Grzegorz Górski
Mateusz Wojsa
Zakład Systemów Multimedialnych i Sztucznej Inteligencji
Wydział Elektroniki i Informatyki
Politechnika Koszalińska
ul. J.J. Śniadeckich 2
75-453 Koszalin

Wybrane ataki mające na celu kompromitację danych poufnych oraz metody zapewnienia bezpieczeństwa aplikacji i usług internetowych

Słowa kluczowe: usługi internetowe, ataki sieciowe, kompromitacja danych poufnych

1. Wprowadzenie

Zagrożenia związane z szeroko rozumianym IT dotyczą wszystkich. Ich skala oraz mnogość kierunków z których przychodzą jest duża, dlatego budowa systemu bezpieczeństwa organizacji musi przebiegać na wielu płaszczyznach. Przykładem takich zagrożeń może być m.in. dostęp osób nieuprawnionych, awaria lub utrata sprzętu, modyfikacja danych przez osoby nieuprawnione, czy nawet przypadkowe ich usunięcie bądź udostępnianie.

Wycieki danych zdarzają się niestety coraz częściej i są dużym problemem dla organizacji, zwłaszcza gdy ich skala jest ogromna. Niestety nigdy nie jesteśmy w stanie stwierdzić na ile i czy nasz system bezpieczeństwa jest szczelny, jednakże możemy skutecznie je ograniczać. Skuteczna ochrona danych poufnych organizacji wiąże się z zapewnieniem bezpieczeństwa na trzech poziomach:

- prawnym – zgodność z przepisami prawa, poziomu świadczenia usług (SLA), precyzyjnie określony zakres odpowiedzialności;
- ekonomicznym – stabilność, dostęp do zasobów, pewność oraz bezpieczeństwo transakcji;
- organizacyjnym: szkolenia, audyty, procedury, zabezpieczenia fizyczne oraz sieciowe.

2. Ataki mające na celu kompromitacje danych poufnych

2.1. SQL injection

SQL injection polega na tym, że atakujący próbuje wstrzyknąć swój kod SQL podczas wykonywania się w aplikacji innego zapytania, które nie ma odpowiedniego formatowania.

Przykładowe zapytanie SQL sprawdzające poświadczenia użytkownika:

SELECT 1 FROM USERS u WHERE u.login = ‘userLogin’ and u.password =’userPasword’

Gdzie userLogin, userPasword to place holdery, które podmieniane są na wartość z formularza.

W loginie użytkownika można wpisać np. jeżeli posiadamy użytkownika admin

admin’;--

W aplikacji nie posiadającej odpowiednich zabezpieczeń na tego typu atak, spowoduje to wygenerowanie zapytania:

SELECT 1 FROM USERS u WHERE u.login = ‘admin’;--' and u.password =’’;

Podobny efekt możemy uzyskać wpisując ‘ or 1=1;-- w pole hasło gdyby pole login było zabezpieczone ale znamy login użytkownika:

SELECT 1 FROM USERS u WHERE u.login = ‘admin’ and u.password =’’ or 1=1;--';

Takie zapytania zwrócą 1 więc framework zostanie poinformowany o tym, że w bazie zapisane są poprawne poświadczenia i dany użytkownik zostanie zalogowany bez hasła.

