Laboratorium 3

Atak SSL Stripping z mitmproxy

Instrukcja laboratoryjna

15 grudnia 2025

Wprowadzenie

Cel laboratorium

Celem tego laboratorium jest praktyczne poznanie ataku SSL Stripping (SSL downgrade attack) - jednej z najgroźniejszych technik ataków Man-in-the-Middle (MitM). Podczas zajęć zobaczysz różnicę między normalnym, bezpiecznym połączeniem HTTPS a połączeniem przechwytywanym przez atakującego.

Czym jest SSL Stripping?

SSL Stripping to technika ataku, w której atakujący wymusza na kliencie korzystanie z niezaszyfrowanego protokołu HTTP zamiast bezpiecznego HTTPS. Klient myśli, że komunikuje się bezpiecznie, ale w rzeczywistości wszystkie dane (hasła, tokeny, dane osobowe) są wysyłane w formie jawnej, którą atakujący może przechwycić.

Scenariusze ataku w rzeczywistości

Atak SSL Stripping może być przeprowadzony w następujących sytuacjach:

- **Publiczne sieci WiFi** - kawiarnie, lotniska, hotele. Atakujący udostępnia fałszywą sieć WiFi lub kompromituje istniejący router.
- **ARP Spoofing w sieci lokalnej** - atakujący podszywając się pod bramę sieciową, przekierowuje cały ruch przez swój komputer.
- **Rogue DHCP Server** - fałszywy serwer DHCP podaje swoją maszynę jako domyślną bramę dla wszystkich klientów w sieci.
- **DNS Spoofing** - atakujący modyfikuje odpowiedzi DNS, przekierowując ruch przez swój serwer proxy.

W tym laboratorium symulujemy sytuację, gdzie atakujący kontroluje routing sieciowy i może przechwytywać cały ruch między klientem a serwerem docelowym.

Pytania kontrolne do odpowiedzi po wykonaniu laboratorium

Pytanie 1

Opisz swoimi słowami rolę każdego z trzech kontenerów w architekturze ataku (ssl_client, ssl_interceptor, target_server).

Pytanie 2

Co widziałeś w tcpdump podczas transmisji HTTPS w Zadaniu 2? Czy możliwe było odczytanie hasła? Wyjaśnij dlaczego.

Pytanie 3

Porównaj wyniki z Zadania 2 i Zadania 3. Co się zmieniło i dlaczego teraz możliwe jest odczytanie wrażliwych danych?

Pytanie 4

Wyjaśnij działanie reguł iptables. Dlaczego porty 80 i 443 są przekierowywane na port 8080? Jaką rolę pełni IP forwarding w tym ataku?

Pytanie 5

Wyjaśnij różnicę między "transparent proxy"a "explicit proxy". Dlaczego w ataku SSL Stripping używa się transparent proxy?

Pytanie 6

Co to jest HSTS (HTTP Strict Transport Security) i jak chroni przed atakami downgrade? Wymień i opisz jeszcze dwa inne mechanizmy obronne przed SSL stripping.

Pytanie 7

Dlaczego curl wymagał opcji -k podczas wykonywania żądań HTTPS? Co oznacza ta flaga i dlaczego jej używanie w produkcji jest niebezpieczne?

Pytanie 8

Jak mitmproxy "podszywa się"pod serwer docelowy wobec klienta? Opisz mechanizm generowania fałszywych certyfikatów i dwukierunkowego połączenia SSL.

Kryteria oceny

Kategoria	Punkty	Opis
Wykonanie ćwiczenia	2	Prawidłowe wykonanie wszystkich zadań laboratoryjnych, dostarczenie wymaganych zrzutów ekranu
Pytanie 1	1	Opis roli kontenerów
Pytanie 2	1	Analiza zaszyfrowanego ruchu HTTPS
Pytanie 3	1	Porównanie normalnego HTTPS z atakiem
Pytanie 4	1	Mechanizm iptables i IP forwarding
Pytanie 5	1	Transparent vs explicit proxy
Pytanie 6	1	HSTS i mechanizmy obronne
Pytanie 7	1	Flaga -k w curl
Pytanie 8	1	Mechanizm podszywania mitmproxy
SUMA	**10**	

1 Zadanie 1: Uruchomienie środowiska

Opis architektury

Środowisko laboratoryjne składa się z trzech kontenerów Docker, które symulują rzeczywisty scenariusz ataku:

ssl_client symuluje użytkownika końcowego (ofiarę ataku), który wysyła żądania HTTPS do serwera. W rzeczywistości może to być komputer w kawiarni, laptop na lotnisku lub telefon podłączony do publicznej sieci WiFi. Kontener ten ma domyślną bramę sieciową ustawioną na adres interceptora, co oznacza że cały jego ruch przechodzi przez potencjalnie wrogi host.

ssl_interceptor pełni rolę atakującego, który przechwytuje i odszyfrowuje cały ruch sieciowy. Działa jako transparent proxy - klient nie wie o jego istnieniu. Kontener ten wykorzystuje iptables do przekierowywania ruchu oraz mitmproxy do aktywnego przechwytywania i deszyfrowania połączeń HTTPS. W rzeczywistości może to być komputer atakującego w tej samej sieci lokalnej, który przeprowadził atak ARP spoofing lub kontroluje punkt dostępowy WiFi.

target_server reprezentuje legalny serwer docelowy (np. bank, portal społecznościowy), z którym klient chce się komunikować. Obsługuje zarówno HTTP jak i HTTPS na portach 80 i 443. Serwer wykorzystuje nginx z samopodpisanymi certyfikatami SSL oraz aplikację httpbin do symulacji prawdziwego serwisu internetowego.

