Zarządzanie sesją w aplikacjach Internetowych

Kraków, 2008-10-23

Paweł Goleń

Agenda

- Po co sesje w aplikacjach internetowych
- Sposoby przekazywania identyfikatorów
- Sposoby ochrony Cookie
- Analiza identyfikatora sesji
- Ataki: Session fixation, Session adoption
- Dobre praktyki, czyli co robić
- Z innej beczki: **CSRF**

HTTP jest bezstanowy

- Protokół HTTP jest bezstanowy
  - Typowa transakcja
    - Nawiązanie połączenia z serwerem
    - Klient wysyła żądanie
    - Serwer przetwarza żądanie i wysyła odpowiedź
    - Zakończenie połączenia z serwerem

Optymalizacje...

- Nie zmieniają faktu bezstanowości HTTP
- Głównie ze względu na wydajność
  - Nawiązanie połączenia TCP jest „drogie”
    - Keep-Alive
  - Transakcja HTTP jest „droga”
    - ...ale to już inny temat (performance)
      - YSlow (http://developer.yahoo.com/yslow/)

HTTPS jest bezstanowy

- Właściwie „prawie” bezstanowy
- HTTPS to HTTP po SSL
  - Taki sam scenariusz jak dla HTTP
- Dodatkowa warstwa SSL
  - Możliwość wykorzystania ID sesji SSL
    - W praktyce rzadko wykorzystywane
      - Nigdy się nie spotkałem z takim rozwiązaniem

Aplikacje są stanowe

- Śledzenie stanu
  - Użytkownik nieuwierzytelniony
  - Użytkownik uwierzytelniony
- Wiązanie kolejnych żądań użytkownika
  - Operacje złożone z wielu kroków
    - Zapamiętanie stanu z poprzednich kroków
    - Właściwa kolejność kroków

Prosty przykład: sklep

- Użytkownik przegląda ofertę sklepu
- Użytkownik dodaje produkty do koszyka
- Użytkownik dokonuje zakupu
  - Produkty dodane do koszyka
  - Uwierzytelnienie lub założenie konta
  - Zapłata za zakupy
  - Realizacja zamówienia

Rozwiązanie: sesje

- Dla klienta tworzona sesja na serwerze
- Żądania rozpoznawane po identyfikatorze
- Sesje obsługiwane przez środowisko
  - Programista korzysta, a nie implementuje
    - Przykładowe identyfikatory sesji
      - PHPSESSIONID, JSESSIONID, ASPSESSIONID, ASP.NET_SessionId
  - Programista nie wie jak sesja jest realizowana

Przekazywanie ID

- Identyfikatory przekazywane w:
  - Cookie
    - ASP.NET_SessionId=qpkdrdaata2haau2ufiymz45
  - Parametrze (np. pole hidden)
    - W GET
    - W POST
  - URL
    - /viewContainer.do;jsessionid=653(...)7D363A

ID sesji w URL jest ZŁE

- **Ułatwia atak Session fixation**
- Duże prawdopodobieństwo ujawnienia:
  - Logi serwera HTTP, PROXY, historia przeglądarki
  - Nagłówek Referer
    - Nawet do innych serwerów
  - Nietypowe przykłady: wydruk, screenshot
- Jeśli ID sesji w parametrze – użyj POST

...ale czasem konieczne

- Przeglądarka nie akceptuje Cookie
  - Przeglądarki wspierają Cookie
  - Problemem jest użytkownik
    - „Cookie to zagrożenie dla prywatności!”
    - ...i inne twierdzenia „fachowej” prasy
  - Cookie sesyjne NIE JEST składowane,
    - A przynajmniej nie powinno
      - Użytkownicy mają w zwyczaju się nie wylogowywać

Gdy nie ma Cookie

- „Fallback” do identyfikatorów w URL
  - ASP.NET:
    - http://site.com/app/(XXXXXXXXXXX)/home.aspx
  - URL Rewriting
    - ;PHPSESSIONID=
    - ;JSESSIONID=
- Dalej zakładam, że cookie jest