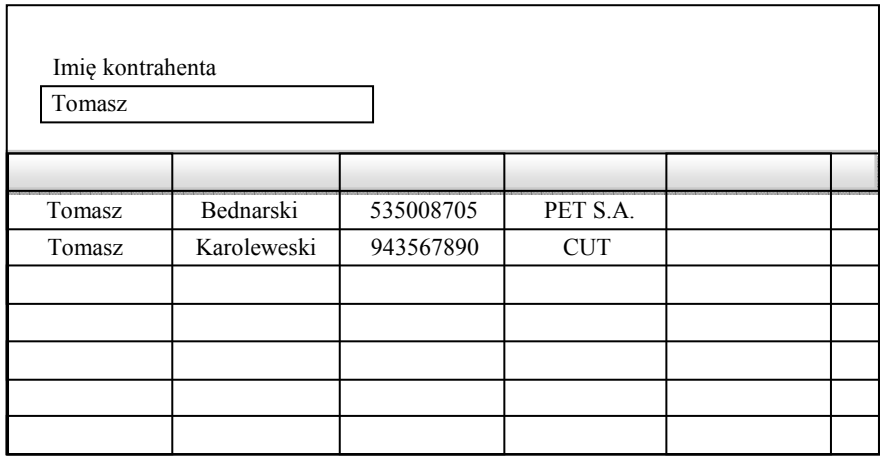
Rozróżniamy trzy główne podatności:

- zwykłe
- ślepe
- poprzez komunikat błędu.

2.1.1. Zwykłe

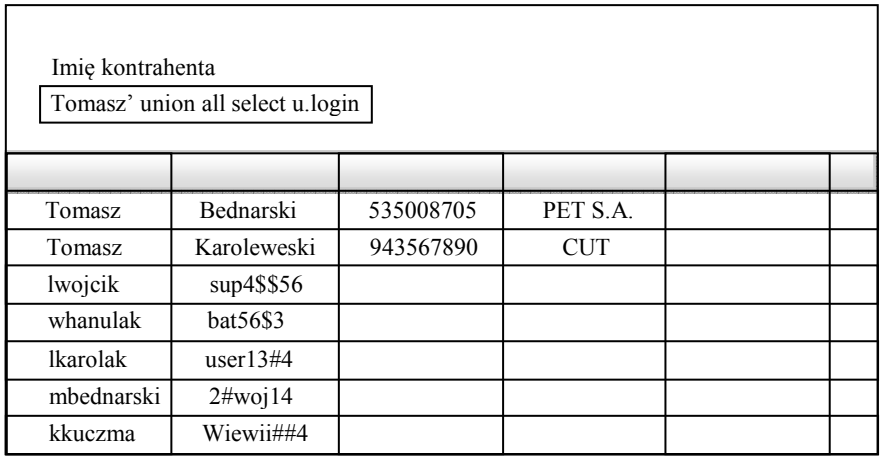
Zwykłe SQL injection to podatność formularzy wyświetlających dane na stronie. Wykorzystując przykładowo okno filtrowania możemy wstrzyknąć poprzez zapytanie dane z innej tabeli np. poprzez zastosowanie złączenia („UNION”). Należy jednak pamiętać, że wstrzyknięta kwerenda powinna posiadać tyle samo kolumn ile wyświetlanych jest w formularzu. Ponadto dane powinny być tego samego typu by silnik bazodanowy poprawnie utworzył tabelę tymczasową z wyłuskanych danych. Najczęściej robi się to drogą dedukcji.

Jeśli aplikacja umożliwia wykonywanie wielokrotnych zapytań albo gdy mamy hasło do bazy danych atakujący może dodać do bazy dowolne dane np. jakiś złośliwy kod, gdy aplikacja jest podatna na XSS (nie sprawdza danych pochodzących z bazy).



Tomasz	Bednarski	535008705	PET S.A.		
Tomasz	Karoleweski	943567890	CUT		

Wpisując np. ‘ union all select u.login, u.password, null, null, null;--



Tomasz	Bednarski	535008705	PET S.A.		
Tomasz	Karoleweski	943567890	CUT		
lwojcik	sup4$$56				
whanulak	bat56$3				
lkarolak	user13#4				
mbednarski	2#woj14				
kkuczma	Wiewii##4				

