

Algorytm DGA wykorzystywany w trojanie Emotet

CERT Orange Polska

Warszawa 29/12/2014

Wstęp.

Trojan Emotet wielokrotnie był już przedmiotem zainteresowania specjalistów. Od czasu jego wykrycia przez firmy zajmujące się bezpieczeństwem teleinformatycznym w połowie 2014 roku, zaobserwowano dużą liczbę jego wariantów. Wersja wykorzystująca algorytm DGA była również opisywana przez CERT Orange Polska, ponieważ kampania masowego rozsyłania spamu związanego z propagacją tej złośliwej aplikacji docierała również do pracowników oraz klientów naszej firmy.

Niniejsza analiza będzie więc skupiać się wyłącznie na algorytmie DGA zastosowanym w trojanie. Jak wiadomo, stosowanie tego typu algorytmów w złośliwych aplikacjach ma na celu utrudnienie neutralizacji botnetów przez organy ścigania oraz osoby zajmujące się bezpieczeństwem teleinformatycznym.

Szczegóły implementacji.

Emotet wykorzystuje między innymi wartości uzyskane z serwera zdalnego do inicjalizowania zmiennych początkowych, które są następnie wykorzystywane do generowania nazw domenowych. Nazwy mają stałą długość dziewiętnastu znaków łącznie z domeną najwyższego poziomu (.eu) i składają się wyłącznie ze znaków alfabetu z przedziału od 'a' do 'z'.

Jak można przeczytać w ogólnej analizie przypadku, zamieszczonej na blogu Orange Polska, trojan przed przystąpieniem do realizacji swoich funkcji sprawdza dostępność połączenia z Internetem poprzez próbę wysłania żądania HTTP do serwera www.microsoft.com. Oprócz sprawdzenia dostępności połączenia, funkcja ta ma również za zadanie zainicjalizować zmienną wykorzystywanej w algorytmie DGA.

Poniżej znajduje się implementacja w języku ANSI C.

```

29 unsigned long ulTimeStampOne; /* First timestamp */
30
31 BOOL __stdcall initFirstTimeStampAndPing(const char* proxy){
32     register BOOL ret = FALSE;
33     DWORD accessType = (proxy != NULL) ? INTERNET_OPEN_TYPE_PROXY :
34                                     INTERNET_OPEN_TYPE_DIRECT;
35     const char* pBypass = (accessType == INTERNET_OPEN_TYPE_PROXY) ? "localhost" :
36                                     NULL;
37     HINTERNET hInternet = InternetOpen("Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0)
38                                     Gecko/20100101 Firefox/34.0",
39                                     accessType, (LPCSTR)proxy,
40                                     (LPCSTR)pBypass, 0);
41     if(hInternet != NULL)
42     {
43         DWORD dContext = 0;
44         HINTERNET hUrl = InternetOpenUrl(hInternet, "http://www.microsoft.com",
45                                     NULL,
46                                     0, INTERNET_FLAG_NO_UI | INTERNET_FLAG_NO_AUTH |
47                                     INTERNET_FLAG_PRAGMA_NOCACHE |
48                                     INTERNET_FLAG_NO_CACHE_WRITE, 0);
49         if(hUrl != NULL) {
50             DWORD oBufLen = 1024;
51             char oBuf[1024 + 1] = {0};
52             ret = HttpQueryInfo(hUrl, 0x40000009, oBuf, &oBufLen, 0);
53             if(ret == TRUE) {
54                 FILETIME fTime = {0};
55                 SYSTEMTIME* sTime = (SYSTEMTIME*)oBuf;
56                 ret = SystemTimeToFileTime(sTime, &fTime);
57                 if(ret == TRUE) {
58                     HMODULE hLib = (HMODULE)GetModuleHandle("ntdll.dll");
59                     if(hLib != NULL) {
60                         _mRtlTimeToSecondsSince1970 RtlTimeSince1970 = (_mRtlTimeToSecondsSince1970)
61                                     GetProcAddress(hLib,
62                                     "RtlTimeToSecondsSince1970");
63                         if( RtlTimeSince1970 != NULL) {
64                             LARGE_INTEGER lInt = {0};
65                             lInt.LowPart = fTime.dwLowDateTime;
66                             lInt.HighPart = fTime.dwHighDateTime;
67                             ret = (RtlTimeSince1970(&lInt, &ulTimeStampOne) > 0) ?
68                                     TRUE : FALSE;
69                         }
70                     }
71                 }
72             }
73             InternetCloseHandle(hUrl);
74         }
75         InternetCloseHandle(hInternet);
76     }
77     return ret;
78 }

```

Jak widać w powyższym listingu, funkcja zwraca wartość typu BOOL (prawda lub fałsz) informującą o tym czy przesłanie żądania HTTP się powiodło, oraz inicjalizuje globalną zmienną "ulTimeStampOne" poprzez konwersję danych zwróconych z wywołania HttpQueryInfo() do struktur SYSTEMTIME i FILETIME przez wywołanie SystemTimeToFileTime() oraz LARGE_INTEGER. Na końcu zapisuje wynik przez wywołanie funkcji RtlTimeToSecondsSince1970().

