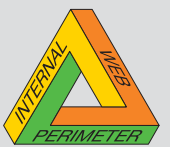


Malicious Code Protector

バッファ・オーバーフロー攻撃の検出・ブロックに対する新しいアプローチ

本書の内容

はじめに	2
バッファ・オーバーフロー攻撃	3
バッファ・オーバーフロー攻撃に対する防御の現状	3
新しいアプローチ Malicious Code Protector	4
アルゴリズム	5
技術革新	6
既存ソリューションに対する利点	7
Malicious Code Protector とチェック・ポイント製品	7
まとめ	8
用語解説	9



Check Point
SOFTWARE TECHNOLOGIES LTD.

We Secure the Internet.

はじめに

バッファ・オーバーフローの脆弱性を利用した攻撃はインターネット・セキュリティの最大の関心事項となっています。最近のCERTアドバイザリの50%以上がバッファ・オーバーフローに関するものであり、実際、2003年の最初の6アドバイザリは、バッファ・オーバーフロー脆弱性に直接関連するものでした。攻撃の威力の大きさや、多くのホストがバッファ・オーバーフローに対し脆弱性が存在することが、この攻撃が広く悪用される原因となっています。過去数年間で発生した最大の攻撃のいくつかは、バッファ・オーバーフロー脆弱性を利用したものでした。

攻撃名	発生年	経済的被害額 ¹	バッファ・オーバーフロー脆弱性
Sasser	2004	35億ドル	Microsoft Local Security Authority Subsystem Service (MS04-011)
SQL Slammer	2003	10億ドル	Microsoft SQL Server and MSDE 2000 (MS02-061)
MS Blaster	2003	7億5千万ドル	Microsoft RPC- Remote Procedure Call (MS03-039)
Code Red	2001	26億ドル	Microsoft IIS Server: Internet Server Application Program Interface (MS01-033)

バッファ・オーバーフロー攻撃は不正コード攻撃またはスタック攻撃とも言われ、攻撃者が悪用する基本的な理由としては、攻撃者がターゲット・ホストに対して思い通りの攻撃を可能とすること、そして、多くのアプリケーションがこの攻撃に対して脆弱であることが挙げられます。この攻撃方法を使用して、ハッカーはファイルを変更したりアプリケーションを起動するほか、悪意のある実行コードを送りつけることもできます。ワームは典型的な実行コード挿入の例で、バッファ・オーバーフローがワームのエントリ・ポイントとして利用されています。次の2つの例で見られるように、このワームは時間の経過とともにインターネットに広く感染していきます。Code Redワームは9時間以内に250,000のホストを攻撃しました²。さらに劇的な例としては、Wittyワームは最初の10秒で110のホストの攻撃に成功しています³。このような観点からすると、バッファ・オーバーフロー攻撃は今後も続くことが予測されます。

このような被害を予防するために、バッファ・オーバーフロー攻撃が出現してから攻撃を認識する対策がさまざまなベンダーにより開発されていますが、攻撃が出現してから攻撃を認識する手法では、新種や亜種の攻撃を認識することはできません。攻撃の拡大や拡散スピードを考えると、これは重大な問題です。この問題を解決するため、チェック・ポイントはMalicious Code Protectorを開発しました。これは、バッファ・オーバーフロー攻撃の検出とブロックを行うための新しいアプローチを提供する、特許出願中の技術です。定義済みの攻撃シグネチャに頼る従来の検出ソリューションとは根本的に異なる手法である、Malicious Code Protectorは、未知の攻撃や既知攻撃の亜種も検出できる効果的な検出方法を採用しています。

1:Computer Economics誌「The Impact of Malicious Code」(2004年6月)

2:CERT/CC, 「CERT® Advisory CA-2001-23 Continued Threat of the “Code Red” Worm」(2002年1月17日)、URL: <http://www.cert.org/advisories/CA-2001-23.html>

3:Colleen ShannonおよびDavid Moore著「The Spread of the Witty Worm」の「Cooperative Association for Internet Data Analysis」、URL: <http://www.caidda.org/analysis/security/witty/>





