情報ネットワーク特論
マルウェアとその解析

名古屋大学 情報基盤センター
情報基盤ネットワーク研究部門
嶋田 創

概要

- マルウェアとは
- マルウェアの分類
- 近年のマルウェアの送り込み方
- マルウェアの静的解析
- マルウェアの動的解析
- 解析されないためにマルウェアが取る対策

マルウェアとは

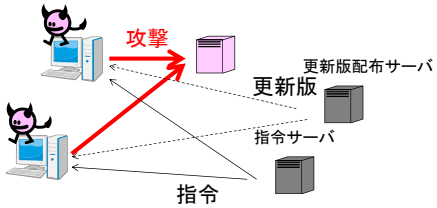
- MALicious softWARE(悪意のあるソフトウェア)の略
- かつてはよくコンピュータウイルスと呼ばれていたが、最近はマルウェアと称すことが多い
- コンピュータウイルスとの違い
  - 愉快犯や技術誇示からサイバー犯罪の道具へ
  - おおっぴらに感染/拡散しない
    - 特定のグループ/ネットワークのコンピュータにのみ感染
    - そもそも、あまりばらまくと発見される可能性が高くなる
  - おおっぴらに怪しい通信したりしない
    - 他の通信にまぎれて通信したりします
  - おおっぴらに破壊活動をしたりしない
    - 発見されると証拠隠滅することはあります

マルウェアの分類

- RAT(Remote Access Trojan)
  - 遠隔で感染したPCを操作可能な形にする
  - トロイの木馬、ボットネットクライアント、などもこれに分類
  - 自分が加害者になる点が怖い
- スパイウェア
  - 金融関係情報や各種サービス用ユーザ名/パスワードの窃取
  - キーロガーやスクリーンショット取得などの機能
- ダウンローダ
  - Drive-by Download攻撃の途中で利用
- 昔ながらのもの
  - ウイルス: 無差別に近い拡散、PCに何らかの以上を発生させる
  - ワーム: 増殖することに特化

RAT(Remote Access Trojan)

- トロイの木馬、バックドア作成、踏み台ツールの発展
- 指令を受け取って攻撃などの動作を取る
  - 昔はIRC経由が多かったが、マークされるようになったので最近はHTTPやHTTPS経由で指令受信
  - 後述するDDoS攻撃などに利用
  - 遠隔で自分自身を更新することも可能



代表的なRAT: Poison Ivy

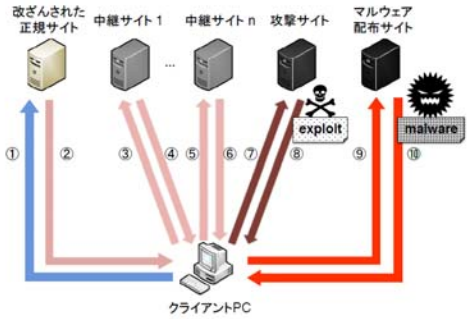
- バージョンアップしながら今も利用されている
- 機能
  - スクリーンショット、音声、Webカメラの画像の取得
  - アクティブなポートの表示
  - キー入力操作情報の収集
  - 開いているウィンドウの管理
  - パスワードの管理
  - レジストリ、プロセス、サービス、デバイス、インストールされているアプリケーションの管理
  - ファイル検索、同時に多数のファイル移動の実行
  - リモートシェルの実行
  - サーバの共有
  - 自身の更新、再起動、終了

マルウェアの送り込み方

- 昔ながらのメール
  - 本体に添付することは減ってきてウェブからのダウンロードが中心に
  - 複数のダウンロードを繰り返すDrive-by Download攻撃も
- ウェブからのダウンロード
  - 攻略されたウェブサイトから配布
  - ウェブ広告にまぎれて配布
- 標的型攻撃(APT: Advanced Persistent Threat)
  - 近年話題の攻撃
  - 標的(機密情報サーバなど)にマルウェアを入れるまでに複数の踏み台PCを経由
  - 応用: 水飲み場型攻撃
    - 特定のユーザがよく見るウェブサイトにマルウェアをしかける

