Kryptografia i bezpieczeństwo danych - ataki na szyfrowanie z uwierzytelnieniem

Sławomir Samolej
ssamolej.kia.prz.edu.pl
ssamolej@prz.edu.pl

Błędy we wcześniejszych wersjach TLS (przed wersją TLS 1.1)

IV dla CBC było przewidywalne: (chained IV)

IV dla następnego rekordu było ostatnim blokiem bieżącego szyfrogramu.

Powodowało to brak bezpieczeństwa na atak z wybranym tekstem jawnym (CPA). (praktyczny atak opisany jako: BEAST attack)

Możliwość przewidzenia padu: podczas odszyfrowywania

jeśli pad był zły wysłanie alertu decryption_failed

jeśli MAC nie był zweryfikowany wys. alertu bad_record_mac

⇒ atakujący dowiaduje się czegoś o danych szyfrowanych (można przeprowadzić atak, będzie omówiony dalej)

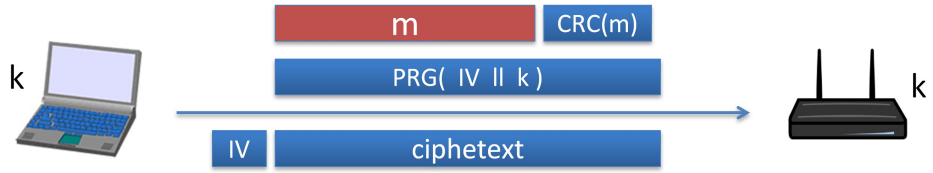
Lekcja: kiedy odszyfrowywanie zgłasza błąd, nie informuj dlaczego nastąpił

2

Starsze wersje TSL (poniżej wersji 1.1) miały wiele błędów. O pierwszym, już kiedyś mówiliśmy. Okazało się, że jako kolejny IV był brany ostatni blok poprzedniego szyfrogramu. Jeśli atakujący dostrzegł tę lukę, to system przestawał być semantycznie bezpieczny (można było na nim wykonać atak z wybranym tekstem jawnym). Jeden z dotkliwych ataków na tak niedopracowany system został opublikowany jako BEAST atak. W wersji TLS 1.1 wyeliminowano tę wadę powodując, że IV wybierane są jako wartości pseudolosowe. Drugą wadą, jaką wykryto we wcześniejszych wersjach protokołu TLS (1.0) było zwracanie różnych wartości błędu, w zależności na jakim etapie odszyfrowywania następował. Jeśli wykryto uszkodzenie pad, wysyłany był komunikat decryption_failed, jeśli nie można było zweryfikować mac, to odsyłano bad_record_mac. Atakujący wtedy wiedział, czy wiadomość ma prawidłowy, czy nieprawidłowy pad i na bazie tylko takiej wiedzy skonstruowano specjalny atak zwany „padding attack". Rozwiązanie tego problemu w wersji 1.1 polegało na zastosowanie jednego komunikatu, niezależnie, czy wykryty jest błąd pad, czy błąd weryfikacji MAC. Ważną lekcją z wykrycia tej luki jest to, że nie wolno naszemu systemowi szyfrującemu pozwolić na odsyłanie informacji, dlaczego odszyfrowywanie się nie udało. To jest trochę nienaturalny odruch, jeśli spojrzymy na ten problem z punktu widzenia budowanie komunikacji w sieciach komputerowych. Ale w systemach ochrony danych takie zachowanie może prowadzić do otwarcia drogi do zaatakowania.

802.11b WEP: jak tego nie robić

802.11b WEP:



Wcześniej dyskutowane problemy:

powtarzanie się IV i powiązane ze sobą klucze

3

Przyjrzyjmy się jeszcze jednemu złamanemu protokołowi: 802.11b WEP. W tym protokole mamy wiadomość m, którą chcemy wysłać do punktu sieci bezprzewodowej. Na początku obliczana jest „zwyczajna” suma kontrolna. Wiadomość i CRC są szyfrowane z zastosowaniem szyfru strumieniowego RC4. Można zauważyć, że klucz szyfrowania jest połączeniem IV (które jest zmieniane dla każdego pakietu) i długoterminowego klucza k. Następnie IV i szyfrogram są przekazywane do odbiorcy. Do tej pory omawiając ten nieudany protokół wspominaliśmy o dwóch wadach: IV się powtarzał, a nie powinien oraz zastosowane klucze były ze sobą powiązane (miały wspólny k). Na takie klucze nie jest zupełnie przygotowany algorytm RC4, który wtedy łatwo złamać. W protokole nożna znaleźć jeszcze jedną lukę, a mianowicie zastosowanie sumy kontrolnej (CRC), która pozwala na fałszowanie danych.

