Bezpieczeństwo komputerowe

Lista nr 4 (A)

Pytania, które będą zadawane oddającym dowolne zadanie z tej listy. Nieznajomość odpowiedzi na którekolwiek z poniższych zagadnień może skutkować odebraniem (wszystkich) punktów.

- czym jest certificate pinning? dlaczego się go stosuje?
- czym jest *Extended validation* dla certyfikatów SSL?
- kto da się nabrać na taki atak (kontekscie zadania 3)?
- czym są CRL, OCSP?
- co się stanie, gdy ktoś pozna klucz tajny serwera www?
- co się stanie, gdy ktoś pozna klucz tajny CA, który podpisywał certyfikat serwera www?
- co się stanie, gdy ktoś pozna klucz tajny jakiegoś CA?
- co się stanie, gdy pewne CA wydaje certyfikaty w oparciu o słabe funkcje haszujące np. MD5?
- czym są downgrade attacks na TLS?
- czym jest HTTP Strict Transport Security (HSTS)?

Zadanie 1 (5 pkt) Wykonaj wszystkie czynności:

1. wygeneruj (np. korzystając z *openssl* klucz $\mathcal{A}$ służący do podpisywania (wybierz pomiędzy DSA a RSA), np. odpowiednio zmodyfikuj komendę:

   ```
   openssl genrsa -out privkeyA.pem,
   ```

   pamiętaj aby wybrać długość klucza na poziomie bezpieczeństwa odpowiadającej co najmniej 112-bitów (zobacz: NIST SP 800-57).
2. wygeneruj żądanie certyfikatu (CSR - certificate signing request):

   ```
   openssl req -new -key privkeyA.pem -out certA.csr.
   ```

Powtórz czynności $1-3$, generując klucz $\mathcal{B}$, ale tym razem utwórz certyfikat "self signed", tj. zostań *CA - certificate authority*;

```
openssl req -new -x509 -key privkeyB.pem -out CAcert.crt -days 15,
```

a następnie kluczem $\mathcal{B}$ wygeneruj certyfikat dla klucza $\mathcal{A}$ z pliku CSR:

```
openssl x509 -req -days 45 -in certA.csr -CA CAcert.crt -CAkey privkeyB.pem -set_serial 01 -out certA.crt.
```

Zadanie 2 (5 pkt) Wykonaj wszystkie czynności:

1. Zainstaluj certyfikat odpowiadający kluczowi $\mathcal{B}$ *CA* w przeglądarce (w sekcji "authorities"). Nie importuj jednak certyfikatu dla klucza $\mathcal{A}$.

2. Na swoim prywatnym serwerze www (może to być Twój komputer) "zainstaluj" klucz $\mathcal{A}$ i certyfikat (*certA.crt*), aby działał adres np. "https://www.moj.serwer.pl".
3. Spraw, aby przeglądarka nie zgłaszała ostrzeżenia (w zadaniu pierwszym musisz podczas generowania *CSR* wpisać odpowiedni adres (*Common Name/hostname*) - może to być np. localhost, albo adres uzyskany przez usługi typu *dynamic dns*).

Porównaj wydajność strony z wykorzystaniem protokołu http z wersją szyfrowaną (https). Skorzystaj np. z Apache Benchmark (`https://httpd.apache.org/docs/2.4/programs/ab.html`). Która część protokołu jest odpowiedzialna za spadek wydajności, jak bardzo?

Zadanie 3 (20 pkt) Stwórz stronę "phishingową" działającą na twoim serwerze (np. w maszynie wirtualnej) przechwytującą wprowadzane hasła np. do serwera poczty studenckiej/Gmaila/... wykonaj w tym celu czynności z zadań 1 i 2.

- strona musi działać w oparciu o protokół https,
- przeglądarka ma akceptować certyfikat.

W tym celu możesz odpowiednio zmodyfikować lokalny plik z hostami (*/etc/hosts* bądź jego odpowiednik w systemie, z którego korzystasz).

Które pary strona-przeglądarka są odporne na taki atak (certificate pinning)?

Zadanie 4* (10 pkt) Skonfiguruj serwer (działający pod adresem IP np. 123.123.123.123) w ten sposób, aby wyświetlał stronę pod wpisanym adresem (np. https://www.moj.serwer.pl). Stwórz program, który nasłuchuje kanał radiowy i w momencie, gdy jakiś komputer z lokalnej sieci wysyła zapytanie DNS o predefinionwany adres (www.moj.serwer.pl), to Twój program wysyła sfałszowaną odpowiedź DNS (m. in. z odpowiednio zmodyfikowanymi polami nadawcy, właściwym Transaction ID i odpowiedzią wskazującą na 123.123.123.123).

Możesz wykorzystać istniejące narzędzia (np. tcpdump) do filtrowania wysyłanych w sieci pakietów.

[10 pkt] Napisz własny program, który tworzy odpowiedni pakiet odpowiedzi i go wysyła.

[5 pkt] Wykorzystaj dostępne rozwiązania (np. hping, scapy).