2.1.2. Ślepe

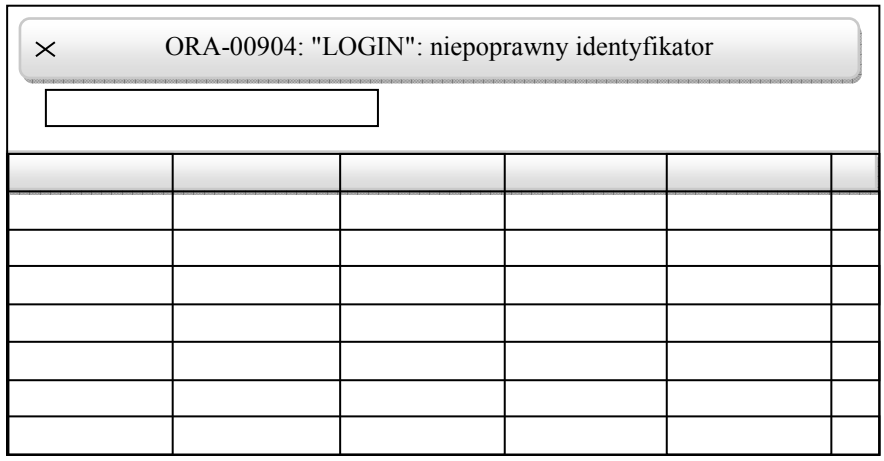
Najczęściej o blind injection mówi się wtedy, kiedy nie zwracane są żadne dane. Pomimo to aplikacja nadal jest narażona na tego typu ataki. Do zapytania można dodać np. próbę zgadnięcia liter w haśle np. przy użyciu metody substring oraz instrukcji case. Jeżeli warunek jest spełniony dodaje się opóźnienie. Natomiast

w przypadku gdy litera nie została odnaleziona można pominąć opóźnienie albo dodać inną jego długość celem upewnienia się, że zapytanie wykonało się poprawnie.

```
UNION SELECT IF(SUBSTRING(user_password,1,1) =
CHAR(50),BENCHMARK(5000000,ENCODE('MSG','by 5 seconds')),null) FROM
users WHERE user_id = 1;
```

2.1.3. Poprzez komunikat błędu

W przypadku gdy aplikacja zwraca błędy bazy danych, można wykorzystać je przy atakach typy SQL Injection. Przykładem tego może być baza danych ORACLE, z której można odczytać nazwę kolumny, co może znacznie przyspieszyć atak.



2.2. Jak się zabezpieczyć

Aby zabezpieczyć się przed tego typu błędami, jeśli dane wejściowe od użytkownika muszą być użyte w zapytaniu SQL, trzeba tak jak w przypadku XSS odpowiednio oznaczyć aby nie były traktowane jako tzn. znaki specjalne. Najlepiej skorzystać z zapytań preparowanych. Polega to na tym, że zanim wykonamy zapytanie, przekazujemy do niego kod SQL z miejscami na wstawienie właściwych parametrów. Następnie przy wywołaniu kwerendy przekazujemy, do tak przygotowanego zapytania dane, które zostaną odpowiednio sformatowane przez bibliotekę bazy danych, w zależności od typu.

```
Select u.* from users u where u.login = :userLogin;
```

