出てきます。また、例え定期的にアップデートを行っても、一日あたり100万個を超える新しいマルウェアの生成スピードに追付く事は実際不可能であり、「いたちごっこ」の現実とセキュリティリスクは常に実在します。今までのセキュリティ技術は、限界に達しており、企業を始め多くのユーザは、高いセキュリティリスクにさらされています。

「いたちごっこ」を打破する別のアプローチ：
システムに変更を加えさせないハードニング技術

前述にもありますように、今までの「検知」ベースのセキュリティ技術は限界に来ており、「いたちごっこ」の終わりは見えません。ここで重要なのは、セキュリティソフトを使うそもそもの目的です。「検知型」のセキュリティ製品の主目的は、マルウェアを「検知」する（見つける）事です。新しいマルウェアを検知すると、そのマルウェアの検体がセキュリティベンダーに送信され、セキュリティベンダーが検体を解析したのち、脅威情報データベースが更新されます。その更新された脅威情報データベースが、定期的にユーザのPCに配信される仕組みです。そして何より「検知型」の場合は、脅威情報データ（つまり過去の情報）をベースにマルウェアを「検知」する為、つい先ほど現れた新種や未知のマルウェアを「検知」する事が難しい現実があります。このようにマルウェアを見つけることが主目的の「検知型」のセキュリティ製品の仕組みは限界にきています。本来であれば、悪意のある攻撃者からシステムを守る事が主目的であり、「システムが常に正しく動作するようにする」、システムの安全性を確保する事が最優先とされるべきです。これを実現するには、「システムに害を与える行為を未然に阻止する」必要があります。この仕組みであれば、マルウエアが未知・既知であろうと関係が無く、システムに対して「悪さ」をしようとする行為すべてを未然に遮断し、システムの安全性を確保します。しかしここで難しいのは、どのような処理を「システムに害を与える行為」と定義するのか、また動的でダイナミックなアプリケーション環境下で様々なプロセスが派生して動作する中、「システムに害を与える行為」をどのように動的に定義するのか、が課題になります。

1990年代、コンピューターシステムを電磁波による誤作動から守る技術として、「コンピューター・ハードニング」という概念が生まれました。ここでの概念は、あらゆる過酷な環境下においてもコンピューターが誤作動せずに、ある一定で特定の動作しか行わないようにします。インターネットの時代を迎え、サイバーセキュリティが問題視されると、この概念は「アプリケーション・ハードニング」に進化し、アプリケーションが特定の動作しか出来ないようにして、例えサイバー攻撃を受けても、本来の動作しか出来ないようにする仕組みです。これにより、システムの安全性を確保します。一見、アプリケーション・ハードニングはサイバー攻撃からシステムを守る、とても有効な手段に見えますが、実システムにおいては大きな問題があります。それは、どの様な動作処理を可能にし、何を不可にするかの定義（ポリシー）をきめ細かく決めなければならない事です。特定用途のコンピューターシステムや限られたアプリケーションが同じ動作を繰り返すような静的に近いシステムの場合は、アプリケーション・ハードニングによるポリシー設定は時間がかかりますが可能です。その後、アプリケーションの変更、バージョンアップ等の変化が無いのであれば、運用面でも大きな問題になりません。しかしながら、システムの多くが様々な業務に使われ、プロセスもそれぞれ異なるダイナミックな動的環境で動作します。また、実際にシステムプロセスが動的に起動している中、不可行為をどのように定義するのかが難しい問題です。このような理由から、アプリケーション・ハードニングは、アプリケーションの変更、バージョンアップ、パッチの適用時にポリシーを修正・編集する必要があり、その度に動作検証テストを行うなどの大変な作業が必要になるため、運用面で大きな問題があります。そういう意味では、アプリケーション・ハードニングは、汎用的なシステムにおいては、実用的な技術ではありません。ただし、仮に正しいシステムのあらゆる動作を指定する事が可能であれば、未知やゼロデイであろうと最新のサイバー攻撃からシステムを守る事ができます。