Warto nadmienić, że wywołanie HttpQueryInfo() z wartością 0x40000009 tak, jak ma to miejsce w tym przypadku zwraca czas pobrany z jednego z nagłówków odpowiedzi HTTP (w tym przypadku odpowiedzi z serwera www.microsoft.com).

Jeżeli wywołanie powyższej procedury powiedzie się, Emotet wywołuje kolejną funkcję inicjalizującą drugą zmienną globalną wykorzystywaną przez DGA.

```

89
90 unsigned long ulTimeStampTwo;          /* Second timestamp */
91
92 BOOL __stdcall initSecondTimestamp(void) {
93     HMODULE hLib = NULL;
94     FILETIME fTime = {0};
95     SYSTEMTIME sTime = {0};
96     sTime.wYear = 0x7DE;
97     sTime.wMonth = 0x0a;
98     sTime.wDay = 0x1c;
99     SystemTimeToFileTime(&sTime,&fTime);
100     hLib = (HMODULE)GetModuleHandle("ntdll.dll");
101     if(hLib != NULL) {
102         _mRtlTimeToSecondsSince1970 RtlTimeSince1970 = (_mRtlTimeToSecondsSince1970)
103                                                         GetProcAddress(hLib,
104                                                         "RtlTimeToSecondsSince1970");
105         if( RtlTimeSince1970 != NULL) {
106             LARGE_INTEGER lInt = {0};
107             lInt.LowPart = fTime.dwLowDateTime;
108             lInt.HighPart = fTime.dwHighDateTime;
109             RtlTimeSince1970(&lInt,&ulTimeStampTwo);
110             return TRUE;
111         }
112     }
113     return FALSE;
114 }

```

W tym przypadku początkowa inicjalizacja struktury SYSTEMTIME bazuje na wartościach na stałe zapisanych w kodzie programu. Wykorzystywane są pola wYear (rok), wMonth (miesiąc), wDay (dzień), pola te są inicjalizowane kolejnymi wartościami 0x7de (2014), 0x0a (10), 0x1c (28) które składają się na datę 28.10.2014 (wartości te mogą być różne w różnych próbkach malware). Następnie tak zainicjalizowana struktura przekazywana jest do funkcji SystemTimeToFileTime , konwertowana do struktury FILETIME i kolejno do LARGE_INTEGER. Wartości tej ostatniej zmiennej przekazywane są do wywołania RtlTimeToSecondsSince1970 którego wynik zapisywany jest w drugiej zmiennej globalnej ulTimeStampTwo. Po inicjalizacji wykorzystywanych wartości trojan uruchamia 16 wątków tworzących nazwy domenowe. Każda z procedur uruchomionych w osobnym wątku działa w pętli wykorzystującej dwa warunki przerwania. W niej wywoływana jest podprocedura, która korzystając z serii instrukcji matematycznych tworzy nazwę domenową na podstawie jednej z uzyskanych wcześniej wartości.

```

127
128 void __stdcall dga(void){ /* procedura uruchamiana w osobnych watach */
129     char domain[0x10 + 0x03 + 0x01] = {0};
130     DNS_STATUS dnsStat = 0;
131     PDNS_RECORD dnsRec = {0};
132     for(;;) {
133         BOOL end = FALSE;
134         EnterCriticalSection(&criticalSection);
135         if(ulTimeStampTwo <= ulTimeStampOne){
136             ulTimeStampTwo += 0x384;
137             end = TRUE;
138         }
139         LeaveCriticalSection(&criticalSection);
140         if(end == FALSE) return;
141         dgaGen(domain);
142         dnsStat = DnsQuery(domain,0x01,0x108,0,&dnsRec,0);
143         if(dnsStat == 0) break;
144         EnterCriticalSection(&criticalSection);
145         printf("[!]. Fake Domain %s \r",domain);
146         LeaveCriticalSection(&criticalSection);
147         memset(domain,0,sizeof(domain));
148     }
149     EnterCriticalSection(&criticalSection);
150     printf("[+]. Active Domain %s\r\n",domain);
151     LeaveCriticalSection(&criticalSection);
152     DnsRecordListFree(dnsRec,1);
153 }

```

Jak widać w powyższym listingu, w pętli porównywane są wartości ulTimeStampTwo i ulTimeStampOne. Jeżeli ulTimeStampTwo jest mniejsza bądź równa ulTimeStampOne ,zostaje

ona zwiększona o wartość 900 i ustawiany jest znacznik kontynuacji. Następnie wywoływana jest procedura dgaGen(domain) która tworzy nazwę DNS. Po powrocie z tej procedury weryfikowana jest poprawność utworzonej w ten sposób nazwy za pomocą wywołania funkcji DnsQuery().

