EXATEL

„BARDZO SIĘ ZDZIWIĘ, JEŚLI TO WYJDZIE NA JAW”

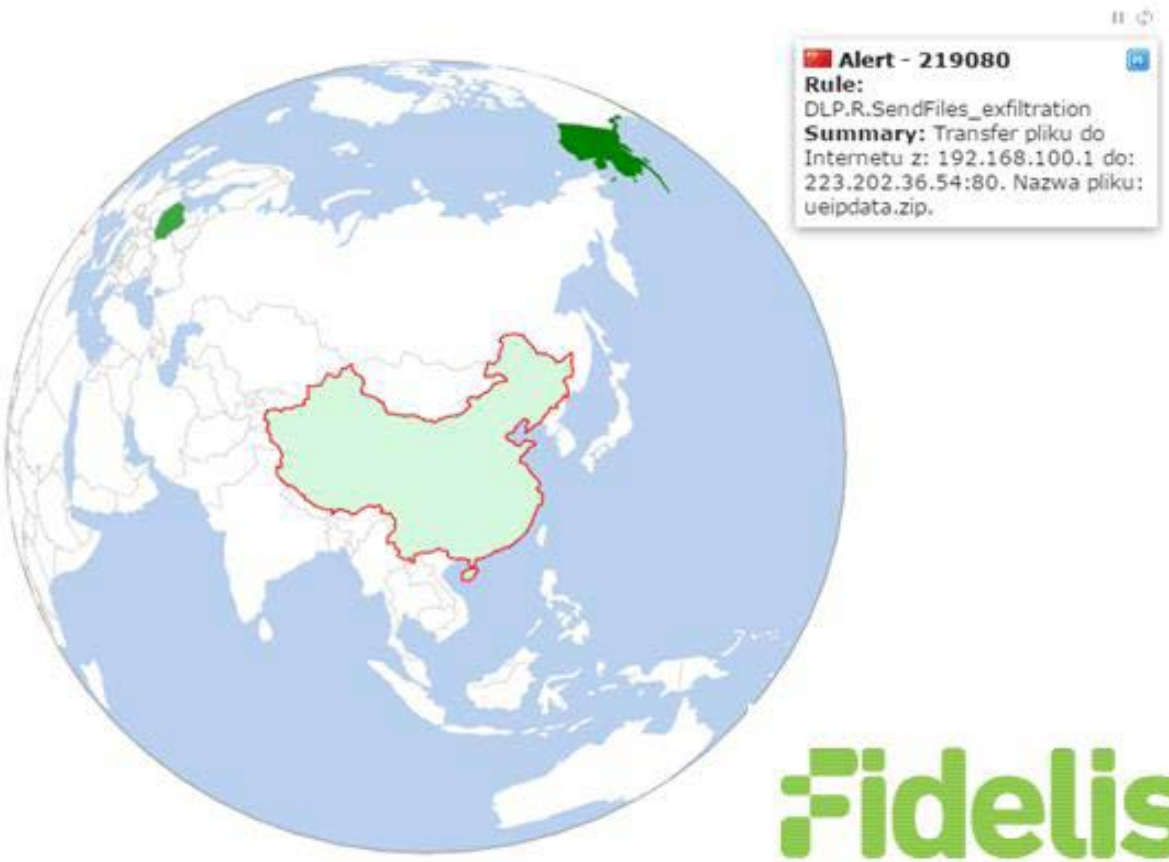
Raport

Centrum Operacyjne Bezpieczeństwa Cybernetycznego, Exatel

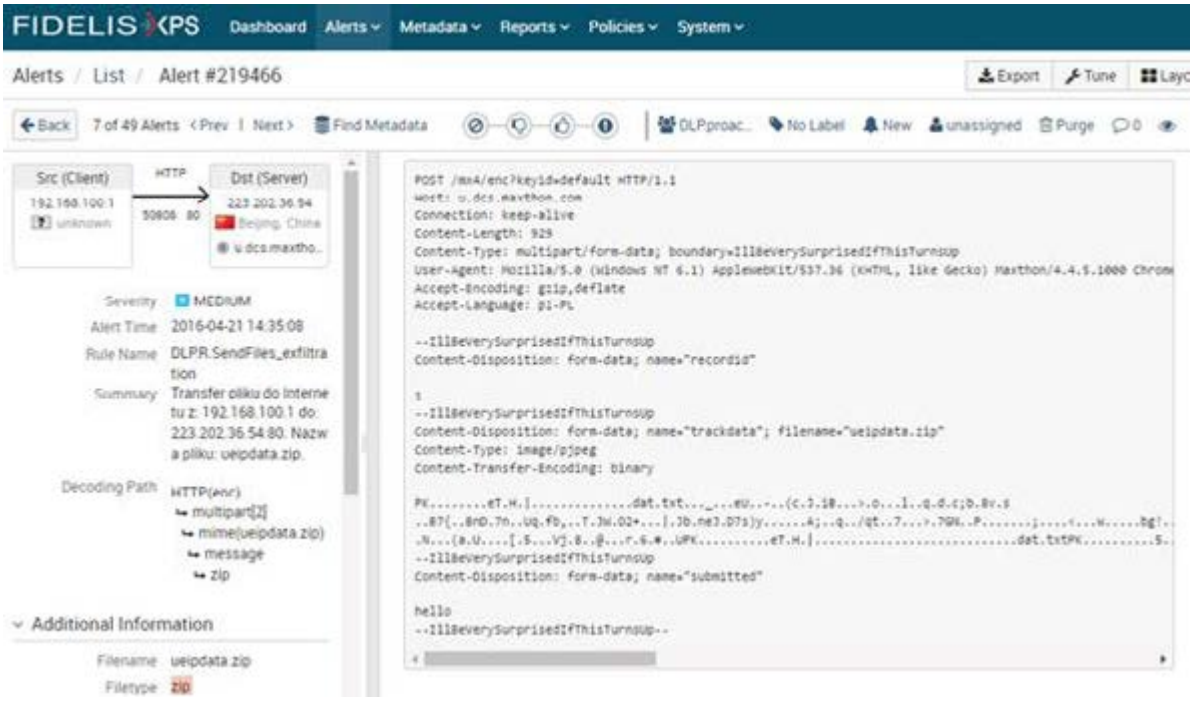
Poniższy tekst przedstawia wyniki raportu z analizy technicznej, przeprowadzonej przez analityków trzeciej linii Centrum Operacyjnego Bezpieczeństwa Cybernetycznego (SOC) Exatel S.A. Raport ten powstał na bazie incydentu, jaki pod koniec marca zidentyfikowała grupa reagowania na incydenty, działająca w ramach SOC Exatel, w trakcie wdrażania zaawansowanego systemu detekcji zagrożeń Fidelis w jednej z lokalizacji.

Dzięki informacjom pozyskanym w trakcie analizy przeprowadzonej z użyciem inżynierii odwrotnej kodu, grupie reagowania na incydenty SOC Exatel udało się dotrzeć do funkcjonalności, którą autorzy dość popularnego narzędzia próbowali zaszyć w oprogramowaniu, celem przesyłania do swoich serwerów treści, o której przesyłaniu użytkownik nie tylko nie został poinformowany przez producenta, ale i został mylnie przeświadczony, że tego typu treść nie opuści jego komputera jeśli nie da na to wyraźnej zgody.

