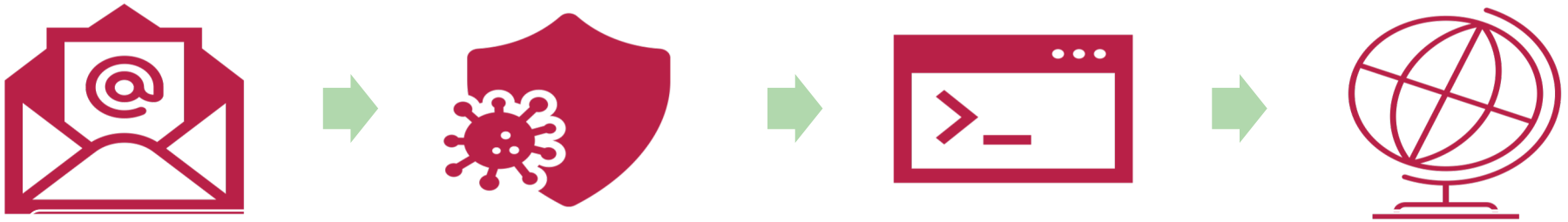
ファイルレス・マルウェアの実態と対策

@Kotokaze__R

ファイルレス・マルウェアとは…

- 名前の通り、ファイルの無いマルウェアのこと
- 厳密には、実行ファイルをストレージ上に書き込みしないマルウェア
- 従来のように、`.exe`ファイルを裏で実行させる必要がないからバレにくい
- マルウェア業界では最近のトレンドとか

攻撃手法は…



スパムメール
フィッシングメール

埋め込まれた
不正コード

Power Shell

ウィルス配布
サイトへ

① インストール用ファイルをDLさせる

- 添付ファイルとしてマクロ付き文書や、`.lnk`ファイルを取り込ませる

② そのファイルを基点に、パワーシェルを介してマルウェア配布サーバに繋ぐ

③ メモリ上にプログラムを展開

④ インストール用ファイルを削除

注)`.lnk` ファイルは、ショートカットに使われるもの

ここが厄介 ファイルレス・マルウェア

- **`.lnk` ファイルを使用している**
  → 本来はショートカットとして使うファイルなので、従来のウィルス対策ソフトでは、監査対象外ファイルに設定されていた
  最近は、対応を進めている？らしいが詳細は不明

- **(実行) ファイルが無い**
  → 従来のシグネチャベース・セキュリティ対策ソフトでは無理
  ※そもそも、実行ファイルが無ければシグネチャなんて無い

- **ビルトイン機能が悪用されている**
  → パワーシェルや、WMI (Windows Management Instrumentation) のような正規ソフトで実行されるため、故意か、ウィルスによるものか判別が困難

- **難読化されたスクリプトが使われている** (場合が多い)

難読化の手法は...

- 文字列の連結
- 無意味な演算の挿入
- 文字列の反転
- 無意味な文字(列)の挿入
- 不要なコメントの挿入
- 値がnullな変数の挿入
- URLの間に、偽のURLを挿入
- エンコーディング
  (Unicode・Bese64・Hex・Dec ...)

これらを自動で行うソフトもネットで出回っている

難読化例：SecCamp Dトラック応募課題

```
${-}="";${-}+=[char](-211+313);${-}+=[char](44577/381);${-}+=[char](584-474);${-}+=[char](-95+194);${-}+=[char](55216/476);${-}+=[char](-707+812);${-}+=[char](1047-936);${-}+=[char](17380
/158);${-}+=[char]([int][Math]::sqrt([Math]::pow(32,2)));${-}+=[char](47034/702);${-}+=[char](-272+376);${-}+=[char](21715/215);${-}+=[char](59+40);${-}+=[char](-102+209);${-}+=[char](482-402);${-}+=
[char](32592/336);${-}+=[char](-18+133);${-}+=[char](876-761);${-}+=[char](-652+771);${-}+=[char](35076/316);${-}+=[char](33516/294);${-}+=[char](96800/968);${-}+=[char](442-402);${-}+=[char]
(-781+817);${-}+=[char](-521+633);${-}+=[char](195-98);${-}+=[char](543-428);${-}+=[char](-586+701);${-}+=[char](90678/762);${-}+=[char](95571/861);${-}+=[char](-202+316);${-}+=[char]
(329-229);${-}+=[char](6068/148);${-}+=[char](86592/704);${-}+=[char](793/61);${-}+=[char](9390/939);${-}+=[char](219-187);${-}+=[char](-665+697);${-}+=[char](11234/137);${-}+=[char](67771
/671);${-}+=[char](251-135);${-}+=[char]([int][Math]::sqrt([Math]::pow(117,2)));${-}+=[char](72390/635);${-}+=[char](-768+878);${-}+=[char](736/23);${-}+=[char](-867+903);${-}+=[char](651-539);${-}+=
[char](44717/461);${-}+=[char](75670/658);${-}+=[char](-860+975);${-}+=[char](365-246);${-}+=[char](69264/624);${-}+=[char](-16+130);${-}+=[char](-737+837);${-}+=[char](824-792);${-}+=[char]
(219-174);${-}+=[char](722-653);${-}+=[char](741-628);${-}+=[char](948-916);${-}+=[char](-530+564);${-}+=[char](204-117);${-}+=[char](17442/342);${-}+=[char](58644/543);${-}+=[char](303-204);${-}+=
[char](-621+669);${-}+=[char](98645/905);${-}+=[char](-905+956);${-}+=[char](519-424);${-}+=[char](73196/631);${-}+=[char](-885+933);${-}+=[char](74+21);${-}+=[char](-875+958);${-}+=[char](49011
/961);${-}+=[char](-387+486);${-}+=[char](501-402);${-}+=[char](1040/20);${-}+=[char](831-722);${-}+=[char](-578+690);${-}+=[char](-454+504);${-}+=[char](-595+643);${-}+=[char](698-648);${-}+=[char]
(-736+784);${-}+=[char](-456+490);${-}+=[char](77-64);${-}+=[char](-261+271);${-}+=[char](-265+390);${-}|iex
```

```
$password=Read-Host -Prompt 'Input the password'
Write-Host "Checking your password..."
Start-Sleep 5
If (CheckPassword($password)){
  Write-Host "The password is correct.`nHere is the flag`n`n"
  Write-Host "+----------------------------+"
  Write-Host "|FLAG{$password}|"
  Write-Host "+----------------------------+`n`n`n"
}Else{
  Write-Host "The password is wrong!"
}
```