Bezpieczne Cookie

- Cookie == dostęp do sesji (prawie zawsze)
- Cookie sesyjne musi być chronione
- Co trzeba przewidzieć:
  - Ataki XSS
  - Podsłuch ruchu sieciowego
  - Wysłanie do innej ścieżki
  - Wysłanie do innego serwera

Ataki XSS

- Typowy przykład: alert(document.cookie)
- Flaga **HttpOnly**
  - Wprowadzona przez Microsoft w IE 6.0
  - Cookie niedostępne przez skrypty
    - Więcej: https://www.owasp.org/index.php/HTTPOnly
  - Możliwość obejścia: XmlHttpRequest
    - Metoda TRACE na serwerze jest ZŁA

Podsłuch ruchu sieciowego

- Użyj SSL
  - SSL „na chwilę” nie wystarcza!
- Flaga **Secure**
  - Cookie wysyłane **tylko** przez SSL
- Szczególnie istotne gdy wiele serwisów
  - Informacyjny po HTTP
  - Bankowy po SSL

Inny serwer

- Atrybut **Domain**
  - Nie pokazuj Cookie innym (pod)domenom!
  - Domyślnie jest dobrze
    - Brak atrybutu Domain
      - Cookie dostępne tylko dla serwera, który je ustawił
    - Choć z (implementacją) cookies może być różnie
      - Cross-Site Cooking

Inna ścieżka

- Atrybut **Path**
  - Określa ścieżkę, której Cookie dotyczy
  - Bardzo często ustawiany na /
  - Błąd jeśli wiele aplikacji na jednym serwerze
    - http://server.com/app1
    - http://server.com/app2
  - Ogranicz Cookie do właściwiej ścieżki!

Cookie powinno być losowe

- Przykład „z boku”: DNS Cache Poisoning
  - Mała losowość
    - Błędy generatorów PRNG
    - Mały zakres wartości
- W przypadku identyfikatora sesji
  - Odgadnięcie Cookie == przejęcie sesji
  - Odgadnięcie MUSI być trudne
    - Długa wartość losowa, a nie „losowa”

Niby dobrze, a źle

- ID sesji „wyliczany”
  - np. md5(username,IP, timestamp)
    - Nazwę użytkownika można zgadnąć
    - IP można zgadnąć (wyszukać)
    - Timestamp jest „mniej więcej” znany
  - Niebezpieczne dla projektów „publicznych”
    - Wynikowa wartość „wydaje się” losowa
    - Ujawnienie „algorytmu” - losowość ZNIKA

...już lepiej

- Już lepiej:
  - HMAC(md5(username, IP, timestamp), klucz)
    - Klucz losowy
    - Unikalny dla każdej instancji aplikacji
- ID sesji generowane przez środowisko
  - Zwykle są to wartości losowe
  - ...ale zawsze można sprawdzić

Analiza losowości

- **Burp Suite**, moduł Sequencer:
  - http://portswigger.net/suite/
- Przykłady:
  - „prawdziwy random”
  - md5(timestamp)
  - random + timestamp
  - z życia wzięty (PHP)

Co teraz?

- Cookie jest chronione
  - HttpOnly, Secure, Path, Domain,
  - Komunikacja jest szyfrowana,
- Może można ustawić identyfikator?
  - Czasami można...

A ja znam Twoje ID!

- Session Fixation
  - http://www.acrossecurity.com/papers.htm
  - Ustalenie znanej wartości ID sesji
  - Podtrzymanie „życia” sesji
  - Ofiara korzysta ze znanego ID sesji
    - Trzeba ją tylko do tego przekonać
- Session Fixation jest ZŁE!
  - Ponownie: nie przekazuj ID sesji w URL

A Twoje ID będzie...