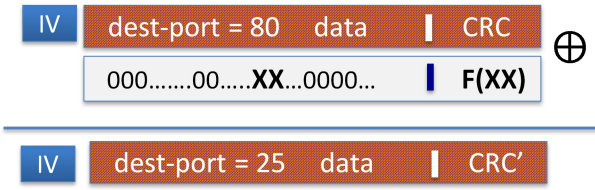
Aktywny atak na WEP

Fakt: CRC jest liniowa,czyli ∀m,p: **CRC(m ⊕ p) = CRC(m) ⊕ F(p)**

Szyfrogram WEP:

atakujący:

XX = 25⊕80



Po odszyfrowaniu: CRC jest prawidłowa, ale szyfrogram sfałszowany!!

4

Właściwość CRC, która robi z niej punkt ataku jest jej liniowość. Okazuje się, że jeśli mamy obliczone CRC z m i chcemy obliczyć CRC(m XOR p) to jest to łatwe do obliczenia następującym sposobem. Wystarczy wartość CRC(m) dodać XOR do łatwego przekształcenia F wartości p. Bazując na tej właściwości CRC atak może wyglądać następująco: Wiedząc, że w zaszyfrowanych danych jest w pewnym miejscu zaszyty nr portu i suma kontrolna przygotowuje specjalne słowo z wartościami 25 XOR 80 i F(25 XOR 80) i wykonuje operację XOR na tym słowie i szyfrogramie. Jeśli stosowany był szyfr strumieniowy (a tak jest w WEP), to wykonanie pokazanego na slajdzie XOR z porcją szyfrogramu skutkuje zamianą jednej wartości w szyfrogramie na inną. W rezultacie szyfrogram staje się sfałszowany, pomimo, że próbowano wykryć ingerencję w treść szyfrowanej wiadomości. Tak naprawdę nie zmieniane są tu zaszyfrowane dane, tylko inny fragment szyfrogramu, ale port.

Atak na padding CBC

Krótkie przypomnienie

- **Szyfrowanie z uwierzytelnieniem**: bezpieczeństwo na atak z jawnym tekstem + zapewnienie integralności
  - Zapewnia poufność w przypadku aktywnych ataków.
  - Zapobiega atakom z wybranym szyfrogramem
- Ograniczenia: nie jest odporne na atak powtórzeniowy oraz na błędnie opracowane implementacje
- Tryby szyfrowania z uwierzytelnieniem:
  - Standardy: GCM, CCM, EAX
  - Ogólna konstrukcja: encrypt-then-MAC

6

TLS record protocol (szyfrowanie CBC)

Odszyfrowywanie: **dec($k_{b\to s}$, record, $ctr_{b\to s}$)** :

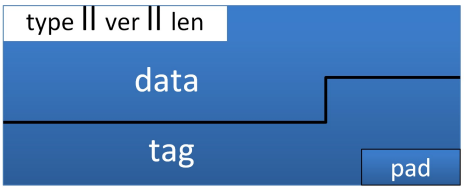
krok 1: odszyfrowanie rekordu ze schematu CBC z kluczem k_{enc}

krok 2: sprawdzenie formatu padu: przerwij, jeśli błąd

krok 3: sprawdzenie tagu z [++$ctr_{b\to s}$ || header || data]
przerwij, jeśli błąd

Dwa typy błędówr:

- **błąd paddingu (padding error)**
- **błąd MAC (MAC error)**



7

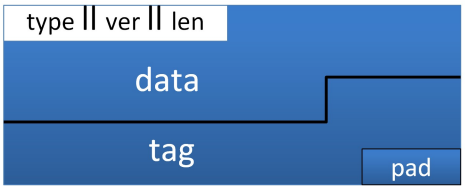
Przypominamy proces odszyfrowywania ze slajdu 25 z poprzedniej prezentacji. Ważna właściwość: rozpoznajemy dwi sytuacje, kiedy odrzucamy szyfrogram: zły format paddingu i zły tag. W poprawnym rozwiązaniu oba błędy nie powinny być rozróżnialne (zgłaszamy 1 kod błędu)

