LES FAUX ANTIVIRUS DÉBARQUENT SUR TWITTER

Nicolas Brulez – nicolas.brulez@kaspersky.fr – Senior Malware Researcher
Global Research and Analysis Team – Kaspersky Lab

mots-clés : CODES MALICIEUX / TWITTER / RÉSEAUX SOCIAUX / ANALYSE DE CODE / RSA

Le 20 Janvier 2011, alors que Twitter frôlait les 200 millions d'utilisateurs, une nouvelle vague de liens malicieux est apparue. Avec plus de 110 millions de « tweets » par jour et 370 000 nouveaux inscrits quotidiennement, Twitter est naturellement devenu une cible idéale pour les criminels. Le nombre de caractères limités par ce site (140) pousse de plus en plus d'utilisateurs à se servir d'adresses internet raccourcies, principalement utilisées sur les réseaux sociaux, sans se douter que des liens malveillants peuvent s'y dissimuler. C'est « goo.gl », le service de raccourcissement d'URL de Google, qui fut utilisé pour mener la campagne d'infection [1].

Lors de la découverte de la campagne, un nouveau ver fut suspecté. Il n'en est rien. Il semblerait que les comptes de milliers d'utilisateurs Twitter furent compromis et utilisés pour diffuser les liens.

1 Les tweets frauduleux

Voici à quoi ressemblaient les milliers de *tweets* malveillants postés sur Twitter.

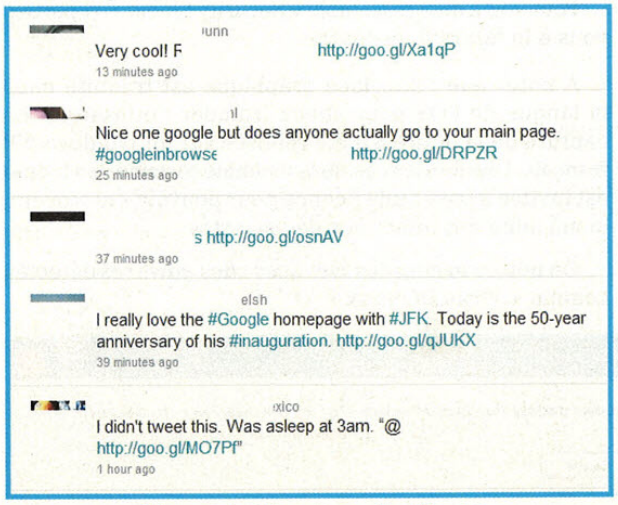

Figure 1 : Tweets malveillants

Plusieurs dizaines de liens raccourcis (goo.gl/XXXX) différents furent employés lors de la propagation des liens frauduleux, dans des tweets anodins aux sujets divers et variés, tels que le 50ème anniversaire du discours d'investiture de John F. Kennedy. Certains utilisateurs se sont rendu compte que des messages Twitter avaient été postés depuis leur compte, à leur insu (voir dernier message de la capture précédente).

Lors de la visite d'un des liens abrégés, l'utilisateur est redirigé à plusieurs reprises avant d'atterrir sur la page du faux antivirus « Security Shield » (Plus connu sous le nom de *Security Tool*).

2 La chaîne de redirections

Tous les liens « goo.gl » redirigeaient les utilisateurs vers une page « m28sx.html » hébergée sur de nombreux noms de domaines différents. En effet, afin que l'attaque soit plus efficace et plus longue à bloquer, une dizaine de sites web compromis ont été utilisés lors des redirections.

```
<HTML>
<HEAD>
<TITLE>Moved Permanently</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000">
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://[redacted]/m28sx.html">here</A>.
</BODY>
</HTML>
```

Figure 2 : Redirection vers page m28sx.html

Cette page HTML se contentait de rediriger les utilisateurs sur un nom de domaine ukrainien statique.