Poniżej przedstawiona została implementacja procedury dgaGen():

```
161 void __stdcall dgaGen(char* domout) {
162     register int i;
163     long int tmp1;
164     unsigned long copyTSone = ulTimeStampTwo;
165     unsigned long int key = 0x51eb851f;
166     copyTSone++;
167     copyTSone *= 0x7fed;
168     copyTSone &= 0x0fffffff;
169     for(i=0;i<0x10;i++) {
170         unsigned char leter = 0;
171         long int tmpLong = 0;
172         unsigned long long int tmp = (unsigned long long int)copyTSone * key;
173         long int tmp1 = tmp >> 32; /*higher part of int*/
174         tmp1 >>= 0x03;
175         tmpLong = (long int)(tmp1 & 0x000000ff);
176         tmp1 *= 0x0d;
177         tmpLong *= 0x19;
178         leter = (char)((copyTSone & 0x000000ff) - (tmpLong & 0x000000ff));
179         leter += 0x61;
180         domout[i] = leter;
181         copyTSone = tmp1;
182     }
183     strncat(domout, ".eu", 0x03);
184 }
```

Na listingu widać, że wykorzystuje ona wartość zmiennej ulTimeStampTwo oraz stałą wartość 0x51eb851f, następnie wykonuje na tych wartościach szereg operacji arytmetyczno-logicznych uzyskując w ten sposób znak alfabetu z przedziału a-z, który zapisywany jest w kolejnych komórkach tablicy wyjściowej. Po ukończeniu pętli do tablicy kopiowane jest rozszerzenie TLD (".eu"). Po zweryfikowaniu "aktywności" uzyskanej w ten sposób nazwy domenowej (wywołanie DnsQuery()) nazwa ta kopiowana jest do tablicy alokowanej w pamięci dynamicznej. W ten sposób uzyskiwana jest cała pula poprawnych i aktywnych domen, które wykorzystywane są kolejno w przypadku, gdy np. odpowiedź na żądanie wysłane do jednej z nich ma niewłaściwy podpis cyfrowy, ponieważ została ona zawieszona lub jest sinkholowana.

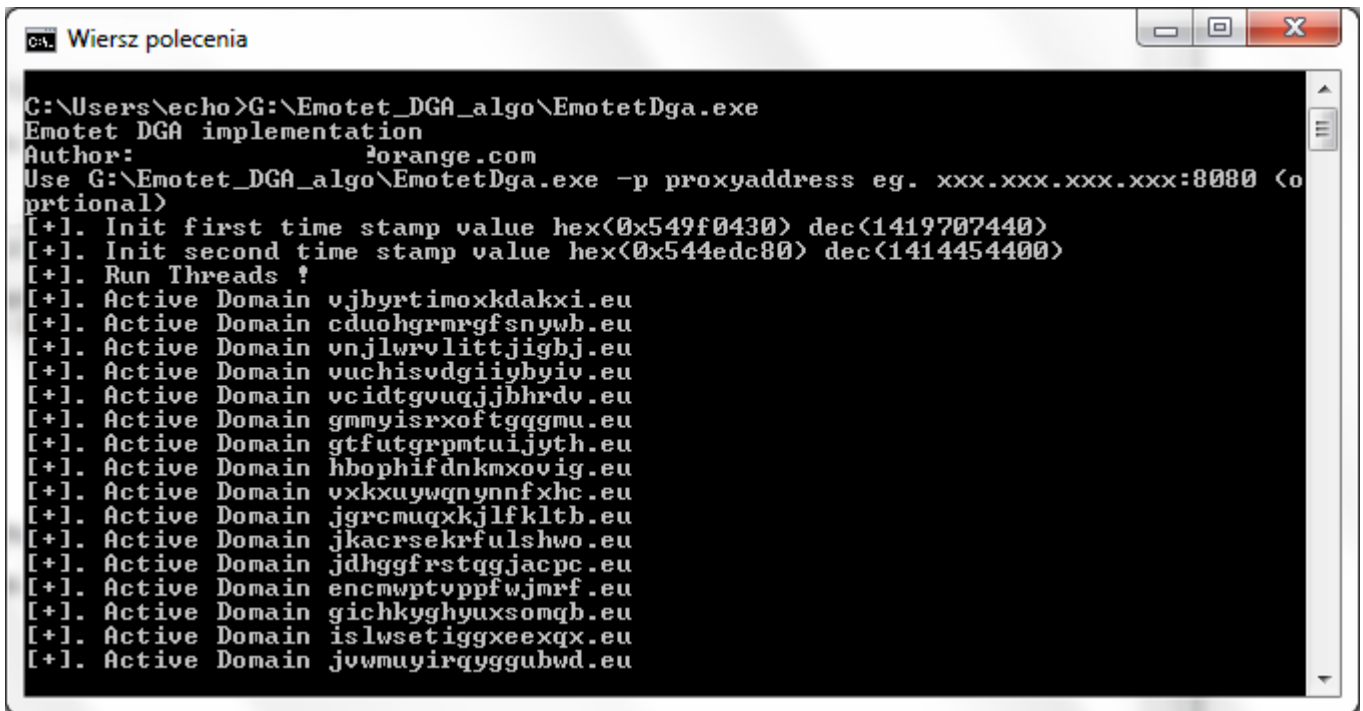
Cały proces badania dostępności łącza i generowania nazw powtarzany jest co 15 minut.

```
[11.03 21:57:47] explorer.exe - www.microsoft.com:80 error : Could not connect to proxy 127.0.0.1:80
[11.03 22:07:27] explorer.exe - www.google.com:80 error : Could not connect to proxy 127.0.0.1:80
[11.03 22:12:49] explorer.exe - www.microsoft.com:80 error : Could not connect to proxy 127.0.0.1:80
[11.03 22:27:52] explorer.exe - www.microsoft.com:80 error : Could not connect to proxy 127.0.0.1:80
[11.03 22:42:55] explorer.exe - www.microsoft.com:80 error : Could not connect to proxy 127.0.0.1:80
```