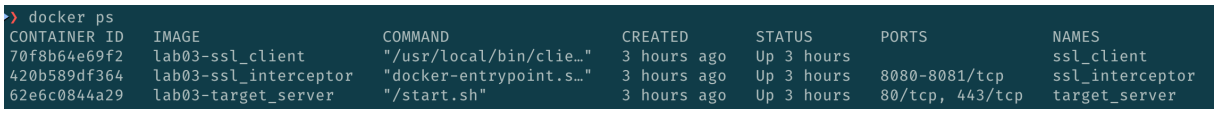
Polecenie

Zbuduj i uruchom środowisko Docker składające się z trzech kontenerów: `ssl_client`, `ssl_interceptor` oraz `target_server`.

Komendy do wykonania:

```
docker-compose -f docker-compose-scenario3.yml build
docker-compose -f docker-compose-scenario3.yml up -d
docker-compose -f docker-compose-scenario3.yml ps
```

Zrzut ekranu 1: Status kontenerów



```
❯ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED       STATUS       PORTS                  NAMES
70f8b64e69f2   lab03-ssl_client       "/usr/local/bin/clie…"   3 hours ago   Up 3 hours                          ssl_client
420b589df364   lab03-ssl_interceptor  "docker-entrypoint.s…"   3 hours ago   Up 3 hours   8080-8081/tcp          ssl_interceptor
62e6c0844a29   lab03-target_server    "/start.sh"              3 hours ago   Up 3 hours   80/tcp, 443/tcp        target_server
```

2 Zadanie 2: Demonstracja normalnego ruchu HTTPS

Kontekst: Jak normalnie dochodzi do ataku Man-in-the-Middle?

W rzeczywistych warunkach atakujący musi najpierw przejąć kontrolę nad ruchem sieciowym ofiary. Najczęstsze metody to:

ARP Spoofing jest techniką, w której atakujący wysyła fałszywe pakiety ARP (Address Resolution Protocol) do ofiary, podszywając się pod bramę sieciową (router). Protokół ARP służy do mapowania adresów IP na adresy MAC w sieci lokalnej. Atakujący informuje komputer ofiary, że jego adres MAC odpowiada adresowi IP routera. W rezultacie wszystkie pakiety przeznaczone dla internetu trafiają najpierw do komputera atakującego, który może je analizować, modyfikować i dopiero potem przekazywać do prawdziwego routera.

Rogue Access Point (fałszywy punkt dostępowy) to sytuacja, w której atakujący tworzy własną sieć WiFi o nazwie podobnej do legalnej sieci. Na przykład w kawiarni istnieje sieć "Starbucks_WiFi", a atakujący tworzy "Starbucks_WiFi_Free"lub dokładnie taką samą nazwę, ale z silniejszym sygnałem. Użytkownicy łączą się z nią nie wiedząc, że jest całkowicie kontrolowana przez atakującego. Każdy pakiet przechodzi przez komputer atakującego, który działa jako router dla tej fałszywej sieci.

DNS Hijacking polega na manipulacji odpowiedziami DNS. Atakujący może zyskać dostęp do routera i zmienić ustawienia DNS, aby wskazywały na jego własny, lub podać fałszywy serwer DNS przez DHCP gdy klienci łączą się z siecią. Gdy użytkownik próbuje odwiedzić stronę (np. bank.pl), jego komputer pyta serwer DNS o adres IP. Fałszywy serwer DNS może zwrócić adres IP serwera kontrolowanego przez atakującego, przekierowując cały ruch przez jego infrastrukturę.

Zyskanie dostępu do routera następuje gdy atakujący wykorzystuje słabe lub domyślne hasła administracyjne routera (np. admin/admin), albo luki w oprogramowaniu urządzenia. Po uzyskaniu dostępu może modyfikować tablice routingu, ustawienia DNS, reguły firewall lub nawet zainstalować własne oprogramowanie. To daje mu pełną kontrolę nad ruchem wszystkich użytkowników danej sieci.

W tym laboratorium pomijamy etap przejęcia kontroli i zakładamy, że atakujący już kontroluje routing. Kontener ssl_interceptor jest skonfigurowany jako domyślna brama dla klienta, co symuluje sytuację po udanym ataku ARP spoofing lub przy połączeniu z rogue access point.

Bezpieczeństwo normalnego HTTPS

W normalnych warunkach, gdy ruch HTTPS przechodzi przez sieć, jest on chroniony przez protokół TLS (Transport Layer Security). Protokół ten zapewnia trzy kluczowe funkcje bezpieczeństwa: szyfrowanie (całość komunikacji jest zaszyfrowana kluczami znanymi tylko klientowi i serwerowi), integralność (każda modyfikacja danych zostanie wykryta) oraz autentykację (certyfikat SSL potwierdza tożsamość serwera).

Nawet jeśli atakujący kontroluje sieć i może przechwytywać wszystkie pakiety, zawartość komunikacji HTTPS pozostaje dla niego całkowicie nieczytelna. Widzi tylko metadane takie jak adresy IP, porty i rozmiar pakietów, ale sama treść żądań HTTP, nagłówki, hasła i dane osobowe są zaszyfrowane i wyglądają jak losowe bajty.

W tym zadaniu zobaczysz jak wygląda przechwycony ruch HTTPS z perspektywy atakującego, który nie prowadzi aktywnego ataku Man-in-the-Middle, a jedynie pasywnie nasłuchuje ruchu sieciowego.

Polecenie

Uruchom monitoring ruchu sieciowego aby pokazać, że normalny ruch HTTPS jest zaszyfrowany i niemożliwy do odczytania.