Przewidywanie paddingu

Załóżmy, że atakujący jest w stanie rozróżnić błędy zgłoszone w procesie odszyfrowywania (inny błąd paddingu, inny błąd MAC) :

⇒ **Przewidywanie paddingu**:
atakujący wysyła szyfrogram i dowiaduje się czy ostatnie bajty wiadomości zawierały poprawny pad

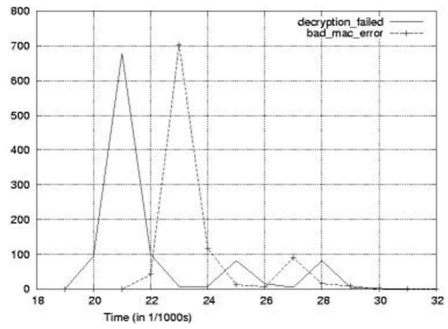
Dobry przykład
ataku z wybranym szyfrogramem



8

Wyjaśnijmy, jak taki atak mógłby być przeprowadzony. Załóżmy, że atakujący rozróżnia błędy odszyfrowania. Wie że błąd zgłoszony został na etapie sprawdzania paddingu i wie, że inny błąd jest zgłoszony na etapie weryfikacji MAC. Możliwy atak nosi nazwę „przewidywanie paddingu". Atakujący przechwycił szyfrogram i chce odszyfrować tajną wiadomość. Może on wysyłać do serwera spreparowane przez siebie szyfrogramy. Jeśli szyfrogram ma zły padding, to do stanie o tym informacje, jeśli zaś szyfrogram ma dobry padding, to otrzyma inną informację np. mówiącą o złym tagu. Powstaje pytanie, czy dysponując tylko taką informacją można odszyfrować szyfrogram. Jest to swoją drogą bardzo dobry przykład ataku z wybranym szyfrogramem.

Przewidywanie paddingu poprzez analizę czasową OpenSSL



Udostępnione przez: Brice Canvel

(Naprawione w OpenSSL 0.9.7a)

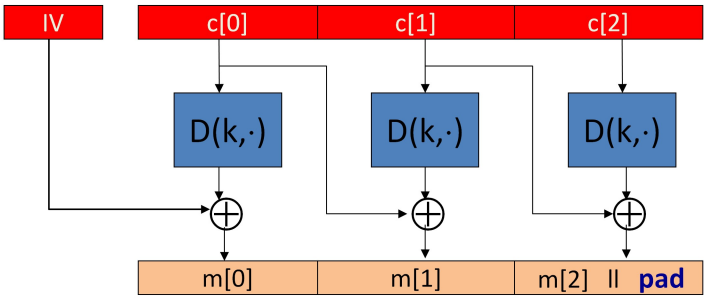
W starszym TLS 1.0:
można przewidywać padding z powodu różnych komunikatów odsyłanych po weryfikacji wiadomości.

9

Pomimo, że w starszych wersjach TSL dostrzeżono możliwość zaatakowania systemu ze względu na rozpoznawanie przyczyn odrzucania szyfrogramów, a potem zmieniono odpowiedź funkcji szyfrującej w taki sposób, że zawsze zgłaszała tylko jeden rodzaj błędów, to dalej można było przeprowadzić atak. Tym rodzajem ataku był atak czasowy. Na slajdzie znajduje się nazwisko osoby, która i opracowała i opublikowała ten atak. Jeśli spojrzeć na sekwencję sprawdzania poprawności wiadomości, to po jej odszyfrowaniu, szybko (bo wymaga to mało obliczeń) stwierdza się poprawność pad (i ew. zgłasza błąd) lub po pewnym czasie (potrzebnym do weryfikacji MAC) zgłasza się ew. drugi błąd. Analizując czas odpowiedzi tak skonstruowanego programu można zauważyć, że jeśli szyfrogram miał zły pad, to zgłoszenie błędu następowało po ok. 21 milisekundach, a jeśli weryfikacji nie przechodził tag, to po ok. 23 milisekundach. Pomimo zgłaszania tego samego kodu błędu atakujący mógł dalej rozróżniać, kiedy błąd wystąpił na weryfikacji pad, a kiedy na weryfikacji tag. W dalszym ciągu możliwa jest deszyfracja szyfrogramu.