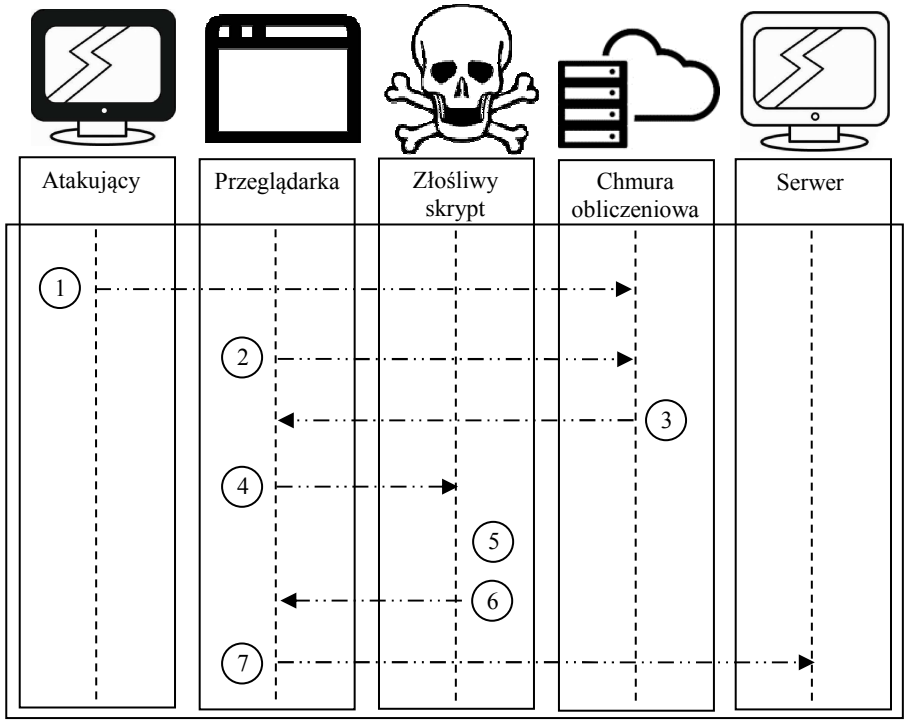
Jeśli biblioteka do bazy danych zwraca wyjątki należy je przechwytywać oraz wyświetlać błąd przyjazny dla użytkownika. Prawdziwy błąd SQL nie powinien być nigdy zwracany do interfejsu użytkownika, zaś pliki logów powinny być zabezpieczone w należyty sposób przed niepowołanym dostępem.

2.3. Session Hijacking

Podatność ta polega na uzyskaniu dostępu do sieci za pomocą luk bezpieczeństwa w systemie. Atakujący próbuje uzyskać dostęp do istniejącej sesji użytkownika, czyli do takiej, której identyfikator został przydzielony już wcześniej. Po uzyskaniu dostępu do sesji użytkownika, atakujący może przejąć część komunikacji klient-serwer symulując jej uczestnika.

Przejęcie sesji ma miejsce, gdy po pomyślnym uwierzytelnieniu logowania, token sesji zostaje wysyłany do przeglądarki klienta z serwera WWW. Atak typu "przejęcie sesji" działa, gdy narusza token, konfiskując go lub zgadując, jaka będzie autentyczna sesja tokenu, uzyskując w ten sposób nieautoryzowany dostęp do serwera sieci Web. Może to doprowadzić do sniffingu sesji, ataków man in the middle czy man in the browser, trojanów, a nawet implementacji złośliwych kodów JavaScript.

Deweloperzy stron internetowych są szczególnie ostrożni podczas przechwytywania sesji, ponieważ pliki cookie HTTP, które służą do podtrzymywania sesji witryny, mogą zostać zaatakowane przez atakującego.



Przebieg ataku:

1. Atakujący hostuje stronę zawierającą złośliwy skrypt
2. Ofiara odwiedza stronę atakującego
3. Przeglądarka pobiera kod HTML wraz ze złośliwym skryptem
4. Następuje uruchomienie skryptu
5. Złośliwy skrypt przechwytuje pliki cookie
6. Nastepuje przekierowanie na serwer atakującego
7. Pliki cookies przekazane zostają do serwera, którego ma nastąpić kompromitacja.

2.3.1. XSS

Cross Site Scripting polega na wstrzykiwaniu złośliwego kodu w oryginalną treść strony, przeważnie poprzez wszelkiego rodzaju formularze internetowe, gdzie zatwierdzona treść jest potem wyświetlana.

Istota podatności XSS to przede wszystkim atak na klienta korzystającego z podatnej web-aplikacji. Przykładem wstrzyknięcia do przeglądarki ofiary fragmentu języka skryptowego, który może być uruchomiony w przeglądarce, jest poniższy kod w języku JS.

```
<script>alert(document.cookie());</script>
```

Skrypt wyciąga plik cookie i wyświetla go w oknie alertu. W efekcie, atakujący ma możliwość wykonania dowolnego kodu skryptowego w przeglądarce.