Pod poniższym adresem znajduje się archiwum .zip zabezpieczone hasłem „Orange2015”, które zawiera kod źródłowy oraz skompilowaną aplikację konsolową implementującą omawiany algorytm DGA.

[EmoDgaTool.zip](#)

Suma MD5 archiwum: ff1b5d0e3db1da9086d00a6930298e69



```
C:\Users\echo>G:\Emotet_DGA_algo\EmotetDga.exe
Emotet DGA implementation
Author: orange.com
Use G:\Emotet_DGA_algo\EmotetDga.exe -p proxyaddress eg. xxx.xxx.xxx.xxx:8080 <optional>
[+] Init first time stamp value hex(0x549f0430) dec(1419707440)
[+] Init second time stamp value hex(0x544edc80) dec(1414454400)
[+] Run Threads !
[+] Active Domain vjbyrtimoxkdakxi.eu
[+] Active Domain cduohgrmrgrfsnywb.eu
[+] Active Domain vnjlwrvlittjigbj.eu
[+] Active Domain vuchisvdgiyhyiv.eu
[+] Active Domain vcidtgvuqjjbhrdv.eu
[+] Active Domain gmmysirxoftgqgm.eu
[+] Active Domain gtfutgrpmtuijyth.eu
[+] Active Domain hbophifdnkmxovig.eu
[+] Active Domain vxkxuywqnyyxfhc.eu
[+] Active Domain jgrcmuqpkjlfklb.eu
[+] Active Domain jkacrsekrfulshwo.eu
[+] Active Domain jdhggfrstqggjacpc.eu
[+] Active Domain encmwptvppfwjmrfe.eu
[+] Active Domain gichkyghyuxsomb.eu
[+] Active Domain islwsetiggxeexqx.eu
[+] Active Domain jvwmyirqygubwd.eu
```

Implementacja ta powstała na podstawie inżynierii odwrotnej posiadanej przez nas próbki i nie jest stuprocentowym odwzorowaniem kodu trojana, a jedynie wykorzystywanego przez niego algorytmu DGA.

Podsumowanie.

Opisywany trojan z pewnością nie należy do skomplikowanych, a kampania jego propagacji jest dość niestandardowa. Wydaje się że dotyka ona wielu krajów, natomiast we wszystkich znanych nam przypadkach celem malware'u są instytucje znajdujące się w Niemczech. Fakt, że analizowany przez CERT Orange Polska wariant korzysta z DGA, a sam Emotet to właściwie dropper może niepokoić. Jednak jak w przypadku wielu innych mechanizmów z obszaru bezpieczeństwa teleinformatycznego (zarówno ofensywnego, jak i defensywnego) algorytmy DGA to broń obosieczna. Potrafi być naprawdę uciążliwa z perspektywy lokalizowania i neutralizacji serwerów C&C, ale jest przy tym stosunkowo „głośna” i potrafi generować naprawdę dużo ruchu sieciowego, którego charakterystyka nie pozostawia wątpliwości co do jego złośliwego przeznaczenia.