Wkrótce po podłączeniu do monitorowania przez system Fidelis wewnętrznej sieci LAN organizacji, grupa reagowania na incydenty z SOC Exatel rozpoczęła rejestrować od kilkunastu do kilkudziesięciu razy dziennie alarm o naruszeniu reguły `DLP.sendfiles.exfiltration`, którą specjaliści z Exatel wbudowali do tego systemu w celu monitorowania, czy dokumenty (w ogólności szerokorozumiane dane) nie są przesyłane na zewnątrz sieci przy pomocy protokołu HTTP i metody POST. W taki właśnie sposób przeglądarki internetowe przesyłają różnego typu dane do zdalnych serwerów, w tym np. pliki załączników do wiadomości, wysyłanych z użyciem poczty webmail’owej. Okazało się, że z 3 komputerów znajdujących się w wewnętrznej sieci korporacyjnej, tym właśnie protokołem, regularnie, wysyłany jest do serwera w Pekinie, niewielki, kilkusetbajtowy plik o nazwie `ueipdata.zip`



System Fidelis, wdrożony w Exatel, posiada moduł pamięci sieciowej gromadzącej nie tylko metadane, ale też szczegóły transmisji, która w jakikolwiek sposób narusza polityki bezpieczeństwa. Jest on w stanie zapamiętać i analizować w głąb - z użyciem DPI (Deep Packet Inspection) - zarówno protokoły komunikacyjne, jak i zróżnicowane metody kodowania zagnieżdżonych w nich plików. Dzięki temu eksperci od bezpieczeństwa z Exatel dysponują pełną wiedzą o incydentach.



Powyższy zrzut ekranu z konsoli monitoringu zdarzeń systemu Fidelis przedstawia szczegóły dotyczące pojedynczego alarmu, wygenerowanego na skutek naruszenia wspomnianej wcześniej reguły, opisującego potencjalną eksfiltrację danych do serwera w Chinach.

To, co rzuciło się najpierw w oczy specjalistom z SOC to fakt, że przesyłany plik `ueipdata.zip` zawiera spakowany pojedynczy plik `dat.txt`, który w rzeczywistości nie jest plikiem tekstowym, lecz zawiera dane o dużej entropii, będące albo wyjściem z generatora losowego, albo wynikiem szyfrowania. Ponadto typ przesyłanego pliku - identyfikowany polem `content-type` protokołu HTTP – został oznaczony jako `image/pjpeg`, czyli... obrazek:

Jednak najbardziej zadziwiała fraza, pojawiająca się kilkukrotnie w treści wysyłanego pakietu HTTP, zawierająca łańcuch tekstowy „`IllBeVerySurprisedIfThisTurnsUp`.

Pierwszym, i chyba najbardziej oczywistym w tych okolicznościach, podświadomym tłumaczeniem frazy było: **„bardzo się zdziwię, jeśli to wyjdzie na jaw”**. A zważywszy na fakt, że zbliżał się akurat 1 kwietnia, eksperci z SOC Exatel początkowo pomyśleli, że być może któryś z ich kolegów testuje, czy nowo nowo zainstalowany w sieci system Fidelis będzie w stanie takie zdarzenie wykryć.

Okazało się to jednak błędnym tłumaczeniem tej frazy.

Dalsza analiza zbieżności nazwy docelowego serwera w Chinach i identyfikatora `user-agent`, zapamiętanego przez Fidelis (identyfikatora, którym identyfikuje się zwykle klient HTTP) umożliwiła ekspertom z Exatel dotarcie do prawdziwego winowajcy (i właściwego tłumaczenia tej frazy).

Winowajcą, stojącym za alarmami systemu Fidelis, okazała się przeglądarka internetowa Maxthon, stworzona i rozwijana przez Chińczyków.

Zgodnie z serwisem **StatsMonkey** w 2014 zajmuje ona szóstą pozycję pod względem popularności, zarówno w Polsce, jak i w Chinach.

Rank	Poland	Market Share
1	Chrome	47.44
2	Firefox	35.83
3	Opera	7.8
4	IE	7.27
5	Safari	1.02
6	Maxthon	0.32

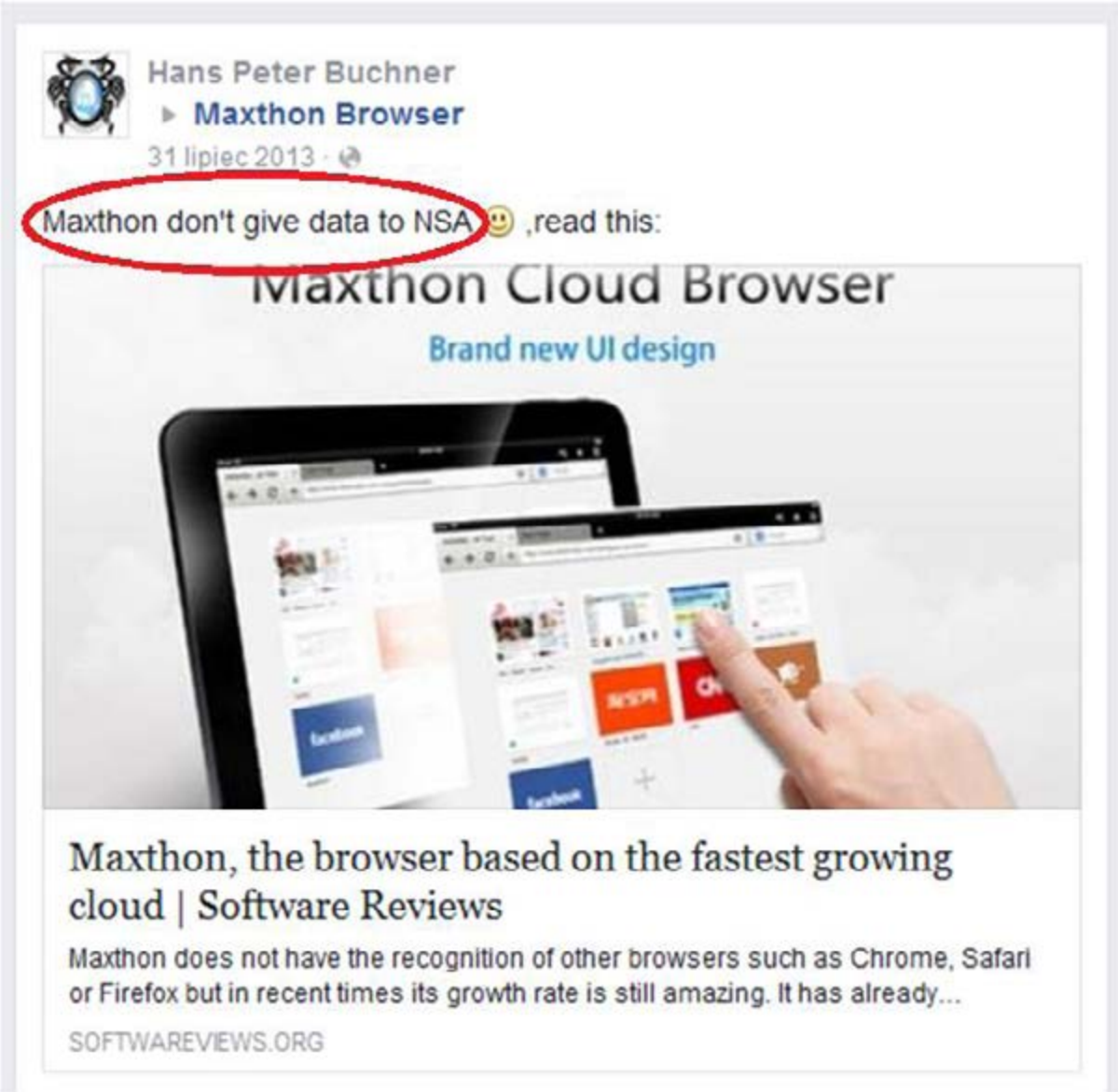
Rank	China	Market Share
1	Chrome	49.51
2	IE	28
3	Sogou Explorer	8.67
4	QQ Browser	4.56
5	Firefox	4.34
6	Maxthon	2.59