「新概念」のセキュリティ技術：APPGUARD

従来のセキュリティ技術：「マルウェアの検知」

今までの脅威情報（過去のデータ）を基に「検知」するので、未知や新種の攻撃から完全に守れない

頻繁なデータベースの更新、AIエンジンのアップデートが必要
業務に影響し、運用の管理が複雑化しコストアップ

新概念：「システムの安全を守る」

AppGuard Isolation技術（特許取得済み）

未知や新種に関わらず、システムに害を与える処理を未然に阻止 システムの「信頼」を維持

アップデート不要、運用管理の簡素化、軽量・軽快動作

AppGuardのIsolation技術：
ダイナミックな動的環境下でシステムの安全性を確保する「新概念」

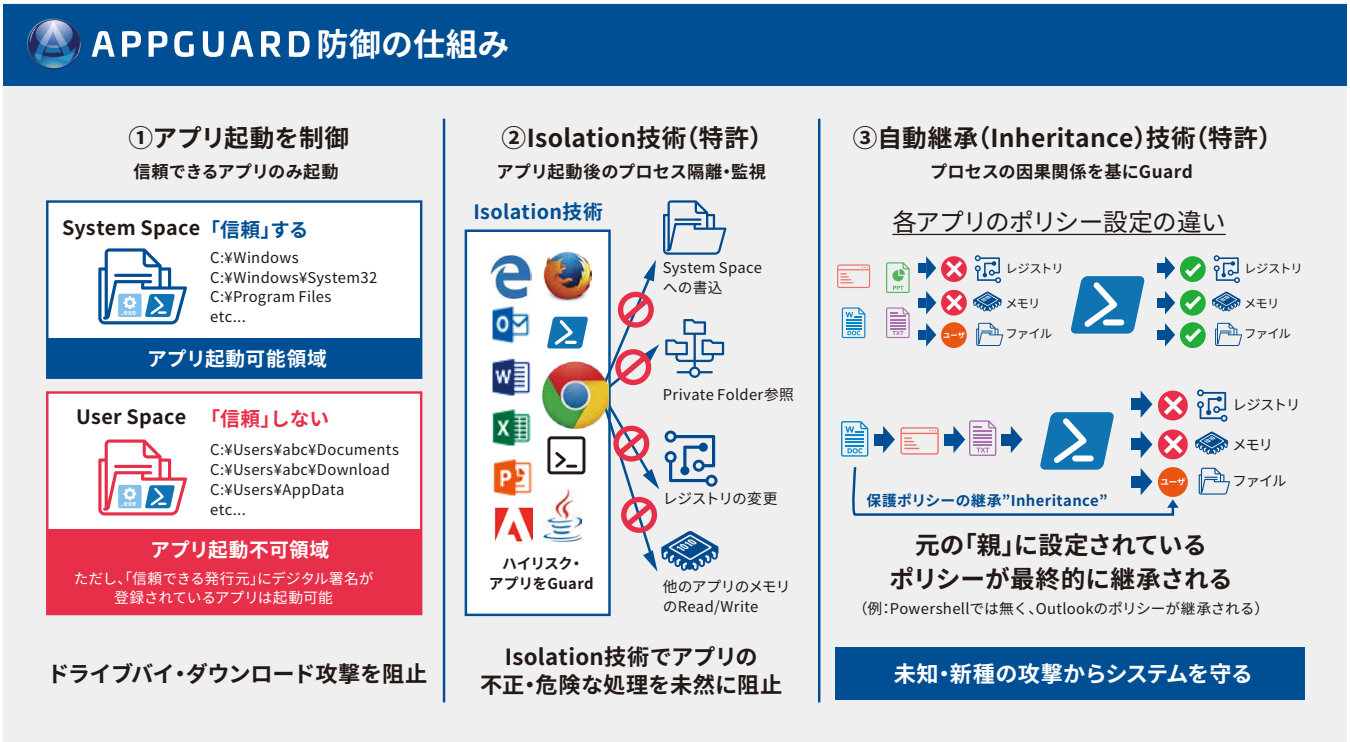
アプリケーション・ハードニングは、ダイナミックな動的環境下でのポリシールールの定義の仕方、そして運用・メンテナンス・検証がとても大変で難しいです。これらの課題を解決したのがAppGuardのIsolation（アイソレーション：隔離）技術です。Isolation技術は、下記の特許「Trusted Enclaves」がベースになっています（Blue Planet-works社の米国子会社AppGuard LLC社が保有する特許）。