バッファ・オーバーフロー攻撃

バッファ・オーバーフロー攻撃は、ホスト・マシンの入力データやメモリ領域の処理を行なう部分を主にターゲットとします。ホスト・マシンでアプリケーションが実行されると、メモリの一部（バッファ）が入力データに割り当てられます。アプリケーションに使用されるバッファが固定サイズなのに、アプリケーション自体がバッファに格納する入力データ・サイズに制限を設けていないことから、この問題が発生します。例えば、プログラマが26バイト以下のデータを想定し、適切なメモリを割り当てます。これに対し、ユーザが27バイトのデータを入力すると、アプリケーションがバッファに割り当てられた以上のデータを書き込み（オーバーフロー）、メモリが損なわれます。

なぜこれが重要なことなのでしょう。このちょっとしたプログラミングのミスが、アプリケーションの実行コンテキストを破り、攻撃に挿入された攻撃者の不正なコードを実行する機会を、攻撃者に与えてしまうことになるのです。バッファをオーバーフローさせる能力（脆弱性）を利用して、攻撃者は特定アプリケーションのバッファより大きく、マシン・アセンブリ言語で記述された実行コードをペイロードに含んだ攻撃を作成します。また、攻撃者はメモリ・ポインタ、または、実行中のアプリケーションをリダイレクトするアドレスをメモリに入れます。その結果、アプリケーションが入力データを取り込み、バッファをオーバーランさせ、攻撃のペイロードに含まれる不正な実行コードを指し示すメモリ・ポインタを検出し、ホストによって不正コードの実行を開始することになります。

ハッカーがこの攻撃を行なう理由は2つあります。まず、入力データを取り込むアプリケーションならどれでも攻撃を適用できる可能性があることです。ネットワークの観点では、Webサーバ（HTTP）、DNSサーバ、FTPサーバ、ネットワーク接続されたマイクロソフトのアプリケーションなどがあります。もう一つは、攻撃の一部として、ホストが攻撃者の不正コードの実行を開始するので、攻撃者は自由度の高い不正コードを作成できる点にあります。ポピュラーな不正コードの例として、リモート・シェル・コマンドの実行（コマンド・ライン）、さらなる攻撃のためのバックドアの開放、ワームの実行、トロイの木馬のインストールなどが挙げられます。

例えば、Microsoft IIS Serversにバッファ・オーバーフロー脆弱性が見つかった場合、攻撃者はこれらのサーバを完全にコントロールできる不正コードを書くことができます。Code Red攻撃は、代表的な例です。最初のCode Redは、単にwww.whitehouse.govを攻撃するためだけにターゲット・ホストを利用しました。Code Red IIは、ホストにトロイの木馬をインストールする、より深刻な攻撃でした。このように、同じ脆弱性と不正コードを利用して、2つの深刻な異なる攻撃が作成されたのです。

バッファ・オーバーフロー攻撃に対する防御の現状

理想的には、実際にバッファに割り当てられた以上のデータをアプリケーションが取り込まないこと、アプリケーションがバッファを通過してメモリ内に書き込むことをホストが許可しないことです。しかしながら、世界中には入力データの検証を行わないアプリケーションが多数あり、これらのアプリケーションを実行するホストは数え切れないほどあります。問題発生の可能性のある規模が大きすぎて、それぞれのアプリケーション、ユーザの入力を取り込むアプリケーション内のそれぞれの場所を特定し、検証を行うことは現実的ではありません。このジレンマを解決するため、バッファ・オーバーフロー攻撃が企業ネットワークに侵入するのを防ぐためのワンストップ・ソリューションとして、企業はネットワーク・ベースのソリューションを検討しています。

IDS/IPSソリューション

不正侵入検知システム（Intrusion Detection Systems; IDS）は、既知の攻撃パターン（シグネチャ）、またはデータ・ストリーム内に攻撃が存在することを示す異常性とデータを照らし合わせ、アプリケーション・レベルの攻撃を検出します。

IDSは、セキュリティ・ポリシーの微調整やセキュリティ問題発生後の対処のために従来から利用されてきましたが、事前の防御としての機能は備えていません。これは、攻撃が単一のバケットに含まれていることを考えると、重大な問題です。例えば、SQL Slammerワームは、攻撃力を持ち、かつ単一バケットに入った276バイトの小さなワームでした。不正侵入検知システムの利点は、シグネチャの検索条件を厳密に設定できるので誤警報の発生率が低いことです。



Check Point
SOFTWARE TECHNOLOGIES LTD.

We Secure the Internet.