Używanie przewidywania paddingu (szyfrowanie CBC) (1)

Atakujący dysponuje szyfrogramem **c = (c[0], c[1], c[2])**
i chce odszyfrować **m[1]**

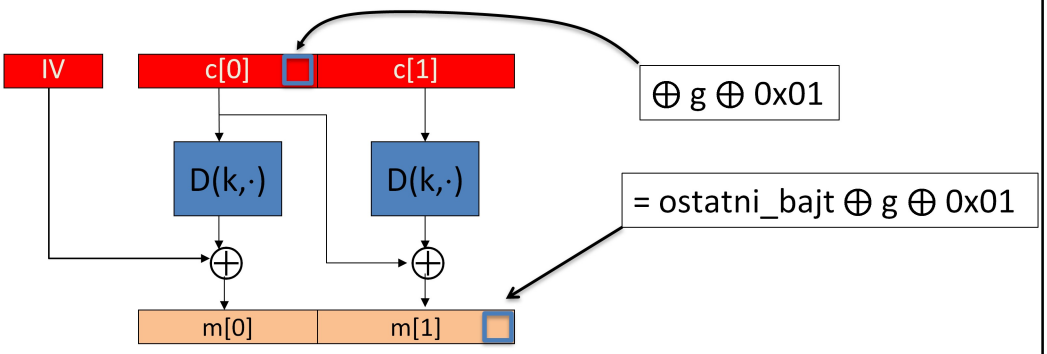


10

Odszyfrowanie szyfrogramu przy uzyskaniu takiej wiedzy, jak pokazane wcześniej jest możliwe i może nastąpić w pokazany teraz sposób.
Atakujący dysponuje szyfrogramem c , jak na slajdzie i chce odszyfrować fragment wiadomości m[1]. Proszę zwrócić uwagę, że podczas odszyfrowywania jednego z bloków szyfrogramu w trybie CBC bierze się poprzedni blok szyfrogramu (c[0]) i obecny blok szyfrogramu (c[1]). Obecny blok szyfrogramu deszyfruje się z zastosowaniem alg. odszyfrowującego, a następnie wykonuje się XOR pomiędzy wcześniejszym blokiem szyfrogramu (c[0]) i odszyfrowanym obecnym blokiem szyfrogramu (D(k,c[1]).

Używanie przewidywania paddingu (szyfrowanie CBC) (2)

krok 1: niech **g** będzie próbą zgadnięcia ostatniego bajtu m[1]



Jeśli ostatni_bajt = g: prawidłowy pad

W przeciwnym wypadku: błędny pad

11

Zaczynamy od odrzucenia w analizie bloku c[2]. Zakładamy, że przewidzieliśmy ostatni bajt fragmentu wiadomości m[1] i nazywamy go g. To jest liczba z przedziału 0 do 255. Wtedy z przedostatnim blokiem szyfrogramu (c[0]) wykonujemy operację XOR g XOR 0x01. Ponieważ g jest „zgadniętą" wartością ostatniego bajta, to po wykonaniu pełnego odszyfrowywania bloku c[1] w ostatnim bajcie tak spreparowanej wiadomości jest prawidłowy pad = 0x01. Teraz możemy zweryfikować nasze zgadywanie. Jeśli dobrze zgadliśmy, to system sprawdzający wiadomość „przepuści" nas przez weryfikację pad, ale pewnie zgłosi błąd podczas weryfikacji tag. Jeśli nie, to musimy próbować podać inną wartość g.

Używanie przewidywania paddingu (szyfrowanie CBC) (3)

Atak: wyślij **(IV, c'[0], c[1])** do przewidywania padu

⇒ atakujący dowiaduje się, czy ostatni-bajt = g

Powtarzaj dla g = 0,1, …, 255 aby w końcu poznać ostatni bajt m[1]

Potem zastosuj pad (02, 02) aby odczytywać kolejne bajty i tak dalej …

12

Scenariusz ataku można teraz podsumować w następujący sposób. Atakujący wysyła zmodyfikowany szyfrogram postaci (IV, c'[0], c[1]) do mechanizmu przewidywania padu. Dzięki możliwości odczytania informacji, czy weryfikacja padu przeszła prawidłowo dowiaduje się, czy „zgadnięte" g było prawidłowe. Przewidywanie padu można prowadzić systematycznie podając kolejno jako g wartości 0, 1, 2,…,255. Dla którejś z nich otrzymamy informację, że pad jest właściwy. Wtedy naprawdę zgadliśmy jeden bajt bloku wiadomości m[1]. Teraz możemy zająć się zgadywaniem w podobny sposób kolejnych bajtów bloku m[1]. Wystarczy, że z c[0] wykonamy XOR z wyrażeniem, które po rozpoznaniu drugiego bajta m[1] da pad (02,02)… Algorytm wykrywania kolejnych bajtów wiadomości można kontynuować, aż do odszyfrowania całego bloku. Dla jednego bloku trzeba wykonać maksymalnie 16 x 256 prób.