StatsMonkey, 2014

To przeglądarka Maxthon, zainstalowana na komputerach 3 pracowników firmy, wysyłała pliki, które zauważył system Fidelis. Co dodaje ironii całej sprawie, autorzy przeglądarki informują na swojej stronie, że jest ona stworzona z myślą o zapewnieniu bezpieczeństwa i prywatności użytkowników, w świetle skandali z łamaniem prywatności przez amerykańską agencję NSA:

http://www.maxthon.com/blog/rightstarups-cloud-browser-with-muscle-security-startup-maxthon-caters-to-html5-users/

Jak można przeczytać w opiniach na temat Maxthona, również użytkownicy darzą szczególną sympatią tę przeglądarkę, ze względu na fakt, iż... jej autorzy nie dzielą się danymi z amerykańską agencją wywiadu elektronicznego NSA:



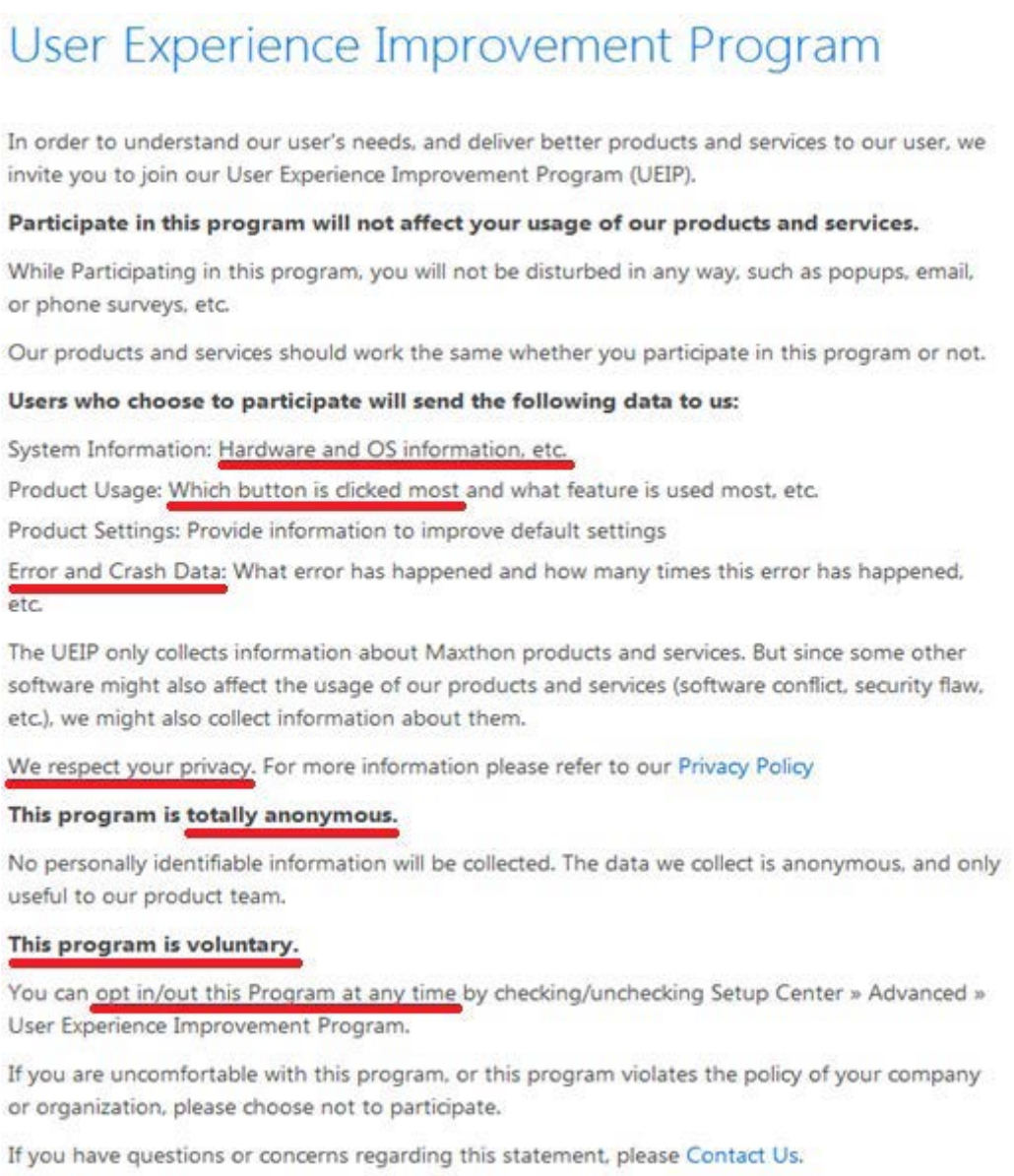
Wracając do łańcucha tekstowego „IllBeVerySurprisedIfThisTurnsUp", który nakierował uwagę ekspertów z SOC, jego pojawienie się w transmisji było wynikiem zarówno zbiegu okoliczności, jak i poczucia humoru jednego z chińskich programistów. Użył on takiego właśnie statycznego łańcucha tekstowego w kodzie biblioteki C++ (opartej o framework MFC), by separować nim pliki zagnieżdżone w transmisji HTTP (w naszym przypadku instruując serwer Maxthona jak zdekodować plik ZIP w pakiecie HTTP (przekazując string zakresu o takiej właśnie wartości w nagłówku `HTTP`, w zmiennej `boundary` pola `content-type`).

Napisana przez niego jeszcze w 2007 roku biblioteka implementująca klienta protokołu HTTP:

```
261    void* pBuffer;
262    LPSTR szResponse;
263    CString strResponse;
264    BOOL bSuccess = TRUE;
265
266    CString strDebugMessage;
267
268    if (FALSE == fTrack.Open(_mFilePath, CFile::modeRead | CFile::shareDenyWrite))
269    {
270      AfxMessageBox(_T("Unable to open the file."));
271      return FALSE;
272    }
273
274    int iRecordID = 1;
275    strHTTPBoundary = _T("IllBeVerySurprisedIfThisTurnsUp");
276    strPreFileData = MakePreFileData(strHTTPBoundary, pcmname, iRecordID);
277    strPostFileData = MakePostFileData(strHTTPBoundary);
278
279    AfxMessageBox(strPreFileData);
280    AfxMessageBox(strPostFileData);
281
282    dwTotalRequestLength = strPreFileData.GetLength() + strPostFileData.GetLength() + fTrack.
GetLength();
283
284    dwChunkLength = 64 * 1024;
285
286    pBuffer = malloc(dwChunkLength);
287
288    if (NULL == pBuffer)
```

została użyta przez twórców Maxthon do stworzenia części funkcjonalności przeglądarki, (funkcjonalności o której napiszemy dalej), a prawidłowe tłumaczenie wspomnianej frazy powinno brzmieć „bardzo się zdziwię, jeśli ta sekwencja znaków pojawi się gdzieś w załączonym pliku, przesyłanym przez program”.