Komendy do wykonania:

Terminal 1 (ssl_interceptor):

```
docker exec -it ssl_interceptor /bin/bash
/usr/local/bin/interceptor_normal.sh
```

Terminal 2 (ssl_client):

```
docker exec -it ssl_client /bin/bash
ip route del default 2>/dev/null
ip route del 172.22.0.0/24 2>/dev/null
ip route add 172.22.0.3/32 dev eth0
ip route add default via 172.22.0.3
curl -k https://172.22.0.4/post -d "username=student&password=tajnehaslo123"
```

Zrzut ekranu 2: tcpdump pokazujący zaszyfrowany ruch

Na powyższym zrzucie widać pasywne przechwytywanie ruchu HTTPS za pomocą tcpdump. Mimo że pakiety przechodzą przez komputer atakującego (interceptor), zawartość jest całkowicie zaszyfrowana. Widoczne są tylko metadane połączenia TCP (adresy IP, porty, flagi) oraz zaszyfrowana treść przedstawiona jako pozornie losowe bajty (np. "E..<..@...", ".MG]....", "Xa....."). Hasło "tajnehaslo123"wysłane przez klienta jest całkowicie nieczytelne i niemożliwe do odzyskania bez złamania szyfrowania, co przy współczesnych algorytmach kryptograficznych jest praktycznie niewykonalne. To pokazuje dlaczego HTTPS jest uważane za bezpieczne - nawet gdy atakujący kontroluje sieć, nie może odczytać komunikacji.

3 Zadanie 3: Atak SSL Stripping

Kontekst: Jak działa SSL Stripping?

Po przejęciu kontroli nad ruchem sieciowym, atakujący może przeprowadzić aktywny atak Man-in-the-Middle używając narzędzia takiego jak mitmproxy. Mechanizm SSL Stripping działa w następujący sposób:

Krok 1: Przechwycenie żądania. Gdy klient wysyła żądanie HTTPS do serwera, pakiet jest automatycznie przechwytywany przez atakującego dzięki wcześniej przejętej kontroli nad routingiem (np. poprzez ARP spoofing). Zamiast trafić bezpośrednio do serwera docelowego, pakiet dociera najpierw do komputera atakującego. Dzięki regułom iptables skonfigurowanym w trybie REDIRECT, cały ruch na porty 80 (HTTP) i 443 (HTTPS) jest automatycznie przekierowywany do lokalnego procesu mitmproxy nasłuchującego na porcie 8080.

Krok 2: Dwukierunkowe połączenie SSL. Mitmproxy tworzy dwa niezależne połączenia SSL/TLS zamiast jednego end-to-end. Pierwsze połączenie jest nawiązywane między klientem a mitmproxy - atakujący generuje dynamicznie fałszywy certyfikat SSL dla domeny docelowej, podpisany przez własne Certificate Authority (CA). Drugie połączenie jest nawiązywane między mitmproxy a prawdziwym serwerem - tutaj używany jest oryginalny, prawidłowy certyfikat serwera. Mitmproxy znajduje się dokładnie pośrodku tej komunikacji.

Krok 3: Deszyfrowanie i ponowne szyfrowanie. Klient szyfruje swoje dane (w tym hasła, tokeny) używając klucza sesji uzgodnionego z mitmproxy. Mitmproxy odszyfrowuje te dane i widzi je w formie całkowicie jawnej - plaintext. Następnie mitmproxy ponownie szyfruje te same dane używając klucza sesji uzgodnionego z prawdziwym serwerem i przekazuje je dalej. Serwer odbiera dane, przetwarza je i wysyła odpowiedź. Odpowiedź przechodzi ten sam proces w drugą stronę - serwer szyfruje, mitmproxy deszyfruje i widzi w plaintext, następnie ponownie szyfruje dla klienta.

Krok 4: Niewidoczny pośrednik. Z perspektywy klienta komunikacja wygląda normalnie - połączenie jest zaszyfrowane HTTPS, widoczna jest kłódka w przeglądarce. Jedynym sygnałem ostrzegawczym jest ostrzeżenie o niezaufanym certyfikacie, które wielu użytkowników ignoruje lub które jest pominięte za pomocą flagi -k w curl. Z perspektywy serwera także wszystko wygląda normalnie - otrzymuje prawidłowe żądanie HTTPS od klienta (w rzeczywistości od mitmproxy). Żadna ze stron nie wie o istnieniu pośrednika, ale atakujący widzi całą komunikację w postaci jawnej i może ją modyfikować w czasie rzeczywistym.

Dlaczego to działa? Bezpieczeństwo HTTPS opiera się na zaufaniu do certyfikatów SSL. Gdy mitmproxy prezentuje fałszywy certyfikat, klient otrzymuje ostrzeżenie, że certyfikat nie jest podpisany przez zaufane CA. Jednak w wielu scenariuszach użytkownicy ignorują te ostrzeżenia, szczególnie na urządzeniach mobilnych lub gdy są przyzwyczajeni do klikania "Kontynuuj mimo ostrzeżenia". W laboratorium używamy flagi -k w curl, która symuluje takie zachowanie. W przypadku gdy atakujący zainstalował swoje CA w systemie ofiary (np. przez złośliwe oprogramowanie lub fizyczny dostęp do urządzenia), ostrzeżenia nie będzie wcale.

Polecenie

Skonfiguruj i uruchom atak SSL Stripping, który pozwoli na odszyfrowanie całego ruchu HTTPS.

Komendy do wykonania:

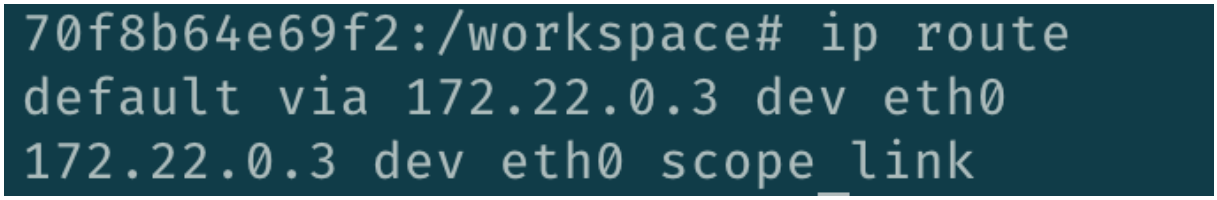
Terminal 1 (ssl_interceptor):

```
# Nacisnij Ctrl+C aby zatrzymac tcpdump
/usr/local/bin/interceptor_attack.sh
```