Protokół IMAP w oparciu o TLS

Problem: TLS renegocjuje klucze, kiedy otrzyma błędny rekord

Weźmy IMAP bazujący na TLS: (protokół do odczytu poczty)

- Co 5 minut klient loguje się do serwera pocztowego :
  LOGIN "username" "password"
- Dokładnie ten sam atak pracuje niezależnie od stosowania nowych kluczy
  ⇒ można uzyskać hasło w ciągu kilku godzin.

13

Można powiedzieć, że omówiony wcześniej atak nie ma racji bytu, ponieważ TLS, kiedy tylko napotka błąd, to zrywa sesję, negocjuje nowe klucze i komunikacja rozpoczyna się od początku. Atakujący zostaje z jedną próbą zgadnięcia 1 bajta w części wiadomości…

Okazuje się jednak, że jeśli w systemie istnieje taka luka, to może się ona objawić w zastosowaniu protokołu do np. innych protokołów wyższego poziomu, jak na przykład IMAP (jeden z dwóch popularnych protokołów do komunikacji z serwerami pocztowymi).
Często protokół IMAP bazuje na implementacji protokołu TLS. Okazuje się, że co np. 5 minut klient IMAP łączy się do serwera i sprawdza, czy nie przyszła nowa wiadomość. Komunikacja z serwerem pocztowym zaczyna się zawsze od zalogowania (przesłania nazwy i hasła), a następnie sprawdzenia, czy nie ma nowej wiadomości w skrzynce. Tu pojawia się problem! Co 5 minut atakujący może przechwycić szyfrogram, w którym wie, że na początku znajdują się dane do logowania i … co pięć minut może przeprowadzić ten jednokrotny atak na zgadnięcie jednego bajta hasła. Atak można przeprowadzić pomimo zmiany kluczy, bo dokładnie wiemy, co w wiadomości musi być przesłane…
Okazuje się, że stosując atak czasowy można wykraść hasło 8-znakowe do email w ciągu kilku godzin.
Zabezpieczeniem przed takim atakiem jest zawsze sprawdzanie MAC w algorytmie deszyfracji TLS, niezależnie, czy sprawdzanie pad zwraca błąd, czy nie. Wtedy atakujący nie dostaje tych „szybkich”, czy „wolniejszych odpowiedzi i nie jest w stanie zidentyfikować, czy to PAD, czy TAG był niepoprawny.

Lekcje

1. Zastosowanie schematu Encrypt-then-MAC pozwala na uniknięcie omówionego wcześniej problemu implementacyjnego:

wtedy MAC jest sprawdzany jako pierwszy, a dopiero wtedy następuje odszyfrowywanie i sprawdzanie padu

2. Konstrukcja MAC-then-CBC zapewnia szyfrowanie z uwierzytelnieniem, ale możliwość przewidywania padów niszczy tę konstrukcję

3. Jakie korzyści otrzymalibyśmy, gdyby TLS było zaprojektowane na bazie schematu MAC-then-CTR?
   - Atak na padding nie byłby możliwy, ponieważ w tym trybie szyfrowania nie stosuje się paddingu.