No dobrze... ale zatrzymaliśmy się na pliku `ueipdata.zip`, który w dziwnych okolicznościach (i formie) opuszcza komputery, na których zainstalowano przeglądarkę. Otóż po krótkim dochodzeniu udało się rozszyfrować skrót UEIP jako „User Experience Improvement Program”. Taką nazwę nosi program, który - jak zapewniają na stronie przeglądarki autorzy - jest dobrowolny i anonimowy, a jego celem jest pomoc autorom w ulepszaniu przeglądarki poprzez dzielenie się informacjami na temat sprzętu, na którym zainstalowano przeglądarkę oraz danymi dotyczącymi systemu operacyjnego oraz ew. błędów w trakcie działania przeglądarki.



User Experience Improvement Program

In order to understand our user's needs, and deliver better products and services to our user, we invite you to join our User Experience Improvement Program (UEIP).

Participate in this program will not affect your usage of our products and services.

While Participating in this program, you will not be disturbed in any way, such as popups, email, or phone surveys, etc.

Our products and services should work the same whether you participate in this program or not.

Users who choose to participate will send the following data to us:

System Information: Hardware and OS information, etc.

Product Usage: Which button is clicked most and what feature is used most, etc.

Product Settings: Provide information to improve default settings

Error and Crash Data: What error has happened and how many times this error has happened, etc.

The UEIP only collects information about Maxthon products and services. But since some other software might also affect the usage of our products and services (software conflict, security flaw, etc.), we might also collect information about them.

We respect your privacy. For more information please refer to our Privacy Policy

This program is totally anonymous.

No personally identifiable information will be collected. The data we collect is anonymous, and only useful to our product team.

This program is voluntary.

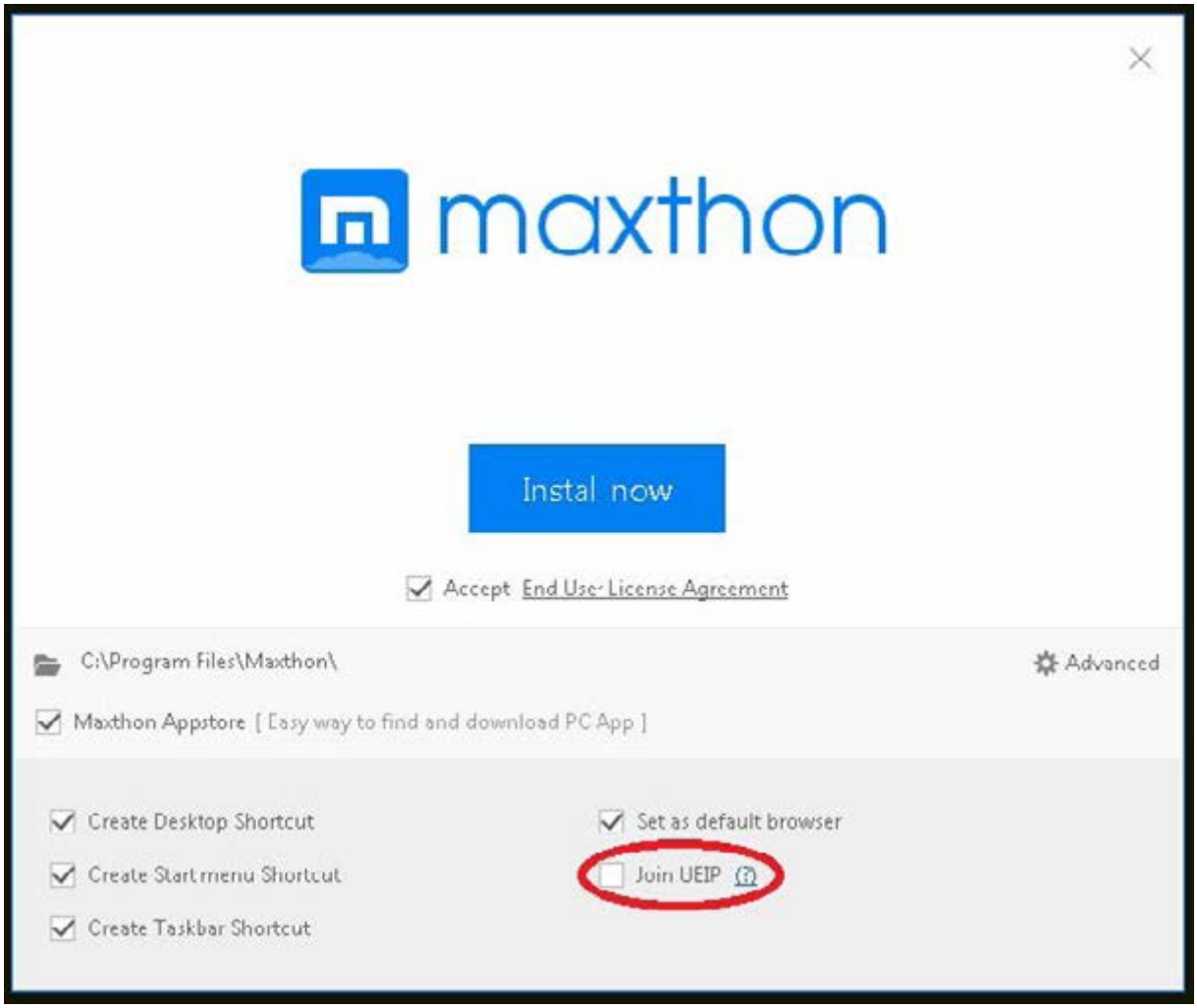
You can opt in/out this Program at any time by checking/unchecking Setup Center » Advanced » User Experience Improvement Program.

If you are uncomfortable with this program, or this program violates the policy of your company or organization, please choose not to participate.

If you have questions or concerns regarding this statement, please Contact Us.

Jak też zapewniają jej autorzy, z programu UEIP można w każdym momencie zrezygnować, a „prywatność użytkownika jest respektowana”.

Eksperci od bezpieczeństwa z Exatel postanowili więc to sprawdzić. Na swojej testowej maszynie zainstalowali przeglądarkę Maxthon, upewniając się, że na ekranie startowym instalatora odznaczyli opcję uczestniczenia w programie UEIP:



Efekt? Niestety brak.

Nasłuch ruchu TCP na interfejsie sieciowym maszyny w trakcie użytkowania przeglądarki, pokazał regularną komunikację z tym samym serwerem Maxthon (`u.dcs.maxthon.com`), zawierającą w swojej treści plik `ueipdata.zip`.

Specjalistów z SOC zaintrygowało kilka kwestii.