Cette page étant constante, il suffisait de la bloquer pour se protéger de cette attaque :

```
<head>
<meta HTTP-EQUIV="REFRESH" content="0; url=http://[redacted].ua/[redacted].php">
</head>
```

Figure 3 : Redirection vers domaine en Ukraine

En effet, cette page est utilisée comme « dispatcher ». Chaque visite génère des redirections différentes vers des IP connues pour la distribution de faux antivirus :

Name	Value
Status: HTTP/1.1 302 Moved Temporarily	
Date:	Thu, 20 Jan 2011 12:24:43 GMT
Server:	Apache/2
X-Powered-By:	PHP/5.2.17
Location:	http://91.[redacted]/index.php?kK=[redacted]

Figure 4 : Redirection vers la première IP du site de l'antivirus factice

Cette page était le dernier maillon de la chaîne, redirigeant encore une fois sur une adresse IP différente :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="refresh" content="0; url=http://91.[redacted]17/index.php?[redacted]">
</head>
<body></body>
</html>
```

Figure 5 : Redirection vers la seconde IP du site de l'antivirus factice

3 Page de l'antivirus factice

Lors de la visite de cette dernière page, l'utilisateur recevait un message d'avertissement sur l'infection de son ordinateur et l'invitait à scanner sa machine.

Pratique classique des sites web des faux antivirus, voici à quoi ressemblait la page de « scan » :

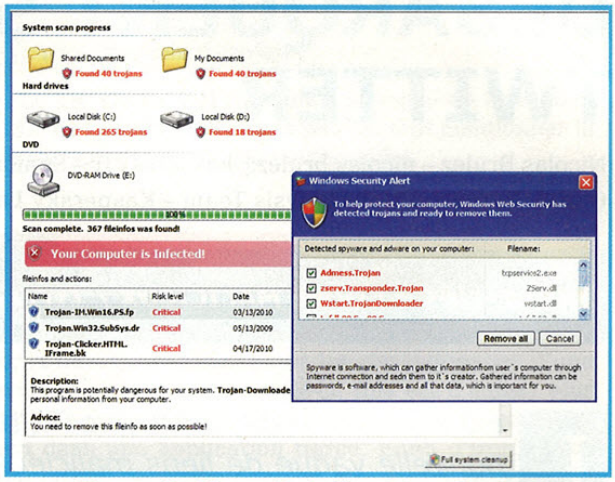

Figure 7 : Site web de l'antivirus factice : infections découvertes

La machine de l'utilisateur est déclarée comme infectée, et il est recommandé de télécharger un antivirus pour la nettoyer. En cliquant sur le bouton ***Remove all***, l'utilisateur est invité à télécharger l'antivirus factice : « Security Shield. »

Ce rogue, plus connu sous le nom de « Security Tool », utilise des techniques de polymorphisme côté serveur.

En effet, à chaque téléchargement, un exécutable différent est généré pour passer à travers les solutions antivirales légitimes. L'utilisation d'un *packer* spécial permet de générer des binaires aléatoirement. Il est important de noter que seule une grande partie de l'exécutable change, pendant plusieurs heures, avant la génération d'une toute nouvelle variante. Chaque exécutable généré utilise des techniques d'anti-émulation de code, principalement basées sur l'appel de fonctions « exotiques » de l'API Windows, avec des paramètres spéciaux.

Pour voir à quoi ressemble « Security Shield », reportez-vous à la figure 8, ci-contre.

À noter que l'interface graphique est traduite dans la langue de l'OS pour mieux tromper l'utilisateur. La capture de la figure 8 a été réalisée sur un Windows XP français. Une fois le scan de la machine terminé, la victime est invitée à payer une licence pour pouvoir « nettoyer » la machine des infections découvertes.

On notera le nom des menaces : des *adwares* détectés comme « Virus.DOS.xxx » :-)

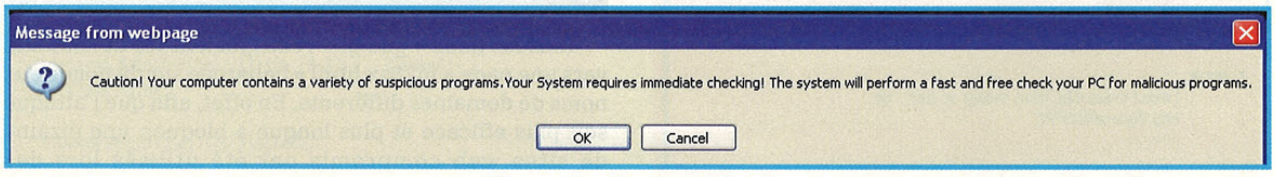

Figure 6 : Alerte d'infection

Figure 8 : Interface de Security Shield en français

4 « Obfuscation » du site factice : RSA et Javascript

Le code source du site web de l'antivirus factice n'est pas disponible en clair. En effet, celui-ci utilise de l'obfuscation de code. L'outil Malzilla n'est pas capable de décrypter la page. Nous allons voir comment il est possible de récupérer une page décryptée. L'intérêt est de pouvoir ensuite programmer un outil automatique de téléchargement des binaires pour détection et classification.

Lors du téléchargement du rogue, un paramètre temporaire est généré par la page web et l'exécutable malicieux n'est jamais téléchargé directement.

Voici à quoi ressemble la page du site :

On remarque la présence de Javascript : **function.js** et **hjosu.js**.

C'est dans ces fichiers que nous allons trouver le code de l'algorithme employé. On remarque aussi la ligne suivante : **a5e.camnqjr(BASE64.decode('Mzxxxxxxxxxxxxxxxx') ;**.

Du Base64 est passé en paramètre à une méthode **camnqjr** de la classe **a5e**.

En examinant les fichiers **.js**, il est possible d'observer ceci :