解説は次スライドから

応募課題に学ぶ難読化

```
${-} = ""; ${-} += [char](-211 + 313); ${-
```

```
${-}="";
${-}+=[char](-211+313); #f
${-}+=[char](44577/381); #u
${-}+=[char](584-474); #n
${-}+=[char](-95+194); #c
${-}+=[char](55216/476); #t
${-}+=[char](-707+812); #i
${-}+=[char](1047-936); #o
${-}+=[char](17380/158); #n
${-}+=[char]([int][Math]::sqrt([Math]::pow(32,2))); #SPC(スペース)
${-}+=[char](47034/702); #c
${-}+=[char](-272+376); #h
${-}+=[char](21715/215); #e
${-}+=[char](59+40); #c
${-}+=[char](-102+209); #k
```

- `${}` は、変数を表す
  - ここでは `-` (変数ハイフン)
- 一行で書かれていたが、実際は短いコマンドの集合
- 改行を挟んだものが下画像
- `char` は、ASCIIコード → 文字列の変換を行う (コメントアウト参照)
- 最初に生成した変数に、ひと文字ずつインクリメント

コードはこちら → https://github.com/Kotokaze/seccamp/blob/master/seccamp.psl

応募課題に学ぶ難読化 - 2

```
1  function CheckPassword($password){
2    Return $password -Eq "W3lc0m3_t0_S3cc4mp2020"
3  }
```

`${-}` に入っていたもの

```
83  ${-}+=[char](-736+784); #0
84  ${-}+=[char](-456+490); #"
85  ${-}+=[char](77-64); #CR(復帰)
86  ${-}+=[char](-261+271); #LF(改行)
87  ${-}+=[char](-265+390); #}
88  ${-}|iex
```

元コード(抜粋)

- 先ほどの一行に長く書かれていた部分は、実はまとめると三行分でしかない
- `iex` は、`Invoke-Expression`のエイリアス(難読化の常套手段としてよく使われる模様)
- `${-}` は、ただの文字列でしか無いため、`Invoke-Expression` にパイプラインで渡して、式と認識させていた

できる対策

- ウィルス対策ソフトを契約するなら EDR 製品にしよう
  - EDR: Endpoint Detection and Response

    ↑シグネチャではなく、プログラムの動作パターンで脅威度を判定する
- 添付ファイルは、なるべく触らないでおこう
- Office のマクロ実行は、なるべく無効化しておこう
- セキュリティの勉強してるならシェルのログを録って、まめに見よう

結論

- ファイルレス・マルウェアは、実行ファイルがないマルウェア
- インストールは、シェルなどのビルトイン機能を介して行われる
  - 正規機能がゆえに、マルウェアによるアクセスか判別困難
- プログラムはメモリ上に展開されるため、感染しても気づけない
- 難読化されたスクリプトが使用されている場合が多い
- 対策法は確立されていない
  - コマンド実行ログを見る・マクロ実行無効化・添付ファイルに気を付ける

レファレンス・リンク集

- セキュリティ対策ソフトも見つけにくい「ファイルレスマルウェア」とは!? (ASCII.jp)
  - https://ascii.jp/elem/000/002/007/2007121/
- MacAfee Labs 脅威レポート 2017
  - https://www.mcafee.com/enterprise/ja-jp/assets/reports/rp-quarterly-threats-sept-2017.pdf
- セキュリティキャンプ全国大会 Dトラック 応募課題
  - https://www.ipa.go.jp/files/000084566.txt
  - https://gist.githubusercontent.com/Sh1n0g1/e42100f2a8e7d767706b4e2c88a2c45d/raw/b02088c51a4639671bf796d9f60ca56645c35146/obf.ps1