Po pierwsze: dlaczego dane programu UEIP - pomimo wyraźnego braku zgody użytkownika - są wysyłane do producenta Maxthon? Po drugie: dlaczego plik `ueipdata.zip`, który zawiera pozornie tekstowy plik `out.txt` jest wysyłany dalej, pod postacią pliku zdjęciowego?

I po trzecie: co właściwie przekazuje do serwerów Maxthon przeglądarka w pliku ZIP?

Eksperci od bezpieczeństwa z Exatel postanowili zbadać całą sprawę. W tym celu zlokalizowali miejsce w kodzie procesu głównego przeglądarki Maxthon, w którym wydawany jest rozkaz zaszyfrowania danych (dane te są zapisywane w pliku `out.txt`, spakowane do pliku `ueipdata.zip` i wysłane do serwera Maxthon). Jak szybko zauważyli, dane są szyfrowane symetrycznym algorytmem Rijndael (AES), z użyciem stałego, statycznie wkompilowanego w kod przeglądarki, 16 bajtowego klucza `eu3o4[r04cml4eir` bez użycia żadnej obfuskacji.

```
.rdata:48016520 ; const WCHAR ClassName
.rdata:48016520 ClassName:                              ; DATA XREF: sub_48010190+40To
.rdata:48016520                                         ; sub_48010190+6BTo
.rdata:48016520                 unicode 0, <Maxthon3Cls_Ueip>,0
.rdata:48016542                 align 4
.rdata:48016544 aEu3o4R04cml4ei db 'eu3o4[r04cml4eir',0 ; DATA XREF: sub_4800FE20+A7To
.rdata:48016544                                         ; sub_480105D0+31To
.rdata:48016555                 align 4
.rdata:48016558 aMxencode_dll:                          ; DATA XREF: sub_4800FE20+CFTo
.rdata:48016558                                         ; sub_480105D0+90To
.rdata:48016558                 unicode 0, <MxEncode.dll>,0
.rdata:48016572                 align 4
.rdata:48016574 aBrowser_svc_ue:                        ; DATA XREF: sub_480105D0+293To
.rdata:48016574                 unicode 0, <browser_svc_ueip_def>,0
.rdata:4801659E                 align 10h
```

Klucz, wraz z buforem danych do zaszyfrowania i jego wielkością (zaraz przed zbudowaniem nowego pliku `ueipdata.zip` i wysłaniem go do serwera Maxthon) jest przekazywany do funkcji `Encode`, znajdującej się w pliku biblioteki dynamicznej Maxthona o nazwie `MxEncode.dll`, odpowiedzialnej za szyfrowanie danych UEIP przesyłanych między przeglądarką, a zdalnym serwerem Maxthon i zawartych w plikach ZIP.

```
4800FEC5 loc_4800FEC5:                                  ; CODE XREF: sub_4800FE20+7Fj
4800FEC5                push    10h             ; Size
4800FEC7                push    offset aEu3o4R04cml4ei ; "eu3o4[r04cml4eir"
4800FECC                lea     ecx, [ebp+var_28] ; int
4800FECF                mov     [ebp+var_14], 0Fh
4800FED6                mov     [ebp+var_18], 0
4800FEDD                mov     byte ptr [ebp+var_28], 0
4800FEE1                call    sub_480022F0
4800FEE6                sub     esp, 18h
4800FEE9                mov     byte ptr [ebp+var_4], 1
4800FEED                mov     ecx, esp
4800FEEF                push    offset aMxencode_dll ; "MxEncode.dll"
4800FEF4                call    sub_48002130
4800FEF9                call    sub_48010210
4800FEFE                mov     ecx, eax
```

Do stworzenia biblioteki MxEncode użyto biblioteki kryptograficznej `Crypto++`, co można zauważyć, analizując tablicę symboli pliku wykonywalnego.

```
AVlogic _ error@std@@
AVlength _ error@std@@
AVout _ of _ range@std@@
AVtype _ info@@
AVbad _ exception@std@@
AV?$BlockCipherFinal@$0A@VEnc@Rijndael@CryptoPP
AV?$BlockCipherImpl@URijndael _ Info@CryptoPP@@VBlockCipher@2
AVexception@std@@
AV?$FixedBlockSize@$0BA@@CryptoPP@@
AVEnc@Rijndael@CryptoPP@@
AV _ Iostream _ error _ category@std@@
AV _ Generic _ error _ category@std@@
AURijndael _ Info@CryptoPP@@
AVNotImplemented@CryptoPP@@
AVAlgorithm@CryptoPP@@
AVDec@Rijndael@CryptoPP@@
AV?$TwoBases@VBlockCipher@CryptoPP@@URijndael _ Info@2@@CryptoPP@@
AV?$BlockCipherFinal@$00VDec@Rijndael@CryptoPP@@@CryptoPP@@
AVNameValuePairs@CryptoPP@@
```

```
AVNullNameValuePairs@CryptoPP@@
AVInvalidKeyLength@CryptoPP@@
AVInvalidArgument@CryptoPP@@
AVbad_alloc@st
```

Dalsza analiza pokazała, że biblioteka MxEncode jest również odpowiedzialna za szyfrowanie i deszyfrowanie lokalnych plików konfiguracyjnych Maxthon, znajdujących się na dysku użytkownika, których to treść producent przeglądarki chroni przed możliwością swobodnego podejrzenia.

W związku z powyższym eksperci z SOC Exatel postanowili założyć nasłuch na komunikacji między przeglądarką Maxthon, a jej modułem szyfrującym `MxEncode.dll` i przeprowadzić atak Man-In-The-Middle na bibliotekę szyfrującą Maxthon.

Wykorzystali fakt, że w klasycznym przypadku przeglądarka - chcąc wysłać zaszyfrowane dane UEIP do serwera w Chinach - załaduje najpierw znajdującą się w swoim instalacyjnym katalogu bibliotekę `MxEncode.dll`, wyśle do niej dane do zaszyfrowania wraz z kluczem szyfrującym, wywołując jej funkcję eksportową `Encode`, a biblioteka - po zaszyfrowaniu danych - zwróci zaszyfrowany bufor wyjściowy do procesu Maxthon, który następnie już zaszyfrowane dane wyśle.