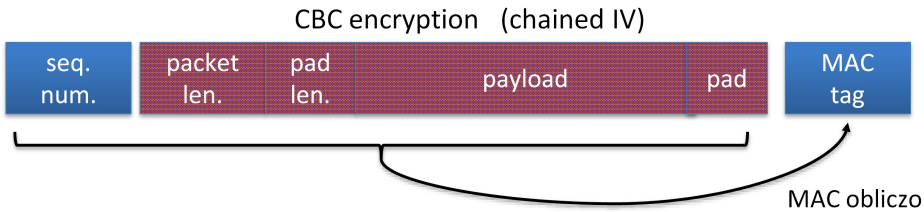
14

Nasza dyskusja o lukach w TLS nie miałaby miejsca, gdyby w rozwiązaniu zastosowane było podejście Encrypt-then-MAC. Wtedy sprawdzanie tagu szyfrogramu odbywałoby się pierwsze. Zgłoszenie wtedy błędu nie miałoby znaczenia, bo nie mamy jeszcze wtedy dostępu do padu (jest on zaszyty w szyfrogramie, który jeszcze nie zostało deszyfrowany). Poprawna weryfikacja MAC pozwoliłby wtedy rozpoczęcie deszyfracji wiadomości. Jakakolwiek modyfikacja szyfrogramu byłaby wychwycona od razu przy sprawdzaniu MAC.
Konstrukcja MAC-then-CBC spełnia warunki szyfrowania z uwierzytelnieniem, ale tylko wtedy, gdy nie ujawnia, dlaczego odszyfrowywanie zakończyło się niepowodzeniem, zarówno jeśli chodzi o zwracane kody błędu (dopuszczalny jest tylko jeden) ,jak i czas przeprowadzania weryfikacji (pomimo błędu pad trzeba policzyć MAC, żeby czas obliczeń nie zdradził, na którym etapie weryfikacji szyfrogramu nastąpiło zgłoszenie błędu).
Gdyby w TLS zastosowano schemat MAC-then-CRT, to konstrukcja byłaby bezpieczniejsza, ponieważ w schemacie CRT nie stosuje się paddingu.

Atak na nie-atomowe odszyfrowywanie

15

SSH Binary Packed Protocol



MAC obliczony dla tekstu niezaszyfrowanego

Odszyfrowywanie:

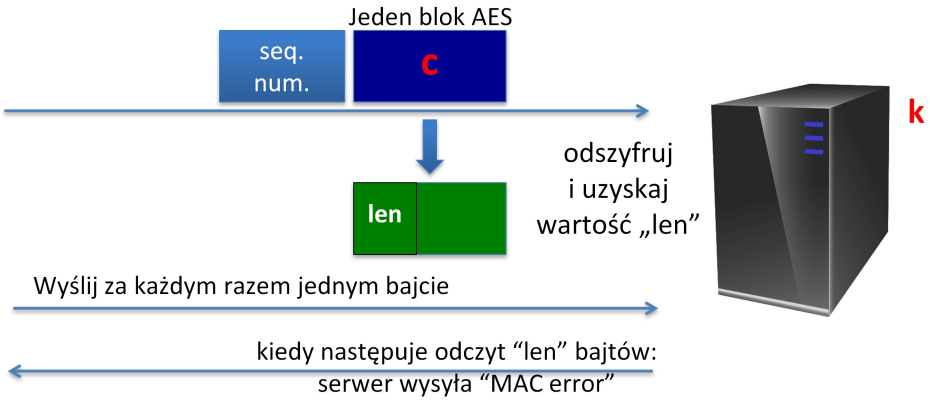
- Krok 1: odszyfruj tylko długość pakietu (!)
- Krok 2: odczytaj tyle danych ile podano w polu długość pakietu
- Krok 3: odszyfruj pozostałe bloki szyfrogramu
- Krok 4: sprawdź MAC i wyślij błąd, jeśli MAC się nie zgadza

16

Protokół SSH (ang. Secure Shell) stosowany jest do łączenia się do zdalnych pulpitów, pracuje w strukturze klient-serwer. Po dokonaniu wymiany kluczy następuje wymiana komunikatów z zastosowaniem protokołu „Binary Packed Protocol". Schemat szyfrowania z uwierzytelnieniem, jaki jest tu zastosowany to Encrypt-and-MAC. Każdy pakiet SSH zaczyna się od numeru sekwencyjnego, potem zawiera długość pakietu oraz długość pad CBC. Kolejne dwa pola to dane oraz pad. Pole zaznaczone na czerwono jest szyfrowane zgodnie ze schematem CBC z IV, które jest powiązane z poprzednim pakietem danych (tzw. chained IV, co samo w sobie jest podatne na atak z wybranym tekstem jawnym, ale tym atakiem się teraz nie zajmujemy). Ostatecznie dla całego tekstu jawnego jest obliczany MAC i dołączany na koniec pakietu protokołu. Przypomnijmy, że takie rozwiązanie jest uważane za jedno z najgorszych, ponieważ stosujemy podejście Encrypt-and-MAC, a MAC jest liczony z jawnego tekstu (MAC nigdy nie było projektowane do zachowania poufności). Zachodzi obawa, że MAC tak policzony może zawierać w sobie informacje na temat jawnego tekstu. Jednak, nie taką właściwość omawianej konstrukcji będziemy brać pod uwagę.
Rozważmy metodę odszyfrowywania pakietów SSH. Na początku odszyfrowywane jest tylko pole z długością pakietu. Potem odczytywane jest tyle danych ile wynosi długość pakietu. Z kolei pobrane dane są odszyfrowywane. Z całości pakietu obliczany jest MAC. Protokół zwraca błąd, jeśli MAC się nie zgadza. Tutaj problemem jest pierwszy etap odszyfrowywania wiadomości. Następuje odszyfrowanie pola długości, a potem bez żadnej weryfikacji jego MAC odbywa się dalsze pobieranie danych. Taka metoda odszyfrowywania okazała się podatna na atak, który w pewnym uproszczeniu zostanie zaraz pokazany.