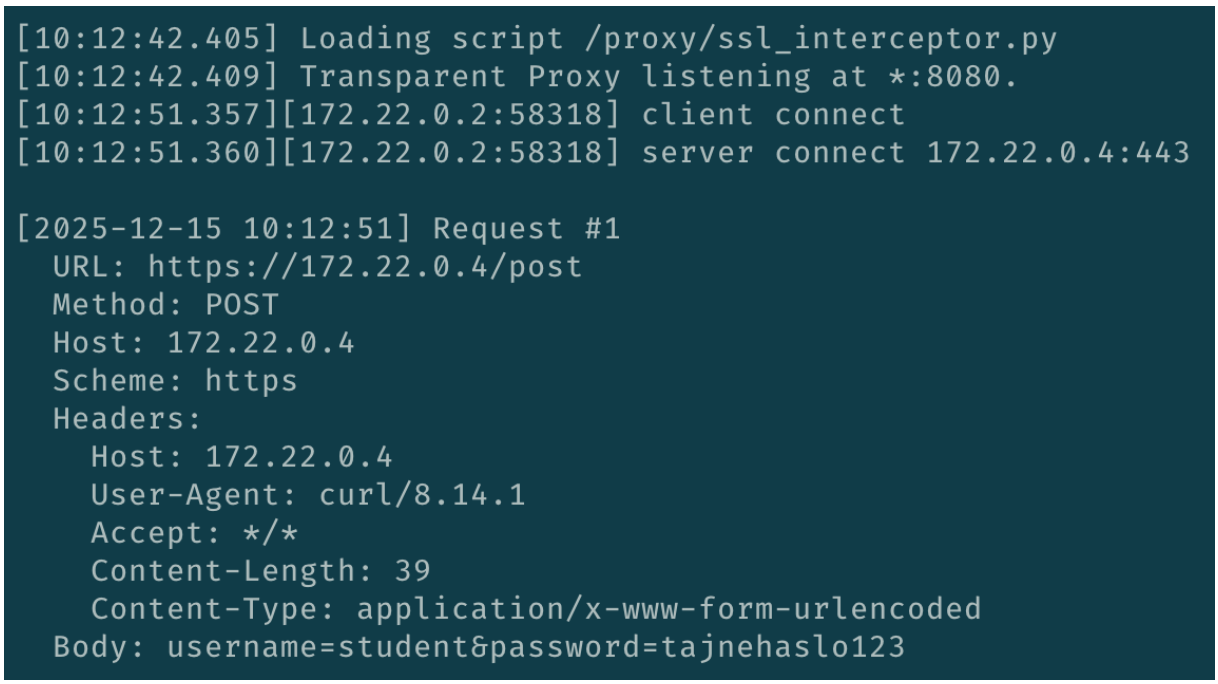
Terminal 2 (ssl_client):

```
ip route
curl -k https://172.22.0.4/post -d "username=student&password=tajnehaslo123"
curl -k -H "Authorization: Bearer token_abc123xyz" https://172.22.0.4/headers
curl -k https://172.22.0.4/get
```

Zrzut ekranu 3: Tablica routingu klienta



Zrzut ekranu 4: Przechwycone dane POST z hasłem



Porównanie wyników

Porównując Zadanie 2 z Zadaniem 3, widzimy fundamentalną różnicę w możliwościach atakującego. W Zadaniu 2 ruch przechodził przez interceptor, ale był tylko pasywnie monitorowany za pomocą tcpdump na poziomie pakietów sieciowych. Tcpdump przechwytywał surowe pakiety IP/TCP, ale zawartość warstwy aplikacji (HTTP) pozostawała zaszyfrowana przez protokół TLS. Atakujący widział tylko metadane połączenia i zaszyfrowane bajty, całkowicie bezużyteczne bez klucza deszyfrującego.

W Zadaniu 3 sytuacja jest inna. Uruchomiliśmy mitmproxy - narzędzie które aktywnie przechwytuje połączenia HTTPS i działa jako Man-in-the-Middle. Mitmproxy nie ogranicza się do pasywnego nasłuchiwania - tworzy dwa niezależne połączenia SSL: jedno z klientem, drugie z serwerem docelowym. Klient negocjuje sesję TLS z mitmproxy (nie wiedząc że to nie jest prawdziwy serwer), a mitmproxy osobno negocjuje sesję TLS z prawdziwym serwerem.

Dzięki temu mitmproxy znajduje się w środku komunikacji i może deszyfrować ruch w obie strony. Dane od klienta są deszyfrowane kluczem sesji client-mitmproxy, oglądane w plaintext przez atakującego, a następnie ponownie szyfrowane kluczem sesji mitmproxy-server przed przekazaniem do celu. Ten proces działa transparentnie dla obu stron - klient myśli że komunikuje się bezpośrednio z serwerem (widzi HTTPS i kłódkę), serwer myśli że komunikuje się bezpośrednio z klientem, ale w rzeczywistości pośrodku siedzi atakujący który widzi wszystko w formie jawnej, włącznie z hasłami, tokenami autoryzacyjnymi, numerami kart kredytowych i wszelkimi innymi danymi osobowymi.

To jest możliwe ponieważ mitmproxy generuje własne certyfikaty SSL dla każdej przechwytwanej domeny, podpisane przez własne Certificate Authority. Gdy klient łączy się, otrzymuje fałszywy certyfikat od mitmproxy zamiast prawdziwego od serwera docelowego. W normalnych warunkach przeglądarka powinna wyświetlić duże czerwone ostrzeżenie o niezaufanym certyfikacie, ale w laboratorium używamy flagi -k która to ostrzeżenie ignoruje. W rzeczywistości wielu użytkowników ignoruje takie ostrzeżenia, szczególnie gdy łączą się z sieciami WiFi które wymagają akceptacji certyfikatu portalu captive.