短所は、新種の攻撃をなかなか検出できないことです。現在の攻撃をほんの少し改造するだけで、不正侵入検知システムは役に立たなくなります。Code Red IIの亜種は、オリジナルと13バイトの差があるだけで、オリジナルの3倍の速さで感染させることができます4。

不正侵入検知システムはマシン・コードを探すものもありますが、マシン・アセンブリ・コードの検証や、MCPが正確で迅速な攻撃検出のために使用する仮想サーバ分析などの機能はありません。実際、不正侵入検知システムは、マシン・アセンブリ命令を見るのではなく、アセンブリ命令が存在することが推察される特定のバイナリ文字列を見えています。実際のアセンブリ・コードの分析ではなくマシン・アセンブリ・コードの推察であることが、不正侵入検知システムが新種の攻撃を検出できず、既知攻撃の亜種の検出率が低いことの根本的な理由です。

不正侵入防止システム (Intrusion Protection Systems; IPS) は、一般的なシグネチャに基づいて、認知されたネットワークへの脅威を能動的にブロックします。しかし、不正侵入検知システムと同様、不正侵入防止システムもシグネチャ・ベースに大きく頼っているため、未知の脅威を事前に防ぐことはできません。

新しいアプローチ: Malicious Code Protector

Malicious Code Protectorは、特許出願中のCheck Pointの技術です。これは、バッファ・オーバーフロー脆弱性を持つアプリケーションをターゲットとする不正コード攻撃を検出する設計になっています。既知の攻撃または亜種のシグネチャを利用せずに、攻撃を認識することができます。以下に、Malicious Code Protectorのゴール、攻撃を正確に検出するための条件、および実際の不正コードを検出するメカニズムについて説明します。

設計ゴール

Malicious Code Protectorは、次のようなゴールを設定して設計されました。

1. 未知の攻撃を検出できる機能。Malicious Code Protectorは、攻撃のシグネチャではなく、実際の攻撃を検出することをゴールとしています。定義上、シグネチャとは新しい攻撃を予測したり、新しい攻撃のために「予め作成」されるものではありません。しかし、ネットワーク・ベースのバッファ・オーバーフロー攻撃の実行コードは、すべて共通の固有な特性を共有しています。実行コード自体を実際に検証してみるとその特性がさらによく分かります。
2. プロトコルに依存しないこと。Malicious Code Protectorは、多くのプロトコルをサポートするCheck Point Application Intelligenceフレームワークと統合するように設計されています。最近の攻撃で見られるように、MS-RPCやSQLといった従来より利用されているプロトコルも新種の攻撃によって突き破られています。
3. 高速。ワイヤ・スピードで機能するソリューションをゴールとして設計されました。つまり、ほとんどの企業ニーズに対応するには、1ギガ/秒以上の処理速度をサポートする標準のPentiumハードウェアで実行しなければならないということです。
4. 複数プラットフォーム互換。Malicious Code Protectorは複数サーバ・プラットフォームをサポートします。現在のソリューションがWindowsおよびIntelプラットフォーム (インターネット上の90%以上のサーバ) のみ防御するのに対し、Malicious Code ProtectorはSun Microsystems SPARCベースのシステムなど他のシステムまで防御の範囲を拡大しました。

4: Baca, Jeremy 著、『Windows Remote Buffer Overflow Vulnerability and the Code Red Worm』2001年SANS Institute刊





設計の前提条件

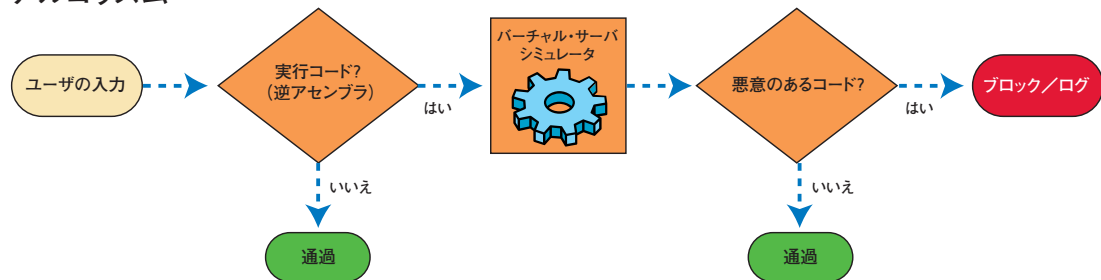
先に述べたように、バッファ・オーバーフロー攻撃は、巧妙に作成された不正コードを使用して、ターゲット・ホストのメモリ・システムを攻撃します。すべてのバッファ・オーバーフロー攻撃は、ネットワーク・ペイロードに含まれた実行コードを送りつけます。バッファ・オーバーフロー攻撃には、必ずターゲット・コンピュータで実行されるコードが含まれています。それがないと、不正行為を実行させることができません。