Trusted Enclaves (US Patent# 7,712,143)

A trusted enclave for a software system of a computer node provides relatively high assurance protection of a section of the software system. The trusted enclave attempts to stop malware from compromising parts of the software system within the trusted enclave. If a software system process outside the trusted enclave becomes compromised, the compromised process may be prevented from compromising software system resources within the trusted enclave. Compromise of a process or resource of the software system refers to, for example, malware access, alteration or control of the process or resource.

Trusted Enclaves特許は、例えソフトウェアシステムが何らかの侵害（例えばマルウェアによる攻撃）を受けても、Trusted Enclave内は完全に保護され、Trusted Enclave外の侵害されたプロセスがTrusted Enclave内を侵害する事が出来ないようにしています。この特許をAppGuardのIsolation技術が応用しています。Isolation技術は、アプリケーションから派生する全てのプロセスの動作を監視（これをIsolationと呼びます。ファイルを物理的に隔離するわけではありません）する事が出来ます。どのプロセスがどこから派生し、どの様に展開されたかの一連の動き全てをIsolation技術で監視します。これにシステムに害を与えるプロセス動作をルール化（ポリシー化）する事により、システムの安全性を確保する事が出来ます。また、アプリケーションから派生する全てのプロセスの動作を監視できるため、派生していくプロセスの因果関係（元々実行しようとしていた動作）が分かるので、動的環境下でもシステムに害を与えるプロセス動作か否かの判断が出来ます。派生していくプロセス全てをIsolation技術でトラッキングできる事をInheritance（自動継承）と呼び、Isolation技術の一つの大きな特徴になります。このIsolation技術の実装方法が非常にコンパクトにできており、AppGuardのエンジンは、１MB以下と超軽量です。このIsolation技術の仕組み自体は不変的なものなので、エンジンのアップデートや更新は不要です。AppGuardは従来の「検知型」製品と比べ大幅に軽量であると同時に、アップデートや更新が不要です。

AppGuardの防御の仕組み：Trust but Verify（信頼あるものでも検証する）



AppGuardはTrust（信頼）をベースに構想され、システムの安全性を確保します。基本のコンセプトは、「正しい動作を守る」、「システムに害を与えるプロセス動作を未然に阻止する」、「システムに悪さをさせない」、「本来のOSの動作を守る」、「マルウェアを含む、システムに害を与えるすべての行為を阻止する」などで表現できます。AppGuardでは、大きく３つのステップでシステムの安全性を確保します。

Step 1：信頼できるアプリケーションのみ起動が可能

AppGuardは、System Space（C:¥Windows, C:¥Windows¥System32, C:¥Program Files¥,　等）領域は、信頼できる領域になっており、ここからのアプリケーションの起動は全て可能です。
AppGuardは、User Space（C:¥Users¥abc¥Documents, C:¥Users¥abc¥Download, C:¥Users¥AppData, C:¥ProgramData, D:¥, E:¥... 等）は、信頼しない領域になっており、ここからのアプリケーションの起動は、デフォルトでは不可になっています。これは、ドライブバイダウンロード攻撃を阻止するためです。User Spaceからアプリケーションを起動する場合は、信頼できる発行元（Trusted Publishers List）にそのアプリケーションのデジタル署名を登録する必要があります。例えば、Windows Update等は、Microsoftのデジタル署名が事前にデフォルトで信頼できる発行元に登録されている為、User Spaceから起動する事が出来ます。