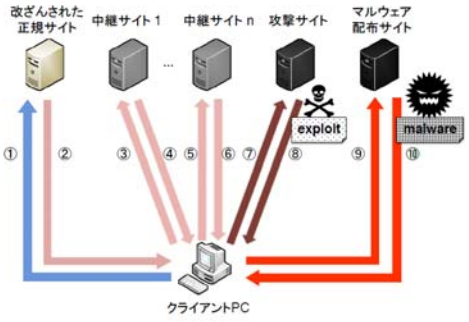
Drive-by Download攻撃

- 複数のダウンロードによってマルウェア配布を欺瞞/隠蔽



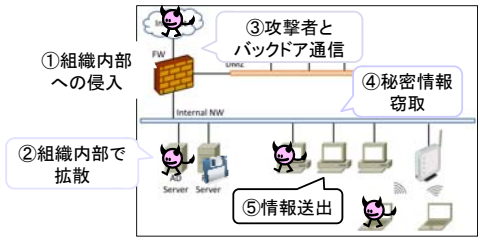
Drive-by Download攻撃

- 複数のダウンロードによってマルウェア配布を欺瞞/隠蔽



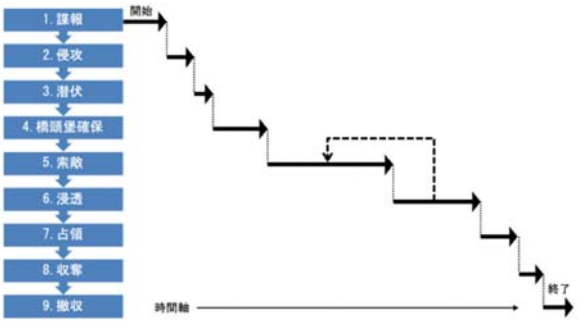
標的型攻撃の動作

- 非常にざっくり書くと5段階
  - 侵入前に組織の内部構成を調査することもある
  - 組織内での拡散において、潜伏、索敵を行うこともある



標的型攻撃の時間軸

- 長いものだと一連の攻撃に数ヶ月かける



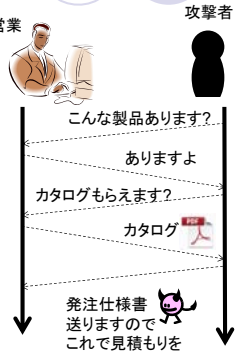
標的型攻撃の実例

三菱重工への標的型攻撃

- 発覚: 2011/8/11にサーバが再起動を繰り返すため
- 影響範囲
  - サーバー 45台、従業員用PC 38台
  - 8種類のウイルスを発見
  - 11の事業所から発見
- 発端: “原発のリスク整理”という添付ファイル
  - 東日本大震災(2011/3)の直後
  - Adobe Flashの脆弱性を利用
  - 送信元は内閣府実在の人物の名前、メールアドレスを騙る
  - 三菱重工は原発を作っている(いた)ので、受け取った人は疑わない

さらに発展した標的型攻撃

- やりとり攻撃
  - 複数回のメールのやりとりの後にマルウェア送付
- 水飲み場型攻撃
  - 「ある仕事をしている人が頻繁に見るページにマルウェアを仕掛ける」ことによる特定業種の業社への標的型攻撃
  - 例: 政府のある機関のプレスリリース、入札公告ページ
    - その機関に関連する会社に対して攻撃
    - さらにIPアドレスを制限する事例もある



水飲み場型攻撃の実例?

EmEditorアップデートファイルを利用した攻撃[1]

- 攻撃対象: 名古屋大学、JAXA、ISAS、朝日新聞、農林水産省など
- 以下の様な.htaccessファイルがアップデート配布ディレクトリに置いてあった
  - 指定したIPアドレスの範囲からアップデート要求があれば別ファイルを配布

```
SetEnvIf Remote_Addr “106¥.188¥.131¥.[0-9]+” install
SetEnvIf Remote_Addr “133¥.6¥.94¥.[0-9]+” install
(… 同様に70行 …)
SetEnvIf Remote_Addr “124¥.248¥.207¥.[0-9]+” install
RewriteEngine on
RewriteCond %{ENV:install} =1
RewriteRule (.*¥.txt)$ /pub/rabe/editor.txt [L]
```