Warto tu również wspomnieć o tym czym może skutkować wykonanie javascriptowego kodu w przeglądarce ofiary:

- wykradanie cookies sesyjnych czyli de facto przejęcie zalogowanej sesji ofiary,
- dynamiczna podmiana zawartości strony www,
- uruchomienie keyloggera w przeglądarce,
- hostowanie malware-u z wykorzystaniem zaatakowanej aplikacji.

Istnieje wiele narzędzi, które w praktyce pokazują różne możliwe negatywne efekty wykorzystania XSS.

Błędy XSS dzielimy na trzy kategorie:

- Persistent/stored XSS – najbardziej złowroga odmiana, polegająca na umieszczeniu kodu JavaScript po stronie serwerowej.
- Reflected XSS – w tym przypadku kod JavaScript zaszyty jest w linku, który atakujący przesyła do ofiary. Ofiara po kliknięciu na linka łączy się z aplikacją, przekazując jej nieświadomie fragment HTML zawierający kod wykonujący JavaScript. Aplikacja zwraca ofierze HTML zawierający wcześniej podany JavaScript, powodując wykonanie kodu w przeglądarce.
- DOM Based XSS – jest atakiem, w którym atak jest wykonywany poprzez modyfikację drzewa DOM w przeglądarce ofiary, w taki sposób, aby kod po stronie klienta działał w sposób "nieoczekiwany". Oznacza to, że sama odpowiedź HTTP się nie zmienia, ale kod strony klienta jest wykonywany inaczej, ze względu na złośliwe modyfikacje, które wystąpiły w środowisku DOM.

XSS jest jedną z najbardziej rozpowszechnionych luk w aplikacjach internetowych. Występuje najczęściej w aplikacjach internetowych, które korzystają z niezatwierdzonych lub niekodowanych danych wejściowych użytkownika w generowanym przez siebie wyjściu.

Wykorzystując XSS, atakujący nie atakuje bezpośrednio ofiary. Zamiast tego osoba atakująca wykorzysta lukę w witrynie lub aplikacji internetowej odwiedzanej przez ofiarę, wykorzystując w naturalny sposób podatną witrynę internetową do dostarczania szkodliwego skryptu do przeglądarki ofiary.

Przykład działania może zostać opisany na popularnym języku JavaScript. Złośliwy JavaScript ma dostęp do wszystkich tych samych obiektów, co reszta strony, w tym do plików cookie. Pliki cookie są często używane do przechowywania