Eksperci z SOC stworzyli więc własną biblitekę DLL, imitującą oryginalną bibliotekę MxEncode, umieszczając w niej tak samo jak w oryginale dwie funkcje eksportowe: szyfrującą Encode i deszyfrującą `Decode`.

```
#include <stdio.h>
#include <windows.h>
extern „C” {
char mxEncodeDLLFile[] = „MxEncodeOrig.dll”;
char encFile[] = „enc.dat”;
char decFile[] = „dec.dat”;
typedefint (*MxDecodePtr)(char *outBuf, char *inBuf,
      intbufSize, unsigned char *key);
typedefint (*MxEncodePtr)(char *outBuf, char *inBuf,
      intbufSize, unsigned char *key);
__declspec(dllexport) intMxEncode(char *outBuf,
      char *inBuf, intbufSize,
      unsigned char *key)
{
      HMODULE lib = LoadLibrary(mxEncodeDLLFile);
      void *ptr = GetProcAddress(lib, „MxEncode”);
      MxEncodePtrMxEncode = (MxEncodePtr) ptr;
      FILE *f=fopen(encFile, „ab”);
      fprintf(f, „[ENC.KEY] %s\r\n”, key);
      fprintf(f, „[ENC.SIZ] %d\r\n”, bufSize);
      fprintf(f, „[ENC.BUF] „);
      fwrite(inBuf, 1, bufSize, f);
      fprintf(f, „\r\n”);
      fclose(f);
      return MxEncode(outBuf, inBuf,
```

```
            bufSize, key);
}
__declspec(dllexport) intMxDecode(char *outBuf,
      char *inBuf, intbufSize, unsigned char *key)
{
      HMODULE lib = LoadLibrary(mxEncodeDLLFile);
      void *ptr = GetProcAddress(lib, „MxDecode”);
      MxDecodePtrMxDecode = (MxDecodePtr) ptr;
      int ret = MxDecode(outBuf, inBuf,
            bufSize, key);
      FILE *f=fopen(decFile, „ab”);
      fprintf(f, „[DEC.KEY] %s\r\n”, key);
      fprintf(f, „[DEC.SIZ] %d\r\n”, bufSize);
      fprintf(f, „[DEC.BUF] „);
      fwrite(outBuf, 1, bufSize, f);
      fprintf(f, „\r\n”);
      fclose(f);
      return ret;
}
BOOL APIENTRY DllMain(HINSTANCE hModule,
      DWORD ul _ reason _ for _ call,
      LPVOID lpReserved)
{
      return TRUE;
}
}
```

W obu funkcjach umieścili kod zapisujący na dysku dane (do wyznaczonego przez nich pliku), których zaszyfrowania w dowolnym momencie przeglądarka może od swojej biblioteki zażądać. Po otrzymaniu żądania zaszyfrowania (i zapisaniu danych na dysk), biblioteka podstawiona przez expertów Exatel powinna załadować właściwą bibliotekę szyfrującą Maxthona, wywołując właściwą funkcję szyfrującą, a zaszyfrowane dane zwrócić do przeglądarki Maxthon, która następnie dane prześle do serwera Maxthon.

W ten sposób pozwolili oni Maxthonowi przepuścić przez swoją bibliotekę całość danych, których zaszyfrowania zażąda przeglądarka przed ich wysłaniem do Chin. Specjaliści z SOC w Exatel otrzymali tą metodą nie tylko całą zdeszyfrowaną już transmisję UEIP do serwerów w Pekinie, ale również dodatkowo pozwolili Maxthonowi rozszyfrować pliki konfiguracyjne, przechwytując do tego klucze deszyfrujące i dane zwracane przez funkcję `Decode` oryginalnej biblioteki `MxEncode`.

Następnie uruchomiono przeglądarkę, aby sprawdzić efekt.

Zaraz po uruchomieniu Maxthona, załadował on bibliotekę MxEncode i zażądał zaszyfrowania pierwszych danych przed ich wysłaniem, przekazując ekspertom z Exatel klucz szyfrujący, pozyskany w trakcie wcześniejszej analizy z wykorzystaniem inżynierii odwrotnej.

```
[ENC.KEY] eu3o4[r04cml4eir
[ENC.SIZ] 384
[ENC.BUF] {"uid":"0","1":"en-us","sv":"5.2.3790.Service Pack
2","cv":"4.9.2.1000","pn":"max4web","d":"ADF17F6774C613DDFBB47FF40A96556B27740000
","ueip":0,"screen":"1020x608","net":"","hd":"{\"cpu\":[{\"name\":\"Intel(R)
Core(TM)2 Duo CPU     T7300  @ 2.00GHz\",\"maxclockspeed\":\"2000\"}],\"mem\":\"5
36248320\"}","db":"C:\\Program Files\\Maxthon\\Bin\\Maxthon.exe","a":1,"cmd":""}

[ENC.KEY] eu3o4[r04cml4eir
[ENC.SIZ] 5072
[ENC.BUF] {"uid":"0","1":"en-us","sv":"5.2.3790.Service Pack
2","cv":"4.9.2.1000","pn":"max4web","d":"ADF17F6774C613DDFBB47FF40A96556B27740000
"}
{"pt":"adblock","dt":"users","dr":"1460930014000:1460930014000","m":"blockednum",
"n":"0","o":"","p":"","data":""}
{"pt":"adblock","dt":"users","dr":"1460930014000:1460930014000","m":"adblockenabl
ed","n":"no","o":"","p":"","data":""}
{"pt":"settings","dt":"users","dr":"1460930016000:1460930016000","m":"startwith",
"n":"home","o":"","p":"","data":""}
{"pt":"settings","dt":"users","dr":"1460930016000:1460930016000","m":"homepage","
n":"","o":"","p":"","data":"{\"url\":\"[\\\"http:\\\/\\\/i.maxthon.com\\\/initial
_configuration.htm\\\"]\"}"}
```

Jak widać, zaszyfrowane i wysłane do serwera zostały: wersja Service Pack systemu Windows, wersja przeglądarki Maxthon, rozdzielczość ekranu wirtualnej maszyny, typ i częstotliwość procesora oraz ścieżka w jakiej zainstalowano Maxthona na dysku. Wysłane również zostały wartości zmiennych konfiguracyjnych: czy włączono `adblock`, liczbę już zablokowanych reklam oraz adres WWW ustawionej strony startowej.

Można uznać, iż powyższe dane zgadzają się z listą informacji o których wysyłaniu autorzy piszą w opisie programu UEIP (abstrahując od faktu iż użytkownik nie wyraził zgody na przystąpienie do programu).

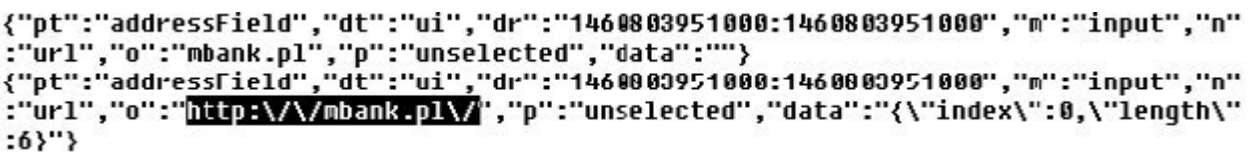
Następnie specjaliści SOC Exatel skupili się na tym jakie operacje szyfrowania z użyciem biblioteki MxEncode wykonuje przeglądarka Maxthon w momencie otwarcia nowej strony. Po wejściu na pierwszą stronę (akurat Onet) okazało się, że... fakt wejścia na tę stronę został również odnotowany i przekazany do serwera Maxthon.

Podobnie jak informacja o każdej, kolejno odwiedzanej stronie.

Logowanie do poczty:

Odwiedziny na stronie sejmu:

Wejście na stronę banku:



Tak więc wszystkie zapytania metodą GET protokołu HTTP były przesyłane do serwera Maxthon.

Co to oznacza w skrócie?

Cała historia przeglądania użytkownika trafia do serwera autorów w Pekinie, wraz ze wszystkimi wpisywanymi zapytaniami w Google.

Kontynuując przeglądanie Internetu z użyciem Maxthona „na podsłuchu", eksperci z Exatel zauważyli, że raz na około 5 wysłanych plików `ueipdata.zip`, przekazywana jest również pełna lista oprogramowania zainstalowanego na komputerze, wraz z dokładnymi numerami wersji.