```
camunqjr: function(c, d, n) {
        var decryptarray = [], deencrypt = '', resultd = '';
        for(var i=0; i<c.length; i+=7) decryptarray.push(c.substr(i, 7));
        for(var u = 0; u < decryptarray.length; u++) {
                if(decryptarray[u]=='')
                        decryptarray.splice(u, 1);
                for(var u = 0; u < decryptarray.length; u++) {
                        var resultmod = this.powmod(parseInt(decryptarray[u], 16), d, n) + '';
                        deencrypt += resultmod.substr(1, resultmod.length - 2);
                }
        }
        for(var u = 0; u < deencrypt.length; u+=2) {
                resultd += this.chr(parseInt(deencrypt.substr(u, 2), 10) + 30);
        }
        return BASE64.decode(resultd);
},

ord: function(chr) {
        return ASCII.ord(chr);
},

chr: function(num) {
        return ASCII.chr(num);
},

mod: function(g, l) {
        return g - (l * Math.floor(g / l));
},

powmod: function(base, exp, modulus) {
        var accum = 1, i = 0, basepow2 = base;
        while((exp >> i) > 0) {
                if (((exp >> i) & 1) == 1) accum = this.mod((accum * basepow2), modulus);
                basepow2 = this.mod((basepow2 * basepow2), modulus);
                i++;
        }
        return accum;
}
```

Figure 10 : Code Javascript pour le décryptage RSA

Toute personne ayant étudié un tout petit peu l'algorithme RSA reconnaîtra les paramètres de la méthode **camunqjr**. On remarque aussi la fonction **powmod**, qui colle bien avec le reste des paramètres :

- **c** pour le *cyphertext* ;
- **d** pour l'exposant privé ;
- **n** pour le modulo.

Pour obtenir le message déchiffré, le déchiffrement se fait comme ceci : **c^d mod N**.