4 Część teoretyczna

4.1 Transparent proxy vs Explicit proxy

Explicit proxy (zwane także forward proxy lub proxy jawne) wymaga ręcznej konfiguracji po stronie klienta. Użytkownik musi w ustawieniach przeglądarki lub systemu operacyjnego podać adres IP i port serwera proxy (np. 192.168.1.10:8080). Klient świadomie wysyła wszystkie żądania HTTP/HTTPS do tego serwera proxy, a proxy z kolei przekazuje je do docelowych serwerów. W żądaniu HTTP klient nie wysyła `GET /index.html`, ale `GET http://example.com/index.html` - pełny URL z nazwą hosta. Serwer proxy zna zarówno źródło (klienta) jak i cel (docelowy serwer) każdego żądania.

Transparent proxy (zwane także intercepting proxy lub przechwytujące proxy) działa całkowicie bez wiedzy i konfiguracji użytkownika. Ruch sieciowy jest automatycznie przekierowywany do proxy przez mechanizmy na poziomie sieci - najczęściej przez reguły iptables na bramie sieciowej lub routerze. Klient wysyła normalne żądania `GET /index.html` myśląc że komunikuje się bezpośrednio z serwerem docelowym. Proxy musi odzyskać informację o oryginalnym celu z nagłówków HTTP (Host:) lub z metadanych połączenia TCP. Z perspektywy aplikacji klienckiej proxy jest całkowicie niewidoczne.

W kontekście ataku SSL Stripping, transparent proxy jest absolutnie niezbędne z kilku kluczowych powodów. Po pierwsze, ofiara nie może wiedzieć o istnieniu pośrednika - cały punkt ataku polega na tym że użytkownik myśli że komunikuje się bezpiecznie. Gdyby wymagana była konfiguracja explicit proxy, użytkownik natychmiast wiedziałby że jego ruch jest monitorowany. Po drugie, transparent proxy działa dla wszystkich aplikacji i urządzeń w sieci - nie tylko przeglądarek, ale także aplikacji mobilnych, urządzeń IoT, smart TV. Nie wszystkie z nich w ogóle obsługują konfigurację proxy. Po trzecie, transparent proxy wymaga dostępu do infrastruktury sieciowej (router, brama), co jest często możliwe w publicznych sieciach WiFi, ale daje atakującemu kontrolę nad całym ruchem wszystkich użytkowników tej sieci jednocześnie.

4.2 SSL Stripping - definicja i przykłady

SSL Stripping to technika ataku Man-in-the-Middle, w której atakujący przechwytuje komunikację między klientem a serwerem i aktywnie deszyfruje połączenia HTTPS poprzez utworzenie dwóch oddzielnych połączeń SSL. Atakujący nawiązuje jedno połączenie SSL z klientem (podszywając się pod serwer docelowy) i drugie połączenie SSL z prawdziwym serwerem (podszywając się pod klienta). Znajduje się dokładnie pośrodku, deszyfruje cały ruch przychodzący, analizuje go i może modyfikować, a następnie ponownie szyfruje i przekazuje do drugiej strony.

Kluczowym elementem ataku jest kontrola nad routingiem sieciowym - atakujący musi zapewnić że pakiety od ofiary będą przechodziły przez jego komputer zamiast bezpośrednio do celu. Dodatkowo atakujący generuje fałszywe certyfikaty SSL dla przechwytywanych domen, podpisane przez własne Certificate Authority. Klient otrzymuje ostrzeżenie o niezaufanym certyfikacie, ale jeśli je zignoruje, połączenie zostaje nawiązane i atakujący zyskuje pełen dostęp do komunikacji.

Przykład rzeczywistego scenariusza: Student Jan siedzi w kawiarni Starbucks i łączy się z siecią WiFi "Starbucks_WiFi". W rzeczywistości to nie jest prawdziwa sieć kawiarni,

ale rogue access point utworzony przez atakującego siedzącego przy sąsiednim stoliku. Laptop atakującego działa jako router dla tej fałszywej sieci i ma zainstalowane narzędzie mitmproxy. Jan otwiera przeglądarkę i loguje się do swojej bankowości internetowej. Mimo że w pasku adresu widzi "https://bank.pl"i zieloną kłódkę, otrzymuje krótkie ostrzeżenie o certyfikacie, które szybko klika "Kontynuuj"(przyzwyczaił się do takich ostrzeżeń w publicznych WiFi). Wprowadza login "jan.kowalski"i hasło "MojeHaslo123!". W tym momencie atakujący widzi na swoim ekranie w mitmproxy całe żądanie POST z loginem i hasłem w postaci plaintext. Dodatkowo przechwytuje token sesji. Jan nawet nie wie że coś jest nie tak - strona banku działa normalnie, może sprawdzić saldo, wykonać przelew. Wieczorem atakujący loguje się na konto Jana używając przechwyconego hasła i wyprowadza wszystkie pieniądze.

Inny przykład: Firma organizuje konferencję w hotelu. Wi-Fi hotelu jest zabezpieczone słabym hasłem "Hotel2024"które jest wydrukowane na ulotkach dla gości. Atakujący także pozna hasło i łączy się z siecią. Następnie przeprowadza atak ARP spoofing na routerze hotelowym, podszywając się pod bramę sieciową dla wszystkich urządzeń w sieci. Każde urządzenie myśli że adres MAC atakującego należy do routera i wysyła do niego wszystkie pakiety. Atakujący przekazuje te pakiety do prawdziwego routera (IP forwarding), ale po drodze przepuszcza je przez mitmproxy. W ciągu kilku godzin przechwytuje dane logowania do służbowych maili, VPN, systemów firmowych dziesiątek uczestników konferencji. Nazajutrz przeprowadza atak na infrastrukturę firmy używając przechwyconego dostępu VPN.