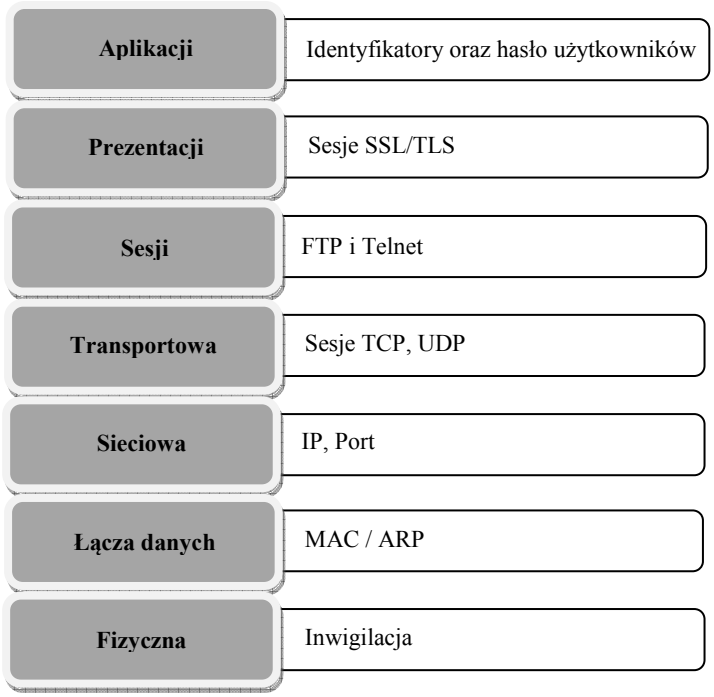
tokenów sesji, jeśli atakujący może uzyskać plik cookie sesji użytkownika, może podszyć się pod tego użytkownika. JavaScript ma dostęp do wielu funkcji przeglądarek takich jak czytanie i dowolna modyfikacja DOM, wykorzystywanie nagłówka XMLHttpRequest do wysyłania żądań HTTP z dowolną treścią do dowolnych miejsc docelowych, wykorzystanie interfejsów API HTML5, umożliwiających dostęp do geolokacji, kamery internetowej, mikrofonu, a nawet określonych plików z systemu plików użytkownika. Podczas gdy większość tych interfejsów API wymaga zgody użytkownika, XSS w połączeniu z pewną sprytną inżynierią społeczną może skutecznie umożliwić atakującemu ich wykorzystanie.
Powyższe, w połączeniu z inżynierią społeczną, pozwala atakującym na podejmowanie zaawansowanych ataków, takich jak kradzież plików cookie, keylogging, phishing (kradzież tożsamości).

2.3.2. Network Sniffing

Network Sniffing koncentruje się na znacznie szerszym podejściu do hakowania, czyli podsłuchiwaniem sieci. Technika ta jest często wykorzystywana przez hakerów. Istnieje wiele darmowych snifferów, które między innymi poprzez nasłuchiwanie pakietów w sieci potrafią w dość szybki sposób złamać klucz WPA. Atak ten obejmuje przechwytywanie, dekodowanie, sprawdzanie oraz interpretowanie informacji zawartych w pakietach sieciowych. Celem ataku jest zwykle kradzież identyfikatorów użytkowników, haseł czy nawet numerów kart kredytowych. Sniffing ze względu na przebieg ataku określa się go jako typ pasywny gdzie atakujący jest niewidzialny w sieci. Utrudnia to znacznie wykrycie atakującego w sieci, dlatego jest to jeden z najniebezpieczniejszych rodzajów ataków.

Narzędzia do nasłuchiwania sieci zostały stworzone z bardziej etycznych pobudek i były wykorzystywane wyłącznie przez profesjonalnych inżynierów sieciowych. Etyczne ich wykorzystanie m.in. przez ludzi potocznie zwanych white hat, którzy mogą dostarczyć organizacji informacji z analizy przechwyconych pakietów, analizy ruchu sieciowego, ewentualnych problemów sieciowych, jednakże cyberprzestępcy wykorzystują je w celach nieetycznych takich jak, wykradanie identyfikatorów i haseł użytkowników sieci, kradzież danych wiadomości błyskawicznych czy e-mail, celem kompromitacji organizacji.

Aby zrozumieć, dlaczego hackerzy „węszą”, musimy wiedzieć, co mogą uzyskać z sieci. Poniższy rysunek pokazuje warstwy OSI oraz informacje, które haker może wyłuskać w każdej warstwie.



Jak można zauważyć haker może zaatakować aż na 7 warstwach modelu OSI, dlatego jasne i szczelne procedury bezpieczeństwa organizacji są szczególnie ważne by skutecznie się przed nim chronić.

2.3.3. Social Engineering

Inżynieria społeczna jest w istocie sztuką uzyskiwania dostępu do budynków, systemów lub danych poprzez wykorzystanie ludzkiej psychiki, a nie poprzez włamywanie się lub używanie technik hakerskich. Na przykład, zamiast próbować znaleźć lukę w oprogramowaniu, inżynier socjalny może zadzwonić do pracownika i przedstawić się jako osoba wspierająca IT, próbując oszukać pracownika w celu ujawnienia jego hasła.