{"pt":"basicInfo","dt":"content","dr":"1467801694000:1467801694000","m":"software list","n":"","o":"","p":"","data":"{\"softwarelist\":\"7-Zip%2015.12:15.12,Maxthon%20Cloud%20Browser:4.9.3.1000,Microsoft%20Visual%20Studio%202010%20Tools%20for%20Office%20Runtime%20(x86):10.0.50903,Mozilla%20Firefox%2043.0.4%20(x86%20pl):43.0.4,Mozilla%20Maintenance%20Service:43.0.4,Notepad%2B%2B:6.9,Microsoft%20Office%20Professional%20Plus%202013:15.0.4569.1506,WinISO:6.4.0.5170,WinRAR%205.31%20(32-bit):5.31.0,VMware%20Tools:10.0.5.3228253,Python%202.7.11:2.7.11150,Microsoft%20Visual%20C%2B%2B%202008%20Redistributable%20-%20x86%209.0.30729.4148:9.0.30729.4148,Microsoft%20.NET%20Framework%204.6.1:4.6.01055,Microsoft%20Visual%20Studio%202010%20Tools%20for%20Office%20Runtime%20(x86):10.0.50908,Microsoft%20Office%20Professional%20Plus%202013:15.0.4569.1506,Microsoft%20Access%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Excel%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20PowerPoint%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Publisher%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Outlook%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Word%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Office%20Korrekturhilfen%202013%20-%20Deutsch:15.0.4569.1506,Microsoft%20Office%20Proofing%20Tools%202013%20-%20English:15.0.4569.1506,Narz%C4%99dzia%20sprawdzaj%C4%85ce%20pakietu%20Microsoft%20Office%202013%20%E2%80%94%20polski:15.0.4569.1506,Microsoft%20Office%20Proofing%20(Polish)%202013:15.0.4569.1506,Microsoft%20InfoPath%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Office%20Shared%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20DCF%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20OneNote%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Groove%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Office%20OSM%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Office%20OSM%20UX%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20Lync%20MUI%20(Polish)%202013:15.0.4569.1506,Microsoft%20.NET%20Framework%204.6.1:4.6.01055,Microsoft%20Visual%20C%2B%2B%202008%20Redistributable%20-%20x86%209.0.30729.6161:9.0.30729.6161,Adobe%20Refresh%20Manager:1.8.0,Adobe%20Acrobat%20Reader%20DC:15.010.20060,Microsoft%20Visual%20C%2B%2B%202010%20%20x86

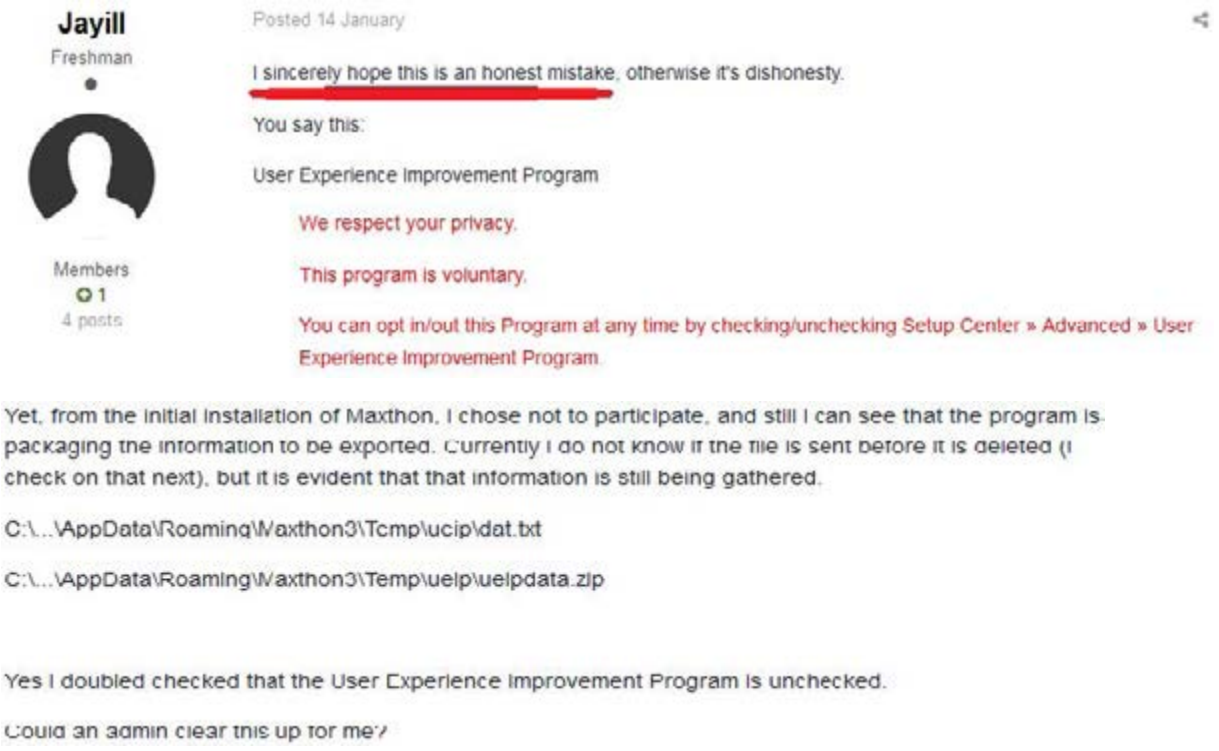
W rzeczywistości: taka ilość informacji, jaka jest przesyłana bez wiedzy użytkownika do serwerów Maxthon otwiera furtkę do przeprowadzenia bardzo precyzyjnego ataku targetowanego. Posiadając wiedzę na temat preferencji przeglądania stron WWW, informację na temat wyszukiwań Google oraz pełną listę zainstalowanego na komputerze użytkownika oprogramowania – atakującemu brakuje tylko adresu email – pod który prześle uwiarygodnioną swoją treścią wiadomość, zawierającą w załączniku uzbrojony exploit zdalnego wykonania kodu.

Dodatkowo, ze względu na kolejny błąd, jaki popełnili autorzy (tym razem jest to błąd w architekturze kryptograficznej) – dane, które płyną bez autoryzacji użytkownika mogą być w każdym momencie podsłuchane i zdeszyfrowane przez każdego potencjalnego atakującego. Wystarczy, że stanie on pomiędzy przeglądarką użytkownika, a serwerem Maxthon, by podsłuchać komunikację. Podsłuchaną transmisję UEIP można zdeszyfrować, używając symetrycznych kluczy algorytmu AES, wydobytych z kodu binarnego Maxthon przy zastosowaniu inżynierii odwrotnej.

Jedną z metod na zdeszyfrowanie w trybie offline plików UEIP przechwyconych w trakcie nasłuchu transmisji TCP może być użycie poniższego fragmentu kodu Python, stworzonego do tego właśnie celu przez grupę Fidelis Threat Research Team. Kod deszyfruje lokalny plik plik dat.txt, używając algorytmu AES w trybie ECB przy pomocy wspomnianego wcześniej klucza symetrycznego pozyskanego w trakcie inżynierii odwrotnej kodu Maxthon:

```
from Crypto.Cipher import AES
data = open(,dat.txt',’rb’).read()
aes = AES.new(,eu3o4[r04cml4eir’, AES.MODE _ ECB)
d = aes.decrypt(data)
print d
```

Tak więc eksperci z SOC słusznie mieli wątpliwości co do bezpieczeństwa użytkowania przeglądarki Maxthon, podobnie jak inni jej użytkownicy, którzy zauważyli na swoich dyskach tworzone pliki `ueipdata.zip`.