4.3 Mechanizmy obronne przed SSL Stripping

HSTS (HTTP Strict Transport Security) to mechanizm działający przez specjalny nagłówek HTTP wysyłany przez serwer: `Strict-Transport-Security: max-age=31536000; includeSubDomains; preload`. Gdy przeglądarka po raz pierwszy łączy się z serwerem przez HTTPS i otrzymuje ten nagłówek, zapamiętuje że przez określony czas (max-age w sekundach, tutaj rok) ma zawsze łączyć się z tą domeną wyłącznie przez HTTPS. Jeśli użytkownik wpisze "http://example.com"lub kliknie link HTTP, przeglądarka automatycznie i wewnętrznie zmieni to na "https://example.com"jeszcze przed wysłaniem jakiegokolwiek żądania do sieci. Nawet jeśli atakujący spróbuje przekierować na niezaszyfrowaną wersję strony, przeglądarka odmówi połączenia.

Dyrektywa "includeSubDomains"rozszerza ochronę na wszystkie subdomeny (np. mail.example.com, www.example.com). Dyrektywa "preload"wskazuje że domena powinna być dodana do wbudowanej listy HSTS w przeglądarkach - lista ta jest dystrybuowana z samą przeglądarką, więc ochrona działa nawet przy pierwszym połączeniu (rozwiązuje to problem trust-on-first-use). Największym ograniczeniem HSTS jest właśnie pierwsze połączenie - jeśli użytkownik nigdy wcześniej nie odwiedził danej strony, nie ma jeszcze zapisanego wymogu HTTPS i jest podatny na atak podczas tego pierwszego połączenia.

Certificate Pinning to technika w której aplikacja lub przeglądarka zapamiętuje konkretny certyfikat SSL lub klucz publiczny danej domeny i akceptuje tylko ten konkretny certyfikat, odrzucając wszystkie inne, nawet jeśli są podpisane przez zaufane CA. W praktyce oznacza to że aplikacja mobilna banku ma "na sztywno"wpisany w kodzie hash klucza publicznego serwera bank.pl. Gdy aplikacja łączy się z serwerem, sprawdza czy prezen-

towany certyfikat zawiera ten konkretny klucz publiczny. Jeśli atakujący przechwytuje połączenie i prezentuje własny certyfikat (nawet podpisany przez prawdziwe CA), aplikacja odrzuca połączenie ponieważ hash klucza publicznego się nie zgadza. To całkowicie uniemożliwia ataki MITM, ponieważ atakujący nie ma dostępu do prywatnego klucza prawdziwego serwera. Wadą jest to że wymaga aktualizacji aplikacji gdy serwer zmienia certyfikat lub klucz, co może powodować problemy jeśli użytkownicy nie aktualizują aplikacji.

VPN (Virtual Private Network) chroni przed SSL Stripping przez utworzenie zaszyfrowanego tunelu od urządzenia klienta do serwera VPN, zanim ruch w ogóle dotrze do lokalnej sieci. Nawet jeśli użytkownik łączy się ze skompromitowaną siecią WiFi, atakujący widzi tylko zaszyfrowany tunel VPN i nie może przechwycić jego zawartości. Cała komunikacja (włącznie z połączeniami HTTPS) odbywa się już wewnątrz tego tunelu. Atakujący kontrolujący lokalną sieć widzi tylko że użytkownik łączy się z serwerem VPN, ale nie widzi do jakich stron później się łączy ani jakie dane przesyła.

Rozszerzenia przeglądarki takie jak "HTTPS Everywhere"automatycznie przekierowują żądania HTTP na HTTPS dla tysięcy znanych stron internetowych. Rozszerzenie posiada bazę danych reguł przepisywania URL - np. "zawsze zmieniaj http://facebook.com na https://facebook.com". Działa to nawet jeśli użytkownik kliknie w link HTTP lub wpisze adres bez https://. Nowoczesne przeglądarki mają też wbudowane funkcje takie jak "Always use secure connections"które preferują HTTPS i ostrzegają gdy strona próbuje użyć niezabezpieczonego połączenia.

Edukacja użytkowników jest prawdopodobnie najważniejszym mechanizmem obronnym. Użytkownicy muszą nauczyć się rozpoznawać ostrzeżenia o certyfikatach i nigdy ich nie ignorować. Muszą wiedzieć że publiczne sieci WiFi są niebezpieczne i unikać logowania do wrażliwych serwisów lub używać VPN. Sprawdzanie czy w pasku adresu widnieje "https://"i kłódka przed wprowadzeniem danych logowania. Nieufność wobec sieci o nazwach podobnych do znanych sieci ale z drobnymi różnicami (Starbucks-WiFi vs Starbucks_WiFi).

4.4 Mechanizm podszywania się mitmproxy

Mitmproxy implementuje atak Man-in-the-Middle poprzez działanie jako pośrednik w dwóch niezależnych połączeniach SSL/TLS, tworząc iluzję bezpośredniego połączenia między klientem a serwerem.

Gdy klient inicjuje połączenie HTTPS (np. https://bank.pl), najpierw nawiązuje połączenie TCP z serwerem na porcie 443. W naszym scenariuszu, dzięki kontroli routingu i iptables REDIRECT, to połączenie TCP trafia do mitmproxy zamiast do prawdziwego serwera. Mitmproxy akceptuje połączenie i rozpoczyna handshake TLS z klientem. W tej fazie klient wysyła ClientHello zawierający listę obsługiwanych algorytmów szyfrowania i wersji TLS. Mitmproxy odpowiada ServerHello wybierając algorytm i prezentuje certyfikat SSL.