```
#fsc{
background: #ededed url(img/1/1/miscewtn.gif) no-repeat 3px;
position:absolute;
right:5px;
bottom:5px;
padding:2px 2px 2px 20px;
font-size:11px;
}

    </style>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
    <script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
    <script type="text/javascript" src="js/functions.js"></script>
    <script type="text/javascript" src="js/1/hjosu.js"></script>
    <script type="text/javascript">
    var _$e9846 = new Array("muiq","confcm-","drvg","bootrbcp","drvrhg.","drv1fo","msot","drvpbj","winjlb","msif","drvjyk","msipc","
    var a55a = new Array("txt","ini","sql","hxx","tmp","exe","ini","cpl","nls","ram","dic","qlm","vxd","scr","hlp","dll","com","tsp"
    var $1f07c = 1, z5e = 1;
    var zac337 = document;
    (function(){
      var rtext = a5e.camunqjr(BASE64.decode('MzE5MDEyMTB1ZWIzNmIzM2U1YzRjMGNkODhlZjFiOGE3OTYyYzI0MjIyMmJiMjJlZDM2NmYzOTExNTcxZmZkMz
      zac337.write(rtext);
    })();
    </script>
    </head>
    <body></body>
    </html>
```

Figure 9 : Code avec obfuscation

L'utilisation de RSA est détournée de son rôle habituel. En effet, la taille du modulo est ridicule, comme vous pouvez le voir ci-dessous :

```
)AyNzA2ZjI2MmFiNTdiMzI3Njg1OWM='), 28832449, 60145889);
        Base64          'd'    'n'
```

Figure 11 : Paramètres M, D et N

Notre modulo ne fait que 26 bits :

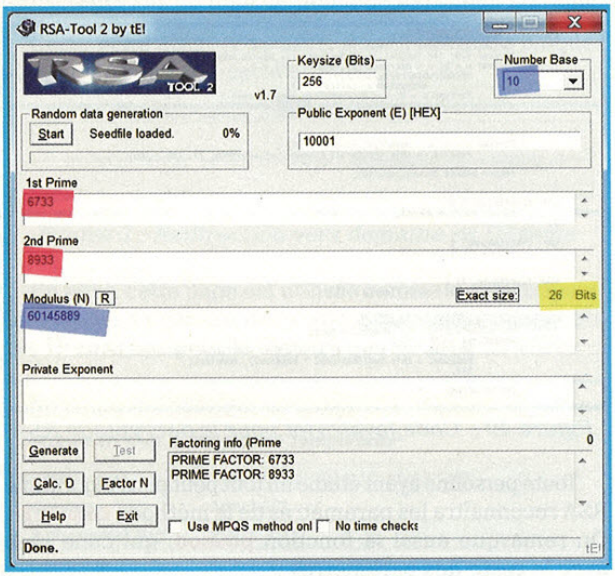

Figure 12 : Taille du modulo : 26 bits

De plus, l'exposant privé est hardcodé dans la page. RSA n'a ici pour but que d'empêcher la lecture du code en clair et d'être un peu différent des algorithmes plus classiques rencontrés sur les sites malicieux.

Toutefois, il est impossible de lire le code en clair avec l'outil Malzilla, par exemple, qui se plaindra d'un code Javascript incorrect. Nous allons voir comment créer une page qui sera interprétée par Malzilla, et donc exécutable.

5 Création d'un template de décryptage Security Shield

La création du *template* est très simple. Il suffit de créer un fichier Javascript simplifié sans utiliser de **classe** ni tableau pour la récupération des valeurs ascii (voir code **chr** original), ce qui semblait troubler l'outil Malzilla.

Après la simple création des fonctions **chr()**, **mod()**, **powmod()** et **decrypt()** dans la même page, suivies d'un appel à la fonction de déchiffrement, il est possible d'obtenir le code décrypté. Voici le code source :