このことに基づき、以下の仮説を立てました。

1. ネットワーク・ベースのあらゆる攻撃は、マシン言語の実行コードを含んでいる。
2. ネットワーク・トラフィックは、常に実行形式のマシン・コードを含んでいるとは限らない。本物の実行コードを転送するケース(.exeファイルのダウンロード等)はまれであるため、このようなケースは簡単に特定できます。通常EXEファイルはサーバからクライアントに送信されますが、攻撃はクライアントからサーバに対して実行されます。
3. ネットワーク・トラフィック上のマシン・コードをかなりの精度(最小限の誤検出かつハイ・パフォーマンス)で検出するアルゴリズムを記述することが可能である。

上記の仮定に基づき、Check PointはMalicious Code Protectorの設計ゴールを達成するアルゴリズムを作成しました。Malicious Code Protectorはネットワーク・トラフィック内のマシン・コードを検出することが可能です。また、上記仮定から攻撃にはマシン・コードが含まれていることがわかっています。そして、このアルゴリズムは、攻撃対象となっている特定のバッファ・オーバーフロー脆弱性に関係なく、実際の攻撃を検出することが可能となります。

アルゴリズム



実行コードの検索

Malicious Code Protectorの心臓部は、ネットワーク・トラフィックの実行コードを検出する逆アセンブラ・エンジンです(バイナリ・データのマシン・アセンブリ言語への逆アセンブル等)。実行コードの検出機能では、FTPによる実行ファイル(*.exe)の転送などの例外を除き、一般的には、実行コードはネットワークを行き来しない、という仮説に基づいています。Malicious Code Protectorはデータ・ストリームを監視し、逆アセンブラ・エンジンがマシン・アセンブリ言語に変換できる一連のデータを検索します。そのようなデータは、ネットワークを介する実行コードの受け渡しの存在を示唆します。

しかし、データ・ストリームに実行コードが入っているというだけでは、不正なコードであると断定するには不十分です。非実行形式のデータによってアセンブリのような出力を生成できる例もあります。例えば、.gifファイルはアプリケーションではありませんがマシン・アセンブリ命令を生成することがあります。したがって、Malicious Code Protectorはネットワーク上のアセンブリに似たデータの「ノイズ」と本物の実行コードを区別できなければなりません。

Check Point
SOFTWARE TECHNOLOGIES LTD.

We Secure the Internet.

仮想サーバ:悪意の特定

疑わしいブロックのデータが実際に実行コードであるかどうか判断するため、Malicious Code Protectorは一連のアセンブリ命令 (Op Codesなど) を調べ、命令とその想定使用法との論理的な関連を探ります。アセンブリ命令の大きなコンテキストと大きな「メタ命令」の存在から、Malicious Code Protectorはこの一連のアセンブリ命令は本物の実行コードの可能性があると判断します。

しかし、存在する実行コードが不正であるとは限りません。Malicious Code Protectorの最終ステップは、一連のアセンブリ命令を調べて特定のメタ命令セットを探すことです。先のバッファ・オーバーフロー攻撃の説明で述べたように、ハッカーは不正コード内に特定のプログラミング技法を使用して、ターゲットに不正コードを実際に実行させます。通常は、バッファのサイズと場所に関連する不確実性に作用する一連の類似の命令を含ませません。Malicious Code Protectorはこのような命令を探して不正アプリケーションを特定します。そして、命令に対する反応を分析するため、その命令を仮想サーバに渡します。この仮想環境が、Malicious Code Protectorが不正コードを検出できる根本的な理由です。最後に、MCPはこれらすべてのステップの結果を分析し、データ・ストリームが本当に不正コードであるかどうかを判断します。

技術革新

逆アセンブリ技術には方々で長くの歴史がありますが、精度とパフォーマンスの観点から、ネットワーク・ベースの攻撃の検出に使われることはありませんでした。しかし、逆アセンブリ技術をCheck Pointのステートフル・インスペクションとApplication Intelligenceの技術に統合したことによって、こうした制限を克服することができました。この組み合わせにより、アプリケーション・フローをより深く理解して精度を増すことができます。さらに、ネットワーク・トラフィックに不正な攻撃を注入できるケースに限って逆アセンブリを適用し、パフォーマンスを大幅に向上させています。