Tutaj następuje kluczowy moment - mitmproxy dynamicznie generuje fałszywy certyfikat dla domeny bank.pl. Ten certyfikat zawiera nazwę domeny "bank.pl"w polu Common Name lub Subject Alternative Names, więc zgadza się z nazwą do której klient próbuje się połączyć. Certyfikat jest podpisany przez własne Certificate Authority mitmproxy - para kluczy CA jest generowana przy pierwszym uruchomieniu mitmproxy i zapisywana

w /.mitmproxy/. Klient sprawdza certyfikat i odkrywa że jest podpisany przez CA którego nie zna (nie ma w systemowym magazynie zaufanych CA). Normalnie przeglądarka wyświetliłaby duże czerwone ostrzeżenie. W naszym przypadku curl z flagą -k ignoruje to ostrzeżenie i kontynuuje handshake.

Klient i mitmproxy negocjują klucze sesji używając wybranego algorytmu (np. ECDHE dla perfect forward secrecy). Na końcu mają wspólny klucz symetryczny którym będą szyfrować całą dalszą komunikację. Z perspektywy klienta połączenie SSL jest nawiązane z "bank.pl"(nie wie że to faktycznie mitmproxy).

Równolegle mitmproxy inicjuje drugie, całkowicie niezależne połączenie SSL do prawdziwego serwera bank.pl. Tym razem mitmproxy występuje jako klient - wysyła ClientHello, odbiera prawdziwy certyfikat bank.pl (podpisany przez Let's Encrypt lub inne prawdziwe CA), weryfikuje go (mitmproxy zna prawdziwe CA), negocjuje klucze sesji. Z perspektywy serwera bank.pl, to normalne połączenie od klienta (nie wie że to mitmproxy).

Teraz mitmproxy ma dwa aktywne połączenia SSL: client mitmproxy (szyfrowane kluczem K1) oraz mitmproxy server (szyfrowane kluczem K2). Gdy klient wysyła żądanie HTTP, np. "POST /login username=jan&password=haslo", to żądanie jest zaszyfrowane kluczem K1. Mitmproxy odbiera zaszyfrowane bajty, deszyfruje je używając K1 i widzi całe żądanie w postaci plaintext na swoim ekranie. Może je zalogować, zmodyfikować, przeanalizować. Następnie mitmproxy szyfruje to samo żądanie (lub zmodyfikowaną wersję) używając klucza K2 i wysyła do prawdziwego serwera. Serwer odbiera, deszyfruje kluczem K2, przetwarza i wysyła odpowiedź. Odpowiedź idzie tym samym procesem w drugą stronę.

Co ważne, mitmproxy generuje osobny certyfikat dla każdej przechwytywane domeny. Jeśli klient połączy się z bank.pl, potem z facebook.com, potem z gmail.com, mitmproxy wygeneruje trzy różne certyfikaty - każdy z odpowiednią nazwą domeny. To sprawia że połączenia wyglądają bardziej prawdziwie (gdyby wszystkie miały ten sam certyfikat, byłoby to bardzo podejrzane). Certyfikaty są generowane "on-the-fly"w czasie rzeczywistym podczas handshake TLS, wykorzystując wcześniej wygenerowany klucz prywatny CA. Operacja ta jest bardzo szybka dzięki nowoczesnym algorytmom kryptograficznym i nie wprowadza zauważalnego opóźnienia.

Dodatkowe funkcje mitmproxy obejmują możliwość modyfikacji żądań i odpowiedzi w locie, zapisywanie całego ruchu do pliku w różnych formatach (mitm, HAR, plain HTTP), implementację skryptów Python (jak nasz ssl_interceptor.py) które mogą reagować na specific eventy, automatyczne dekodowanie i formatowanie różnych typów contentu (JSON, XML, images), obsługę WebSocket, HTTP/2 i innych protokołów. W kontekście ataku SSL Stripping najważniejsza jest jednak podstawowa funkcja: transparentne przechwytywanie i deszyfrowanie ruchu HTTPS bez wiedzy żadnej ze stron.

5 Obrona przed atakiem SSL Stripping

5.1 Mechanizmy obronne dla użytkowników końcowych

Zwracanie uwagi na ostrzeżenia przeglądarki jest pierwszą linią obrony. Nowoczesne przeglądarki wyświetlają bardzo widoczne ostrzeżenia gdy certyfikat SSL jest problematyczny. Nigdy nie należy klikać "Zaawansowane"i "Przejdź mimo to"bez dokładnego zrozumienia co się dzieje. Kłódka w pasku adresu i zielony napis "https://"powinny być sprawdzane przed wprowadzeniem jakichkolwiek wrażliwych danych. Kliknięcie na kłódkę pokazuje szczegóły certyfikatu - warto sprawdzić czy wystawca certyfikatu to znana firma (Let's Encrypt, DigiCert) a nie dziwna nazwa.

Używanie VPN w publicznych sieciach jest najbardziej efektywną obroną techniczną dostępną dla przeciętnego użytkownika. VPN tworzy zaszyfrowany tunel od urządzenia użytkownika do serwera VPN jeszcze przed opuszczeniem lokalnej sieci. Nawet jeśli atakujący kontroluje punkt dostępowy WiFi lub router, widzi tylko zaszyfrowany tunel VPN którego nie może odszyfrować. Cała komunikacja z bankami, emailem, social media odbywa się już wewnątrz tego tunelu. Ważne jest jednak aby używać renomowanego, płatnego VPN (nie darmowego) i aby automatycznie łączył się przy detekcji publicznej sieci.

Rozszerzenia przeglądarki takie jak HTTPS Everywhere (obecnie wbudowane w większość przeglądarek jako "Always use secure connections") automatycznie wymuszają HTTPS tam gdzie jest dostępne. Ustawienie to powinno być włączone - zapobiega sytuacjom gdzie użytkownik kliknie stary link HTTP i zostanie podatny na atak. Nowoczesne przeglądarki coraz częściej domyślnie próbują HTTPS przed próbą HTTP.