Step 2：ハイリスクアプリケーションの起動後の動作プロセスの監視 - 信頼あるものでも検証する –

AppGuardでは、信頼できるアプリケーションのみが起動可能ですが、信頼できるアプリケーションの中でも、マルウェアの感染経路になりやすいものをハイリスクアプリケーションと設定されています。これらのハイリスクアプリケーションは、Guard対象になり、アプリケーションの起動後もプロセスの動作がIsolation技術により監視されます。システムに害を与える行為として、各ハイリスクアプリケーションに対し、下記の4つの処理についてのポリシーがデフォルトで設定されています。

1.System Spaceへの書込みが可能か否か
2.Private Folderの参照が可能か否か
3.OSのレジストリの変更が可能か否か
4.動作中の他のアプリケーションのメモリの読込・書込みが可能か否か（Memory Guard）

上記の4つに対し、プロセスが禁止されている動作を実行しようとすると、未然に阻止されます。

デフォルトでのハイリスクアプリケーションには、マルウェアの感染経路になり易いものやファイルレスマルウェアに利用されやすいものが設定されています。下記が、デフォルトで設定されているハイリスクアプリケーションの例になります。ユーザ自身がハイリスクアプリケーションを追加・編集する事が可能です。

・ブラウザー（MS IE, MS Edge, Chrome, Firefoxなど）
・MS Office: Outlook, Word, Excel, Powerpoint等
・Adobe Acrobat Reader
・Java
・Command Line
・PowerShell　等

デフォルトのポリシー例として、MS Outlookは、System Spaceへの書込みは不可、Private Folderへのアクセスは可、OSのレジストリの変更は不可、動作中の他のアプリケーションのメモリの読込と書込みは不可になっています。

Step 3：動的環境下でシステムに害を与える行為か否かを識別できるInheritance（自動継承）

AppGuardのIsolation技術の特徴的な部分がInheritance（自動継承）です。Step 2に記されるようにIsolation技術はハイリスクアプリケーションの動作処理を監視していますが、これをアプリケーション単体で監視しているのでは無く、そのアプリケーションから派生する全ての実行可能なファイルやロードされるDLLをIsolation化し（監視対象下に置き）、システムに害を与える行為を行わないか、Step 2に記された4つの動作を見ています。ここで重要なのが、Inheritance（自動継承）です。Inheritanceは、最初に起動されるハイリスクアプリケーション（親のアプリケーション）に適用されているポリシーが派生していく子や孫のプロセスにも反映されます。例え派生する子のプロセスのポリシーが親と異なっていても、親のポリシーを継承する仕組みです。

例えば、NotepadにてSystem Spaceにある設定ファイル（コンフィグレーションファイル）を書き換える行為は認められています。但し、NotepadがPowerShellやOutlookのメーラーから派生したプロセスであれば、PowerShellやOutlookのポリシーが適用されます。OutlookやPowerShellのポリシーでは、System Spaceの書き換えが禁止されているので、この場合Notepadは、System Spaceにある設定ファイルを書き換える事が出来ません。これにより、メールに添付されているドキュメントの中に悪意のあるコードが埋め込まれ、PowerShellを使いSystem Spaceを書き換えようとする巧妙な攻撃があった場合、これを未然に阻止する事が出来ます。Outlookメーラーは、そもそもSystem Spaceを書き換えたりしないので、このInheritanceにより親のポリシーが適用され、PowerShellやスクリプトを巧妙に使う攻撃であった場合でも、システムに害を与える行為を未然に阻止します。