精度

マシン学習手法において精度は重要な問題です。検出率の向上と誤検出率の低下の狭間で、どちらも同時に達成できる機能が求められています。

逆アセンブラのみを使用して不正コードを検出しようとすると、検出率は高くなりますが、同時に誤検出率も高くなってしまいます。これを避けるため、Malicious Code Protectorは逆アセンブリの第2層を使用しています。これは、マシン・アセンブリ・コードをループや分岐などの論理プログラミング要素に変換するものです。このアプローチでの論理的な文法構造は高く、自然発生的に出現するランダム構造の可能性は低いため、誤検出率はかなり低くなります。実行コードが実行しようとすることを理解できることにより、高い確率で悪意を区別することができます。

検出精度を高めるため、Malicious Code Protectorはハッカーがバッファ・オーバーフローで使用する論理的な文法構造を検索します。すべてのバッファ・オーバーフローはサイズが限られており、ネットワーク・ベースの攻撃から継承された制限を共有しているので、不正コードを示唆する復号化ループなどの繰り返し構造を区別することができます。

いくつかの検出システムを実行した後で、Malicious Code Protectorはマシン学習技術を使用して各フローのアルゴリズムによって達成されたウェイトを組み合わせます。ウェイトの組み合わせにより例外を除外して共通のコンテキストを見ることにより、精度を高めることができます。





パフォーマンス

マシン・アセンブリ言語へのバイト逆アセンブリは、コンピュータに激しい操作です。コード分岐も追跡できる双方向の逆アセンブリは、より多くのコンピューティング資源を必要とします。このようなパフォーマンスの問題を解決してワイヤ・スピードで動作するため、Malicious Code Protectorにはいくつかのパフォーマンス拡張技術が実装されています。パフォーマンスの考慮点には、以下のようなものがあります。

1. 不正コードを挿入可能な場所のみアルゴリズムを適用することで、パフォーマンスが大きく向上します。例えば、クライアントとWebサーバ間のWeb通信について言えば、サーバがユーザ入力を検索する場所は数箇所 (URL、フォームなど) に限られます。入力に伴ってサーバが動作するのはこれらの場所だけであるため、ハッカーが不正コードを挿入するのも同じ場所です。Application Intelligenceによってアプリケーション通信のコンテキストを理解し、Malicious Code Protectorは高速かつ高精度で稼動することができます。
2. Malicious Code Protectorはストリーム・ベースであるため、1バイト単位に調べることができます。これにより、パケットについて保管ではなく転送するハイ・パフォーマンス環境で使用することができます。
3. Malicious Code Protectorは、反復操作計算を避けるためのキャッシュ、ハッシュ、情報共有など、特定の最適化技術を実装します。リクエストは反復する傾向にあるので (サーバ名など)、これは特に有用です。

既存ソリューションに対する利点

攻撃のシグネチャ (IDS/IPS) を使用するネットワーク・ベースのシステムと比べ、Malicious Code Protectorには以下の2つの根本的な利点があります。

1. 先制の防御。既知の攻撃の分析から作成されたシグネチャに依存する必要がないため、Malicious Code Protectorには攻撃の出現からシグネチャが定義されて公表されるまでの時間差がありません。仮想サーバ環境でのマシン・アセンブリ言語の分析により、Malicious Code Protectorはバッファ・オーバーフロー脆弱性を突こうとする新種の攻撃を検出することができます。これにより、「Day - Zero」からの攻撃防御を提供します。
2. 精度。実際マシン・アセンブリ言語と仮想サーバ環境でのその振る舞いを見ることにより、Malicious Code Protectorは、不正コードに続いて発生する亜種を検出することができます。(例えばCode Red IIIはオリジナル攻撃の7日後に発生しました)。一般的に攻撃亜種のバイナリ「シグネチャ」は、既存のシグネチャと大きく異なってしまうため、攻撃自体は同じバッファ・オーバーフロー脆弱性を突いているにもかかわらず、亜種という把握はできません。Malicious Code Protectorは、不正コードを示すバイナリ文字列ではなく不正コードのアクションを調べるため、この欠陥を克服することができます。