Atak na pole z długością pakietu (uproszczony)

Atakujący ma jeden blok szyfrogramu **c = AES(k, m)** i chce pozyskać **m**



atakujący uczy się o 32 początkowych bitach m !!

Jak można by efektywnie poprawić projekt SSH?

- Można wysyłać pole „len” niezaszyfrowane
- Można dodać osobny MAC dla pola „len” w ramce protokołu

17

Załóżmy, że atakujący przechwytuje **jeden** szczególny blok szyfrogramu zaszyfrowany z zastosowaniem algorytmu AES i chce odszyfrować dane w nim ukryte. Tworzy więc sfałszowaną wiadomość, która składa się z numeru sekwencyjnego oraz tego jednego bloku szyfrogramu (c). Protokół SSH oczekuje teraz ciągu bloków danych o liczbie zaszyfrowanej w bloku c. Atakujący wysyła więc dane bajt po bajcie i czeka na reakcję serwera. Serwer teraz „gromadzi” bajty w celu złożenia danych do odszyfrowania. Po „zgromadzeniu” bajtów następuje obliczenie MAC, które oczywiście się nie udaje, i serwer zwraca błąd MAC. Zauważmy, że atakujący wie dokładnie ile bajtów wysłał do serwera zanim otrzymał błąd. W rezultacie liczba przesłanych bajtów jest równa odszyfrowanej 32-bitowej wiadomości m, którą atakujących chciał zdobyć. Jako wejście ataku można podać jakikolwiek blok szyfrogramu i otrzymać jego jawną postać.
Jakie popełniono błędy w tym projekcie systemu kryptograficznego? Po pierwsze odszyfrowywanie nie było „operacją atomową”, czyli jednorazową nieprzerwaną akcją na szyfrogramie, która ma dać w odpowiedzi odszyfrowaną wiadomość lub błąd. Tutaj następuje odszyfrowywanie w kilku etapach – najpierw odszyfrowywanie pola długości, a potem zgromadzenie odpowiedniej liczby danych i dalsze ich odszyfrowywanie. Takie postepowanie, jak się okazuje nie jest bezpieczne, a w naszym przykładzie pozwoliło na złamanie szyfrowania z uwierzytelnieniem. Drugim błędem było zastosowanie pola długości bez jego uprzedniego uwierzytelnienia. Tak na prawdę odszyfrowywanie nie powinno mieć miejsca zanim nie potwierdzi się jego autentyczności (nie sprawdzi się jego MAC).
Jak można by „uodpornić” protokół SSH na ten rozdaj ataku? Okazuje się, że wystarczyłoby nie szyfrować pola „len”, lub dodać MAC liczony tylko dla tego pola w opisie protokołu.
Podsumowując rozważania na tym etapie, należy stosować schematy szyfrowania z uwierzytelnieniem zgodne ze standardami. Jeśli nie, to przynajmniej należy stosować schemat Encrypt-then-MAC i nigdy nie wolno dopuścić, że w systemie są używane odszyfrowane dane ale nie uwierzytelnione.

Literatura uzupełniająca

- The Order of Encryption and Authentication for Protecting Communications, H. Krawczyk, Crypto 2001.
- Authenticated-Encryption with Associated-Data, P. Rogaway, Proc. of CCS 2002.
- Password Interception in a SSL/TLS Channel, B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux, Crypto 2003.
- Plaintext Recovery Attacks Against SSH, M. Albrecht, K. Paterson and G. Watson, IEEE S&P 2009
- Problem areas for the IP security protocols, S. Bellovin, Usenix Security 1996.