- ... a Session Adoption jest jeszcze gorsze
- Session Adoption
  - Podobne do Session Fixation
  - Zaakceptowanie przekazanego ID sesji,
  - Stworzenie sesji o wybranym ID,
- Przykład:
  - http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Willis.pdf

Scenariusz ataku: zdalnie

- Aplikacja akceptuje ID sesji z URL
- Atakujący „zastawia pułapki”
  - Maile phishingowe z URL,
  - Linki do aplikacji na innych stronach,
- Atakujący czeka na ofiarę
  - Przejmuje sesję ofiary

Scenariusz ataku: lokalnie

- Współdzielone komputery
  - Kafejka internetowa
  - „małe” firmy
- Atakujący zastawia „pułapkę”
  - Ustalenie Cookie
  - „Zaczajenie” się przy innym komputerze
  - Przejęcie sesji ofiary

Dobre praktyki

- Używaj SSL
- Zmieniaj ID sesji
  - Przy zmianie jej stanu
    - Użytkownik anonimowy → zalogowany
    - Użytkownik zalogowany → wylogowany
      - Niszcz „starą” sesję
  - Przy przejściu z HTTP na HTTPS
  - Przy przejściu z HTTPS na HTTP

Dobre praktyki

- Czas życia sesji
  - Użytkownicy często się nie wylogowują
  - Trzeba dobrać do typu aplikacji
    - Zbyt krótki – irytuje użytkowników
    - Zbyt długi – też źle
      - Możliwy atak denial-of-service
        - Każda sesja zajmuje zasoby serwera

Dalsze utrudnianie życia

- Osadzenie w sesji danych identyfikujących
  - Adres IP,
  - User-Agent,
- Weryfikacja danych przy każdym żądaniu
  - Ograniczenie powierzchni ataku,
    - User-Agent można sfałszować,
    - Adres IP sfałszować trudniej,
  - Nieprawidłowe dane: NISZCZ SESJĘ

Czyść sesję!

- Dane zapisane w sesji trzeba usunąć
  - Czasami oddzielne identyfikatory (cookie)
    - Sesji,
    - Uwierzytelnienia,
  - Dwóch różnych użytkowników w sesji
  - Dane poprzedniego użytkownika zostają w sesji
    - Nowy użytkownik ma „brudną” sesję

Czyść sesję: przykład

- Przykład:
  - Co się stanie jeśli w sesji jest flaga IS_ADMIN
    - Flaga jest ustawiana, jeśli użytkownik to administrator,
    - Jeśli flaga jest ustawiona – funkcje administracyjne,
    - Flaga nie jest czyszczona...
    - Loguje się nowy użytkownik ze starym ID sesji...
    - ...UPS!

Podsumowanie

- Używaj SSL
- Zmieniaj ID sesji
- Dobierz czas życia sesji
- Czyść sesję przed i po użyciu
- Niszcz sesję przy wylogowaniu
- ...to jeszcze nie koniec :)

Coś z zupełnie(?) innej beczki

Cross Site Request Forgery

Czyli jak działać w cudzej sesji, bez jej przejmowania

Wysyłanie Cookie

- Przeglądarka wysyła Cookie automatycznie
  - Gdy je ma, oraz:
    - zgadza się host/domena
    - zgadza się ścieżka
- Cookie współdzielone
  - Przez różne zakładki
  - Przez różne okna (w tym samym procesie)

Przykład: img src=

- Jak „wysłać” żądanie?
  - Umieścić odwołanie na innej stronie
    - <img src=”https://www.bank.pl/przelej?do=mnie”>
  - Przeglądarka pobiera „obrazek”
  - Jeśli Cookie istnieje, zostanie wysłane:
    - Nie pomoże flaga HttpOnly
    - Nie pomoże flaga Secure
    - Nie trzeba zgadywać Cookie
    - „Dobre praktyki” na nic?

Jak się bronić przed CSRF

- Można sprawdzać nagłówek Referer
  - Czasami skuteczne, niezbyt eleganckie
    - Możliwe do obejścia
- Wymaganie losowej wartości w żądaniu
  - „atakujący” nie może jej przewidzieć
  - „atakujący” nie może jej poznać
  - Patrz: CSRFGuard (OWASP)

Pytania?