Malicious Code Protectorとチェック・ポイント製品

先の設計ゴールの説明で述べたように、Malicious Code Protectorはプロトコルに依存しない設計になっています。いずれのタイプのネットワーク・トラフィックに対しても実行することができ、ターゲット・ホストのバッファ・オーバーフロー脆弱性を突く不正コードを検索することができます。Malicious Code Protectorの最初のリリースはWeb Intelligenceとともに提供され、Web (HTTP) トラフィック検査に使用されます。Web Intelligenceに統合され、Malicious Code Protectorは、Webサーバへのユーザ入力 (URL、Webページ内のフォーム等) を伝送する可能性のあるWebトラフィックの検査に使用されます。

Web Intelligenceは以下の製品に統合されています。

- VPN-1 NG with Application Intelligence、R55W以降のバージョン: Web IntelligenceはVPN-1ゲートウェイの背後のWebアプリケーションを保護します。
- Connectra: Web IntelligenceはConnectraゲートウェイを介してアクセスされるWebサーバを保護し、同時にConnectra自身も保護します。





We Secure the Internet.

ステップ	説明	データ・ストリームの表示
1	不正コード攻撃が埋め込まれたデータ・ストリームの例です (不正コードを青および斜体文字で示しています)。	<pre> ...ff 73 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 6f 6f 6f 6f 6f 6f 6f 6f 6f 6f 42 42 42 42 8b 89 e8 77 eb 15 5b 53 68 aa 01 78 58 ff d0 31 c9 b1 11 58 e2 fd 31 c0 48 c3 e8 e6 ff ff 73 74 61 72 74 20 63 61 6c 63 2e 65 78 65 00 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a 20 63 6c 6f 73 65 0d 0a 0d 0a 0d 0a...</pre>
2	HTTP Webプロトコルを理解しているため、Web Intelligenceはこのストリームをユーザ入力データのあるHTTP GETとして認識し、ハッカーが不正コードを埋め込む可能性のある場所としてとらえます。	<pre> GET / HTTP/1.1 Host: oooooooooooooBBBBie_w_ [Sh!_ xX _1_ X_1_H_μ start calc.exe Connection: close</pre>
3	この場所を攻撃の可能性のあるポイントとし、Malicious Code Protectorの逆アセンブラ・エンジンはデータ・ストリームを調べてマシン・アセンブリ・コードを探します。データ・ストリーム内で、ある種のデータはアセンブリ・コードにランダムに変換されることがあります。しかし、この場合はアセンブリ・コードの文字列により実行コードの存在が明らかになります。	<pre> JMP+26, CALL-26, POP EBX, PUSH EBX, PUSH 0x7801AAAD, POP EAX, CALL EAX</pre>
4	最後のステップでは、Malicious Code Protectorが仮想サーバ環境で実行コードを実行します。実行コードの振る舞いの分析結果に基づき、実行コードがバッファ・オーバーフロー脆弱性を突き破ろうとする不正コードであるかどうかを判断することができます。	<pre> graph LR A[バーチャル・サーバシミュレータ] --> B{振る舞いの分析} B -- はい --> C[ブロック/ログ] B -- いいえ --> D[通過]</pre>

表1: Web IntelligenceがMalicious Code Protectorを使用してWebトラフィックを検査するステップ

まとめ

アプリケーションのバッファ・オーバーフロー脆弱性を突く攻撃がセキュリティ上の深刻な問題となり、何十億ドルもの経済的被害が発生しています。

脆弱性を持つ何千ものアプリケーションがあり、日々新しい脆弱性が見つかる中で、これは今後も継続した問題となる可能性があります。現在、攻撃の出現から攻撃シグネチャの記述と配布までの間に脆弱性が存在します。不正コードを含むワームが何千ものホストにわずか1分で感染できることを考えると、防御におけるこの遅れは大きな問題です。Malicious Code Protectorは、バッファ・オーバーフロー攻撃に対する防御を提供する新しいアプローチです。実際の実行コードを調べてそれを仮想サーバ環境で実行できる機能により、Malicious Code Protectorはインターネットに出現する攻撃を最小限に抑えるために欠かせない「Day - Zero」防御を実現します。





We Secure the Internet.

