Bezpieczeństwo Komputerowe - kolokwia

dr Filip Zagórski

opracował: Mikołaj Pietrek

Semestr zimowy 2018/2019

Termin dziesięciotygodniowy

Zadanie 1. SQL Injection - jako przykład prosta, podatna funkcja autoryzująca napisana w PHP:

1. Jak może wyglądać atak?
2. jak się przed tym zabezpieczyć?

Zadanie 2. Zabezpieczanie haseł funkcjami haszującymi:

1. Dlaczego to działa?
2. Czy sól powinna być losowa?
3. Czy długość soli wpływa na bezpieczeństwo?

Zadanie 3. Memory hard functions:

1. Co to jest?
2. Wymień 2 przykłady zastosowania.
3. Dlaczego to jest skuteczne?

Zadanie 4. (Zadanie z Timing Attack z poprzedniego roku)

Semestr zimowy 2017/2018

Termin dziesięciotygodniowy (identyczny z piętnastotygodniowym?)

Zadanie 1. Czy następujące stwierdzenia są prawdziwe? Krótko uzasadnij.

1. „Security through obscurity" (bezpieczeństwo przez zaciemnianie) odnosi się do praktyki ukrywania hasła użytkownika podczas jego wprowadzania, aby nikt nie mógł go zobaczyć na ekranie.
2. Jeżeli jutro Domino odkryje efektywny algorytm obliczania największego wspólnego dzielnika dla dwóch bardzo dużych liczb, to pozwoli na złamanie RSA.
3. Każdy schemat szyfrowania dla którego rozmiar przestrzeni kluczy jest równy rozmiarowi przestrzeni wiadomości i dla którego klucz wybierany jest jednostajnie jest „perfectly secret" (doskonale tajny).

Zadanie 2. Czy następujące stwierdzenia są prawdziwe? Krótko uzasadnij.

1. Dla kryptograficznej funckji haszującej h nie istnieje takie x, że $h(x) = y$.

2. Powiemy, że funkcja haszująca h ma kolizję jeżeli istnieje para $(x, y), x \neq y$ taka, że $h(x) = h(y)$.

3. Funkcje haszujące wykorzystywane są w schematach podpisów cyfrowych.

Zadanie 3. Określenie „Secured by 256 bit SSL" oznacza:

1. Do uzgadniania klucza sesyjnego używa się kluczy RSA długości 256 bitów.

2. Identyfikator sesji SSL jest 256 bitowy.

3. Długości bloków kryptogramów AES wynoszą 256 bitów.

Zadanie 4. Alicja i Bob rozmawiają przez telefon, chcą losowo wybrać czy idą do kina czy na mecz. Zaproponuj protokół gwarantujący uczciwy wybór, aby żaden z uczestników nie miał przewagi. Możesz wykorzystać: szyfrowanie, uzgadnianie kluczy Diffiego-Hellmana, zobowiązania bitowe...

Zadanie 5. Czy w przypadku szyfrowania symetrycznego może się zdarzyć, że szyfrując dwa różne teksty dwoma różnymi kluczami otrzyma się ten sam kryptogram? Odpowiedź uzasadnij.

Zadanie 6. Odpowiedz na każde z pytań (prawda/fałsz) oraz uzasadnij jednym zdaniem odpowiedź.

1. Alicja wygenerowała parę kluczy RSA i opublikowała klucz publiczny. To wystarczy Alicji do wysłania Bobowi bezpiecznie zaszyfrowanego maila.

2. Niech E_k będzie bezpiecznym szyfrem blokowym. Wtedy następujący schemat szyfrowania jest odporny na chosen-plaintext attack: wysyłając wiadomość m, wysyłający wybiera losową wartość r i wysyła dwa ciągi r i $E_k(r) \oplus m$

3. Okazało się, produkt zabezpieczony kluczem 64 bitowym jest podatny na replay attack. Postanowiono wydłużyć długość kluczy do 256 bitów. Po tej poprawce system jest bezpieczny.

Zadanie 7. Poniższy kod (php) przedstawia generowanie certyfikatu po stronie serwera. Co może pójść nie tak?

```
$passphrase = $_POST["a"];
$tmpfname = tempnam("/tmp","kek");

$str = "openssl genrsa -aes256 -passout pass:\"$passphrase\" -out
       $tmpfname 2048 2>/dev/null >/dev/null";

system($str);
$privkeyout = file_get_contents($tmpfname);
unlink($tmpfname);
$pkey = openssl_pkey_get_private($privkeyout, $passphrase)
        or die("Złe hasło do certyfikatu");

//debug($pkey);
//*
$akcja = $this->Session->read('Auth.User.akcja');
//$this->dn["organizationalUnitName"] = $akcja ."/".$this->Session-read('ou');
$dn = array(
"countryName" => "PL",
"stateOrProvinceName" => "Mazowieckie",
"localityName" => "Warszawa",
"organizationName" => "KBW",
```

```
$dn["organizationalUnitName"] = $akcja ."/KBW";
$csr = openssl_csr_new($dn, $pkey);

openssl_csr_export($csr, $csrout);
$this->User->id = $this->Session->read('Auth.User.User.id');
$this->User->save(array('csr' => $csrout, 'cert_status' => 1));
$dateNow = date('Y-m-d H:i:s');
$nameKey = "klucz_asd";//.$this->Session->Read('nr_dn');
header("refresh: 2; url=./license");
header("Content-Type: application/download");
header("Content-Length: " . strlen($privkeyout));
header("Content-Disposition: attachment; filename=\"". $nameKey . ".pem\"");
print($privkeyout);
die();
```

Zadanie 8. W pewnym serwisie hasła przechowywane są w „plaintexcie". Funkcja sprawdzająca równość ciągów jest następująca:

```
public static boolean isEqual(byte digesta[], byte digestb[]) {
    if (digesta.length != digestb.length)
        return false;

    for (int i = 0; i < digesta.length; i++) {
        if (digesta[i] != digestb[i]) {
            return false
        }
    }
    return true;
}
```

Opisz możliwy timing-attack.