Ostrożność w publicznych sieciach WiFi jest kluczowa. Należy całkowicie unikać logowania do wrażliwych serwisów (bankowość, służbowy email, systemy firmowe) w publicznych WiFi bez VPN. Jeśli to konieczne, lepiej użyć danych komórkowych (LTE/5G) które są znacznie bezpieczniejsze niż WiFi. Zawsze weryfikować nazwę sieci WiFi z personelem - atakujący często tworzą sieci o podobnych nazwach ("Starbucks-Guest"vs "Starbucks_Guest").

Uwierzytelnianie dwuskładnikowe (2FA) znacząco zmniejsza szkody nawet jeśli hasło zostanie przechwycone. Atakujący przechwyci hasło ale nie będzie miał dostępu do drugiego składnika (kod SMS, aplikacja authenticator, klucz sprzętowy). Preferowane są aplikacje 2FA (Google Authenticator, Authy) nad SMS ponieważ SMS może być przechwycony przez SIM swapping. Najlepsze są klucze sprzętowe FIDO2 (YubiKey) które są odporne na phishing i MitM.

5.2 Mechanizmy obronne dla administratorów i deweloperów

Wdrożenie HSTS (HTTP Strict Transport Security) jest podstawowym wymogiem dla każdej strony obsługującej wrażliwe dane. Serwer powinien wysyłać nagłówek `Strict-Transport-Secur` `max-age=63072000; includeSubDomains; preload`. Max-age=63072000 to dwa lata - wystarczająco długo by chronić użytkowników między wizytami. Dyrektywa includeSubDomains chroni wszystkie subdomeny. Dyrektywa preload wskazuje gotowość do dodania do HSTS Preload List przeglądarek - należy złożyć wniosek na hstspreload.org. Lista ta jest wbudowana w przeglądarki, więc ochrona działa od pierwszego połączenia (eliminuje problem trust-on-first-use).

Certificate Pinning w aplikacjach mobilnych całkowicie eliminuje ataki MitM. Aplikacja ma wbudowany hash klucza publicznego serwera lub hash całego certyfikatu. Przy każdym połączeniu weryfikuje czy serwer prezentuje ten konkretny klucz. Jeśli nie zgadza się, odmawia połączenia nawet jeśli certyfikat jest podpisany przez zaufane CA. Implementacja wymaga planowania - trzeba mieć mechanizm aktualizacji pinów (np. pinowanie dwóch certyfikatów: aktualnego i przyszłego, albo pinowanie klucza publicznego CA) oraz fallback plan gdyby trzeba było awaryjnie zmienić certyfikat.

Automatyczne przekierowanie HTTP na HTTPS powinno być skonfigurowane na poziomie serwera web. Dla nginx: `return 301 https://$server_name$request_uri;`. Ważne jest używanie kodu 301 Moved Permanently (nie 302 Found) aby przeglądarki zapamiętały przekierowanie. Niektóre frameworki webowe mają wbudowane middleware do wymuszania HTTPS.

Wyłączenie obsługi HTTP całkowicie dla wrażliwych aplikacji. Serwery bankowe, medyczne, rządowe nie powinny w ogóle nasłuchiwać na porcie 80. Jedynym dozwolonym punktem dostępu powinien być port 443 z TLS. Eliminuje to możliwość ataku SSL Stripping ponieważ nie ma możliwości połączenia HTTP.

Właściwa konfiguracja TLS obejmuje wyłączenie starych wersji protokołów (SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1 wszystkie mają znane luki), używanie tylko TLS 1.2 i TLS 1.3. Lista szyfrów powinna preferować forward secrecy (ECDHE) i silne algorytmy (AES-256-GCM). Narzędzie Mozilla SSL Configuration Generator dostarcza gotowe konfiguracje dla różnych serwerów web. Certyfikaty powinny być od renomowanych CA (Let's Encrypt jest darmowe i automatyczne), używać minimum 2048-bit RSA lub 256-bit ECDSA, i być regularnie odnawiane (Let's Encrypt co 90 dni).

Monitorowanie i alerting obejmują analizę logów pod kątem podejrzanej aktywności: duża liczba ostrzeżeń o certyfikatach, połączenia z nietypowych lokalizacji, niepowodzenia uwierzytelniania. Na poziomie sieci LAN implementacja wykrywania ARP spoofing (narzędzia takie jak arpwatch, XArp) - alarmują gdy widzą duplikat adresu MAC dla tego samego IP lub zmiany w tablicy ARP. Systemy IDS/IPS (Intrusion Detection/Prevention) takie jak Snort, Suricata mogą wykrywać wzorce ataków MitM. Segmentacja sieci ogranicza zasięg potencjalnego ataku - gości WiFi oddzielić od sieci firmowej VLANem.

Podsumowanie

Atak SSL Stripping demonstruje fundamentalną słabość w modelu zaufania internetu - użytkownicy często ignorują ostrzeżenia bezpieczeństwa, a samo szyfrowanie HTTPS nie chroni jeśli atakujący znajduje się między klientem a serwerem. Laboratorium pokazało jak relatywnie prosta konfiguracja (kontrola routingu + mitmproxy + iptables) pozwala na pełne odszyfrowanie ruchu HTTPS.

Kluczowe wnioski to znaczenie wielowarstwowej obrony: HSTS chroni na poziomie przeglądarki, Certificate Pinning na poziomie aplikacji, VPN na poziomie sieci, edukacja na poziomie użytkownika. Żaden pojedynczy mechanizm nie jest doskonały, ale połączenie kilku znacząco podnosi poprzeczkę dla atakującego.

Dla użytkowników najważniejsza lekcja to nigdy nie ignorować ostrzeżeń o certyfikatach i zawsze używać VPN w publicznych sieciach. Dla administratorów - implementacja HSTS z preload, wymuszanie HTTPS, regularne testy bezpieczeństwa i monitoring podejrzanej aktywności są absolutnym minimum.