Inheritanceがある事により、IOA（Indicator of Attack）情報を収集する事が出来ます。通常の検知型ですとIOC (Indicator of Compromise)という事で、マルウェアに侵害されたインジケーターが通報され、即時対応が求められます。IOAの場合、システムに害を与えようとする行為を未然に阻止したと言う情報がログとして上がり、また、Inheritanceによりその末端の不正行為がどのアプリケーションから始まったのかと言うプロアクティブな情報を入手する事ができます。IOAは、言わばフライトレコーダーのようなものです。第一に、システムは侵害されていませんのでシステムの安全性は確保されています。第二にどこからシステムに害を与えるプロセスが始まったのかが瞬時に分かります。IT管理者は、このログ情報を基にリアクティブな緊急対応では無く、プロアクティブな攻めのIT運用管理を実施する事が出来ます。

システムに害を与える行為を未然に阻止する：Kernelインターセプター

Kernelインターセプターにより、不正な動作を未然に阻止する事が出来ます。Kernelインターセプターは、システムの重要な動作をリアルタイムで把握します。例えばプロセスの動作、ファイルシステムへのアクセスや書込みの動作、コンフィグレーションを変更しようとする動作、ネットワーク動作等があります。これらをKernelレベルで監視しています。Windows OSの場合、全てのインターセプターはMicrosoftから提供されている正規なAPIを使っており、多くのソフトウェア会社が使用するAPI Hook（フッキング）は一切使用していません。フッキングの問題は、Windows OSがアップデートされた場合等、動作が不安定になり、システムがブルースクリーンになる現象があります。2018年1月にあったIntelプロセッサにまつわるMeltdownやSpectreの問題が発生した際、フッキングを使っていたセキュリティソフトは、Windowsのアップデートを行う前にそのセキュリティソフトをアップデートしなければブルースクリーンエラーが起こると警告していました。

AppGuardでは、Kernelインターセプターによりシステムの動作をリアルタイムで監視し、設定されているポリシールールと照合し、その動作を許可または阻止します。これにより、システムの正しい動作を守りシステムの安全性を確保します。Kernelインターセプターはリアルタイムで動作する為、システムのパフォーマンスへの影響はありません。AppGuardのIsolation技術が阻止するものは、あくまでもある特定の動作です。アプリケーションそのものが動作しなくなる訳ではありません。そのアプリケーション内で、例えば不正にOSのレジストリを書き換えようとする行為があれば、その行為のみを未然に阻止します。アプリケーション自体は普通に動作し続けます。

まとめ：AppGuardの優位性

AppGuardのエンジンは1MB以下で、とても軽量で軽快にサクサクと動作します。プロセッサへの負荷もほとんどありません。また、AI機械学習、振舞検知、アンチウィルスなどの「検知型」セキュリティ製品に必要な、脅威データの頻繁なダウンロードやAI機械学習エンジンのアップデートは全く不要です。ハードディスクを定期的にスキャンする必要もありません。AppGuardの目的はマルウェアを検知する事では無く、システムの安全性を確保する事にあり、システムに害を与える行為を特許取得済みのIsolation技術で未然に阻止します。また、AppGuardにより、セキュリティのトータルコストを大幅に削減する事が可能になります。システムの安全性が確保される為、24時間365日のエンドポイントの監視は必要ありません。IOC（Indicator of Compromise－システムが侵害された警報）により、リアクティブな緊急対応を強いられる事もありません。フライトレコーダーのようなIOA（Indicator of Attack）情報により、プロアクティブな攻めのIT運用管理が可能になります。また、AppGuardには多層防御の機能が含まれています。上記の「AppGuardの防御の仕組み」に記されたStep 1, 2、3が代表的な多層防御機能です。これ以外にもTrusted Enclave機能、Tamper Guard等ここでは書ききれない様々な機能が含まれており、システムの安全性を確保し、最新の脅威から守ります。ユーザは、一切AppGuardの機能を意識することなく、安心して業務に集中する事が出来ます。AppGuardの目的はシステムの安全性を確保する事です。ユーザの本来の業務に影響を与える事が無く、ゼロデイ、未知の脅威、ランサムウエア等の最新のサイバー攻撃からシステムを守る「新概念」です。