[1] https://jp.emeditor.com/general/ 今回のハッカーによる攻撃の詳細について /

信頼性の高い攻撃用メールアカウントの準備

- 従来だったら
  - セキュリティのゆるいフリーメールアドレスを利用する
  - ただし、あまりにも評判が悪くなると後述のブラックリストで対策される
- 近年では
  - そこそこメジャーな組織のメールアカウントを乗っ取って送信
  - できれば、その組織に対して関係が深い組織のメールアドレス
    - 知り合いからのメールと見せかければ開封される可能性は数十倍高い
  - 部署までやりとりに関係する所があればさらに好ましい
  - 多段階でメールアドレス乗っ取りをかけることも

マルウェアの識別

- バイナリ全体のハッシュ値
- 部分コード列との一致
- 埋め込みデータとの一致

→まとめてシグネチャにする
→シグネチャ型IDS/IPS、一般的なアンチウイルスソフトウェア

マルウェアの名付け

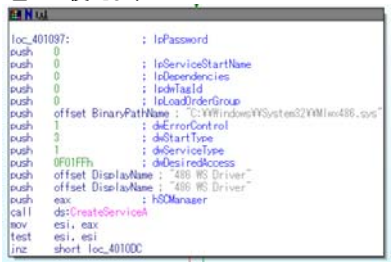
- Antivirusメーカごとに名付けが違う
  例
  - Kaspersky: Trojan-Ransom.Win32.Agent
  - AVG: Trojan.Generic35
- それどころか、ファミリ分けも違う
  例
  - AVG
    - Trojan.Win32.Agent
    - Trojan.CoinMiner.AKQ
    - Trojan.Dropper.Generic_r.AF
  - Kaspersky: 全部Trojan.Inject2.MDE

VirusTotalによるマルウェアの識別

- 様々なAntivirusの結果を見ることができるサイト
- URL(ウェブサーバ)に対しても評価してくれる
  - というか、ちょこちょこ巡回しているらしい

静的解析とは

- マルウェアのプログラムを解析するもの
  - プログラム解析のために少し程度はプログラムを動かすのもあり
- ごく一般的なプログラムのリバースエンジニアリングのテクニックが色々と使えます



静的解析でやること

- バイナリに埋め込まれた情報
  - アクセス先のホストなどの情報
  - 利用するライブラリや関数
- 普通のデバッグに近い解析
  - 変数の変化を追う
  - (ブレークポイントを設定して動作させて)制御を追う

静的解析でヒントとなる物(1/2)

- 特定のランタイムライブラリの呼び出し
  - 例: Advapi32.dll(レジストリ)
- 特定の関数の呼び出し
  - 例: GetProcAddress(プロセスのアドレス確認), SetWindowHookEx/LowLevelKeyboardProc(キー入力フック), など
  - 怪しい関数はMSDNのWinAPI検索で調べたり
  - 自作関数も命名から動きを想像できたり(例: URLDownloadToFile)
- mutex関係の関数の利用
  - 同じマルウェアが実行されていないか確認するためによくmutex関係の関数が利用される

静的解析でヒントとなる物(2/2)

- サービス関係の関数
  - OS起動時に自動起動するようにサービス登録
- Navigate関係の関数
  - ウェブへのアクセスで利用
- 変な例外処理
  - スタック上の例外処理用ジャンプテーブルをBoFなどで書き換えて(上の権限で)実行
- WinAPIではなくネイティブAPIの利用

動的解析とは

- 実際にプログラムを動かしてその挙動より解析
  - 外部のどこに接続しに行こうとするか
  - 外部から何をダウンロードして実行しようとするか
  - どのようなファイル、レジストリを触ろうとするか
  - LAN内の他のマシンにちょっかいを出すか
- 通常は仮想マシン上で動作させて確認
  - できれば、他を攻撃しない無害な通信は通過させた状態で動作
    - インターネットへの接続性で動作を帰るマルウェアは多い
    - 外部からのコマンドを受けて動作を継続するマルウェアもある
  - ただし、外部への接続は「解析者がいることを攻撃者に伝える」可能性もある

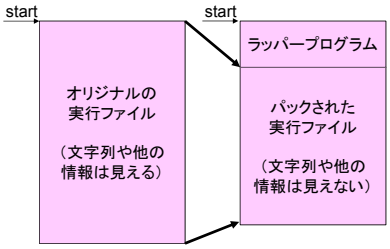
マルウェア解析を妨げるもの

- パック: プログラムを単純に圧縮
- 難読化
  - JavaScriprtなどソースコードにおいて難読化
  - 文字列として解釈できるデータを減らしてシグネチャのマッチを回避
- 逆アセンブル対策
- 動的解析(デバッガ)妨害
- 仮想マシン対策