用語解説

マシン・アセンブリ言語: CPUによって実行される実際のコンピュータ言語。C++などのハイレベル・プログラミング言語の観点から言うと、マシン・アセンブリ言語はハイレベル・プログラミング言語によって記述されたプログラムを取り込み、コンパイラによってプロセッサ用にコンパイルされたものです。

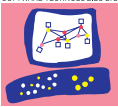
逆アセンブラ: バイナリ・データを取り込み、特定のプロセッサ (Intel、SPARC等) に基づいてマシン・アセンブリ言語に変換するツール。変換の結果、プログラムを実行するために使用される実操作命令となります (XOR、NOOP等)。

Check Point Software Technologiesについて

チェック・ポイント・ソフトウェア・テクノロジーズ (www.checkpoint.com) はインターネット・セキュリティにおける世界トップ企業として、特に企業向けファイアウォール、パーソナル・ファイアウォール、およびVPNの市場においてマーケット・リーダーとして広く認められています。チェック・ポイントは Next Generation製品ラインナップを通じ、インテリジェント性を兼ね備えた境界、内部、およびWeb環境に対するセキュリティ・ソリューションを提供し、エンタープライズ・ネットワークをはじめ、アプリケーション、エンドポイント、支店・支社環境、更にはパートナー各社のエクストラネットなどに対する包括的なセキュリティ保護を実現します。チェック・ポイントの一部門である Zone Labs (www.zonelabs.com) は、インターネット・セキュリティの分野で高い信頼性を誇るブランドとして数々の賞に輝くエンドポイント・セキュリティ・ソリューションを提供し、世界中で何百万台ものコンピュータをハッカーやスパイウェア、データの盗難などから守っています。またチェック・ポイントは、350社を超える各分野のトップベンダーが提供する最高のソリューションとの統合および相互運用性を実現するフレームワークであるOPSEC (Open Platform for Security) により、自社ソリューションの能力をさらに拡大します。現在チェック・ポイントは世界88ヶ国、2200社を超えるパートナー・ネットワークを通じてソリューションの販売、導入、サービス提供を行っています。



Check Point
SOFTWARE TECHNOLOGIES LTD.



We Secure the Internet.



Check Point[®]
SOFTWARE TECHNOLOGIES LTD.

We Secure the Internet.

チェック・ポイント・ソフトウェア・テクノロジーズ株式会社
〒160-0022 東京都新宿区新宿5-5-3 建成新宿ビル6F
<http://www.checkpoint.co.jp/> E-mail : info@checkpoint.co.jp Tel : 03 (5367) 2500

©2004-2005 Check Point Software Technologies Ltd. All rights reserved. Check Point, Application Intelligence, Check Point Express, Check Pointのロゴ, AlertAdvisor, ClusterXL, Cooperative Enforcement, ConnectControl, Connectra, CoSa, Cooperative Security Alliance, Eventia, Eventia Analyzer, FireWall-1, FireWall-1 GX, FireWall-1 SecureServer, FloodGate-1, Hacker ID, IMsecure, INSPECT, INSPECT XL, Integrity, InterSpec, IQ Engine, Open Security Extension, OPSEC, Policy Lifecycle Management, Provider-1, Safe@Home, Safe@Office, SecureClient, SecureKnowledge, SecurePlatform, SecuRemote, SecureServer, SecureUpdate, SecureXL, SiteManager-1, SmartCenter, SmartCenter Pro, Smarter Security, SmartDashboard, SmartDefense, SmartLSM, SmartMap, SmartUpdate, SmartView, SmartView Monitor, SmartView Reporter, SmartView Status, SmartViewTracker, SofaWare, SSL Network Extender, TrueVector, UAM, User-to-Address Mapping, UserAuthority, VPN-1, VPN-1 Accelerator Card, VPN-1 Edge, VPN-1 Pro, VPN-1 SecureClient, VPN-1 SecuRemote, VPN-1 SecureServer, VPN-1 VSX, Web Intelligence, ZoneAlarm, ZoneAlarm Pro, Zone Labs, Zone Labsのロゴは, Check Point Software Technologies Ltd. およびその関連会社の商標、サービス・マーク又は登録商標です。その他の企業、製品名は各企業が所有する商標または登録商標です。本書で記載された製品は米国の特許No.5,606,668、5,835,726および6,496,935により保護されています。その他の米国における特許や他の国における特許で保護されているか、出願中の可能性があります。

P/N 505081-J 2005.5



Intelligent Security