Jayill
Freshman

Members
1
4 posts

Posted 14 January

I sincerely hope this is an honest mistake, otherwise it's dishonesty.

You say this:

User Experience Improvement Program

We respect your privacy.

This program is voluntary.

You can opt in/out this Program at any time by checking/unchecking Setup Center » Advanced » User Experience Improvement Program.

Yet, from the initial installation of Maxthon, I chose not to participate, and still I can see that the program is packaging the information to be exported. Currently I do not know if the file is sent before it is deleted (I check on that next), but it is evident that that information is still being gathered.

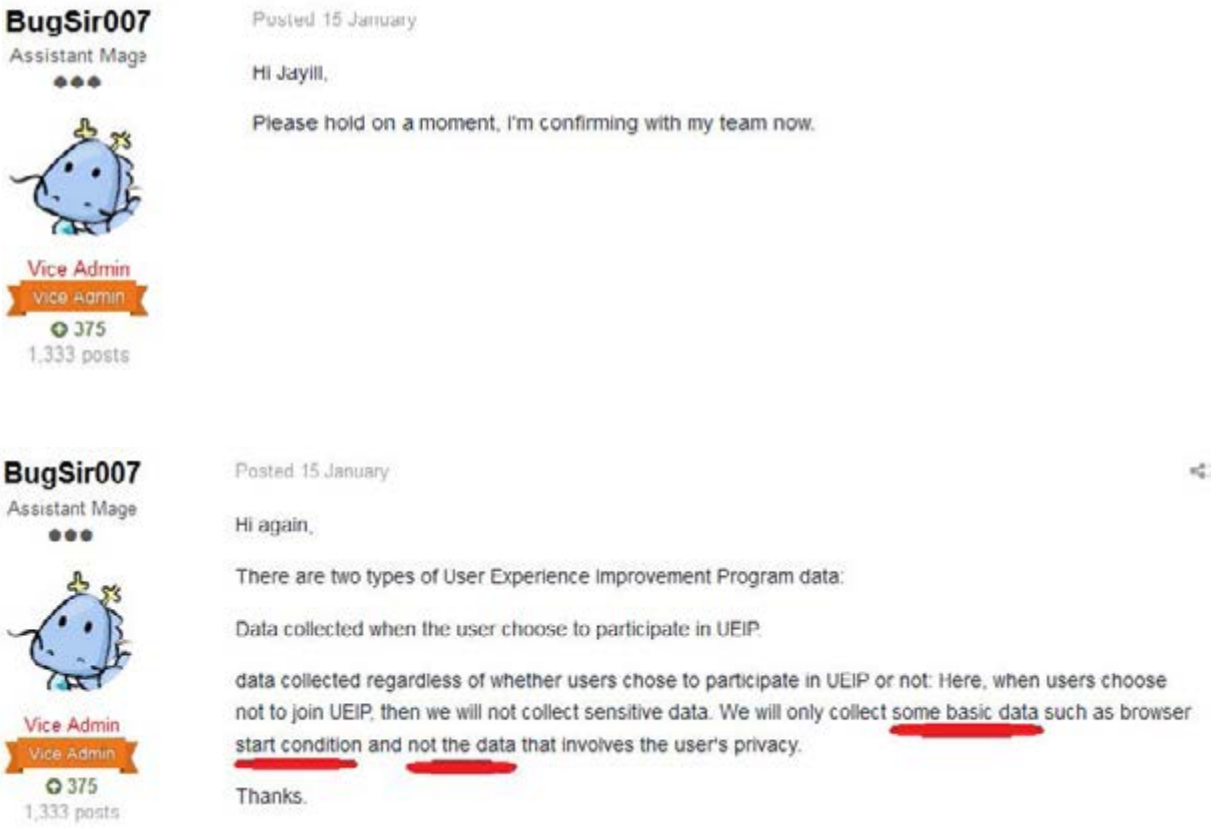
C:\...\AppData\Roaming\Maxthon3\Tcmp\ucip\dat.txt

C:\...\AppData\Roaming\Maxthon3\Temp\uelp\uelpdata.zip

Yes I doubled checked that the User Experience Improvement Program is unchecked.

Could an admin clear this up for me?

Od producenta uzyskano tylko wymijającą odpowiedź, że istnieją... dwa różne typy programu UEIP.



BugSir007
Assistant Mage

Vice Admin
Vice Admin
375
1,333 posts

Posted 15 January

Hi Jayill,

Please hold on a moment, I'm confirming with my team now.

BugSir007
Assistant Mage

Vice Admin
Vice Admin
375
1,333 posts

Posted 15 January

Hi again,

There are two types of User Experience Improvement Program data:

Data collected when the user choose to participate in UEIP.

data collected regardless of whether users chose to participate in UEIP or not: Here, when users choose not to join UEIP, then we will not collect sensitive data. We will only collect some basic data such as browser start condition and not the data that involves the user's privacy.

Thanks.

Natomiast kolejna prośba użytkownika, trochę do autorów, a trochę do innych użytkowników, o pomoc w ujawnieniu dokładnej zawartości plików `ueipdata.zip`, którego utworzenie na swoim dysku zarejestrował użytkownik:

zakończyła się skierowaniem do użytkownika informacji, iż odpowiedzi na swoje pytania znajdzie w opisie polityki prywatności.

Reasumując powyższe: przeglądarka Maxthon nie jest bezpieczna.

Umożliwia przeprowadzenie ataku targetowanego na wybranego użytkownika, wysyłając do autorów przeglądarki pełną listę dokładnych wersji programów podatnych na atak, zainstalowanych na maszynie użytkownika.

Użycie symetrycznej kryptografii i wkompilowanych w kod statycznych kluczy szyfrujących do zabezpieczenia transmisji danych UEIP, umożliwia tak naprawdę dowolnemu atakującemu przeprowadzenie ataku Man-In-The-Middle i rozszyfrowanie przechwyconych między przeglądarką, a pekińskim serwerem Maxthon danych UEIP.

Warto również podkreślić fakt, że Exatel skontaktował się z autorami przeglądarki Maxthon, przesyłając szczegółowy raport techniczny z prośbą o reakcję, np. w postaci poinformowania użytkowników o typie danych wysyłanych z ich przeglądarek do serwerów Maxthon w Pekinie, czy wypuszczenia poprawki, która umożliwiałaby zaniepokojonym użytkownikom efektywne wyłączenie przesyłu plików UEIP do ich serwerów. Prośba ta została zignorowana.

Najnowsza wersja przeglądarki (wersja 4.9.3.1000), pobrana ze strony autorów, została także przetestowana przez ekspertów Exatel i wciąż wysyła dane UEIP, nie respektując w żaden sposób wyboru użytkownika dot. uczestnictwa w tym programie. Do momentu przekazania tych treści do publikacji, nic się nie zmieniło.