パック

- まずパック部をデコードするラッパープログラムを起動
- 静的解析ではラッパーを解析することしかできない
- UPX(http://upx.sourcegprge.net/)が有名

→アンパック手法



コードの難読化

- JavaScriptの難読化

```
function yJDyoDLGpObSGUU(tJtMEppKbaFeYmsys){eval(tJtMEpp
urn String.fromCharCode(PuOEGGlvMC)} yJDyoDLGpObSGUU(x(1
x(103-6)+x(116-2)+x(41-9)+x(107-9)+x(40-8)+x(66-5)+x(32-
x(109-8)+x(107-0)+x(114-4)+x(117-0)+x(101-2)+x(115-8)+x(
-3)+x(105-4)+x(122-7)+x(46-0)+x(103-4)+x(117-6)+x(116-7)
8-1)+x(100-0)+x(110-1)+x(112-7)+x(119-9)+x(51-4)+x(116-7
x(49-2)+x(61-6)+x(52-1)+x(50-4)+x(105-4)+x(122-2)+x(107-
```

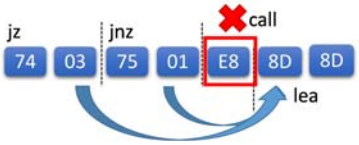
- 難読化を解除しましょう
  - 例: Online JavaScript beautifier http://jsbeautifier.org/

可変長命令ならではの逆アセンブル妨害

- x86命令は可変長命令セット
- 一部のバイト列を飛ばすと別の命令列になることも

例

- 普通に逆アセンブルするとjz->jnz->call
- 実際の動作はjz->leaもしくはjnz->lea



動的解析(デバッガ)妨害

- デバッガが使う関数を監視
- デバッガがアタッチする際の動作(Using the WinAPI)をスキャン
  - いくつかの関数はデバッガ検知に使われる
  - ドキュメントに載っていない機能を使うことも
- メモリ構造をチェック
- デバッガの残留物をサーチ
  - Windows側の出力や仮想マシンの出力より

デバッガ検出に使えるWinAPI

- IsDebuggerPresent
  - Process Environment Block (PEB) 構造体の中
  - デバッガが起動しているかどうか
- CheckRemoteDebuggerPresent
  - Process Environment Block (PEB) 構造体の中
  - デバッガがプロセスに接続されているかどうか
- NtQueryInformationProcess
  - あるプロセスがデバッグされているかどうか
- OutputDebugString
  - デバッガに文字列を送信する関数
  - 適当なエラー値をセットしてデバッガに文字列を送信し、エラー値を確認(そのままならばデバッガあり)

→API hookで防衛しましょう

他のデバッガ対策

- わざと0となっている変数除算して例外処理へ
- プロセス毎のPEB構造体のデバッグステータスを確認
- プロセス列挙APIを叩いて一覧からデバッガ探し
- レジスリのAeDebugキーの値がデバッガか?
  - Ae = Application Error
- 実行中のウィンドウのタイトル名がデバッガか?
- 例外から戻る時間などを計測
- デバッガがアタッチされる前に実行されるThread Local Storage領域で実行
  - 普通は使わないので存在していたら怪しいと思え
- single-step exceptionを発生する命令によりシングルステップ実行妨害

仮想マシン対策

- レジストリの中/プロセスリストに仮想マシンの痕跡を探す
- NICのMACアドレスが仮想マシンの物か?
  - VMwareは00:0C:29から開始
- cpuidなどのハードウェア情報にアクセスする命令の返り値における不審
- ホストOSとの通信I/O叩き

デバッガ/仮想マシン対策対策

- プロセスの終了に関するコードへの制御を削除
  - あやしいexitを削除
  - プログラムの最後へのジャンプを削除
  - 自身を削除するルーチンへのジャンプを削除
- 時間計測箇所の間でステップ実行しない
- 例外をトラップしない

参考文献

- Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software
  - 各種デバッガ系ツールの使い方などもあり
  - 演習問題も各種あり
  - http://www.amazon.co.jp/Practical-Malware-Analysis-Hands-Dissecting/dp/1593272901