```
function chr(num)
{
  return String.fromCharCode(num);  // Beaucoup plus classique maintenant
}
function mod(g, l) {
  return g - (l * Math.floor(g / l));
}
function powmod(base, exp, modulus) {
    var accum = 1, i = 0, basepow2 = base;
    while((exp >> i) > 0) {
      if (((exp >> i) & 1) == 1) accum = mod((accum * basepow2), modulus);
      basepow2 = mod((basepow2 * basepow2), modulus);
      i++;
    }
    return accum;
}
function decrypt(c, d, n) {
    var decryptarray = [], deencrypt = '', resultd = '';
    for(var i=0; i<c.length; i+=7) decryptarray.push(c.substr(i, 7));
    for(var u = 0; u < decryptarray.length; u++) {
      if(decryptarray[u]=='')
        decryptarray.splice(u, 1);
      for(var u = 0; u < decryptarray.length; u++) {
        var resultmod = powmod(parseInt(decryptarray[u], 16), d, n) + '';
        deencrypt += resultmod.substr(1, resultmod.length - 2);
      }
    }
    for(var u = 0; u < deencrypt.length; u+=2) {
      resultd += chr(parseInt(deencrypt.substr(u, 2), 10) + 30);
    }
    return resultd;
  }
gameover = decrypt('Base64decoded_data', 'paramètre d ici', 'paramètre n ici');
document.write(gameover);
```

Il suffit de copier ce template dans Malzilla, de remplir la partie Base64, les paramètres **d** et **n** et d'exécuter le script pour obtenir le code de la page :

```
<div class="left_box_line doc"><a href="#">My Documents</a></div><div class="left_box_line sd
 Panel</a></div></div><div class="karma">Details</div><div class="left_box"><div class="left_
"><div class="right_hr">System scans progress</div><div class="fb"><div id="tiestol"
"tiest02" class="touch1"></div><div class="f_spark">My Documents</div></div><div class="right
"hdd_icon">Local Disk (C:)</div></div><div class="fb"><div id="tiesto4" class="touch1"></div>
><div class="dvd_icon">DVD-RAM Drive (E:)</div></div><div class="clear"></div><div class="pro
/div></div><div class="art"><span id="elysium">none</span></div><div id="win"><h2>Your <span>
/div><table><tr><td width="166" class="td_cell1">Name</td><td width="105" class="td_cell1">Ri
 infected</td><td width="120" class="td_cell1">State</td></tr><tr class="none" id="dj1""
 class="td_cell2">03/13/2010</td><td class="td_cell2">15</td><td class="td_cell2">Waiting rem
<td class="td_cell2 stat">Critical</td><td class="td_cell2">05/13/2009</td><td class="td_cell
"td_cell2 name">Trojan-Clicker.HTML.IFrame.bk</td><td class="td_cell2 stat">Critical</td><td
 removal</td></tr></table><div class="none2" id="desc"><b>Description:</b><br/>This program <
> <span>stealing</span> passwords , credit <span>cards and</span> other personal  information
;fileinfo as soon as possible!</div><button id="fsc">Full <span>system</span> cleanup</button
 computer, Windows Web Security has detected trojans and ready to remove them.</p><span class
 class="b"><button>Remove all</button><button>Cancel</button></div><p class="f">Spyware is so
 them to it's creator. Gathered information can be passwords, e-mail addresses and all that d
.preventDefault();prdl = true;getW().location = "77b841J3-Lfqc";prdl = false;return false;});"
;if (left_bar_height < right_bar_height) {$(".left_bar").height(right_bar_height);}$(document
);$("#white").css("display", "none");$("#page_progress").css("display", "none");$(".left_bar"
 process.progress1 + process.progress2 + process.progress3 + process.progress4; $("#elysium")
```

Figure 13 : Code source après déchiffrement

En jaune, vous pouvez voir le paramètre aléatoire utilisé lors du téléchargement de l'antivirus factice. Il suffit maintenant de reprogrammer les quelques lignes de Javascript dans le langage de votre choix (Python ? 2ème dédicace à Phil) pour automatiser le déchiffrement des pages de *Security Shield*, et de programmer un *crawler* de *samples*.

Game Over ■

■ RÉFÉRENCES

[1] ***Goo.gl redirects Fake AntiVirus*** : http://www.zdnet.com/blog/security/twitter-worm-hits-googl-redirects-to-fake-anti-virus/7938

[2] **Malzilla** : http://malzilla.sourceforge.net/