Przeprowadzenie skutecznej inżynierii społecznej zależy od tego, czy ludzie są świadomi swoich cennych informacji i czy zwracają szczególną uwagę na ich ochronę.

Ludzka natura, czyli ufność jest to podstawa każdego ataku inżynierii społecznej.

Ignorancja na temat inżynierii społecznej i jej wpływ na siłę roboczą sprawia, że organizacja jest łatwym celem. Inżynierowie społeczni by ujawnić informacje opierają się na ludzkiej chciwości np. obiecując coś za nic, bądź poczuciu

moralności np. prosząc o pomoc, a atakowani zgodnie z poczuciem moralnego obowiązku ujawniają informacje w ramach wyższego celu.

2.4. Spoofing

Jest to atak, którego ideą jest przyjmowanie fałszywej tożsamości celem uzyskania nieuprawnionego dostępu do systemu i usług. Ponieważ podszywanie dotyczy wszystkich warstw systemu OSI, bardzo trudno się przed nim bronić. Wyróżnia się dwie główne odmiany spoofingu:

- Blind spoofing – polega na przesyłaniu danych uwierzytelniających, przy braku dostępu do informacji, aby określić parametry sieci, śledzić zmiany ustawień lub, w szczególnym przypadku, uzyskać połączenie.
- Active spoofing – monitorowanie, generowanie, uszkadzanie bądź usuwanie pakietów wysyłanych w trakcie komunikacji, między serwerem a uwierzytelnionym użytkownikiem, w celu uzyskania uprawnień.

Jedyną obroną przed spoofingiem są ścisłe reguły filtrowania pakietów, stosowanie zabezpieczeń kryptograficznych np. SSH, czy PGP oraz walka z lekkomyślnością użytkownika, który przez nieuwagę, czy też niewiedzę zezwala na połączenie. Często użytkownicy sami pobierają programy umożliwiające wykradzenie klucza, zwłaszcza gdy łatwowiernie korzysta się z automatycznych aktualizacji programów, ponieważ nawet zaufane witryny często padają ofiarami podszywania. Pobieranie plików i programów z nieznanych źródeł bez odpowiednich poświadczeń to następna otwarta furtka dla cyberprzestępców.

Większość programów malware, działa w tle na poziomie ukrytym, wysyłając informacje przez Internet, lub nasłuchując na określonych portach poleceń od

hackera. Instalowanie poprawek, nie tylko systemowych, sprzyja bezpieczeństwu i wydajności systemu. Programy takie jak konie trojańskie rezydują w systemie, uruchamiając się dzięki wpisom do kluczy rejestru. Często też tworzą zapisy w plikach uruchomieniowych, czyli zarządzających każdym startem systemu. Należy regularnie przeglądać listy działających programów, pliki rozruchowe, ustawienia kluczy autostartu oraz dzienniki zdarzeń. Badanie wydajności systemu, pozwala na wykrycie nieprawidłowości w wykorzystaniu zasobów komputera i odnalezienie programów, użytkowników i usług mogących negatywnie wpływać na system. Uwagę powinny zwracać zwłaszcza nietypowe parametry ruchu sieciowego, takie jak błędy w transmisji, czy liczba odrzuconych pakietów. Dzięki analizie wykorzystania połączenia sieciowego komputera, można odnaleźć szkodliwe programy łączące się z siecią, komunikujące się z nieznanymi lokacjami, lub powtarzające się w określonym czasie, żądania nawiązania połączenia z różnymi usługami w sieci.

2.5. Metody zapobiegawcze

By skutecznie chronić się przed wyżej wymienionymi atakami należy przede wszystkim używać generatora liczb losowych, które generują silnie kryptograficzne liczby losowe, zapewniające odpowiednio mocne klucze sesyjne. Nie należy używać skompromitowanych algorytmów szyfrujących. Dobrą praktyką jest wyłączanie wszystkich niepotrzebnych usług systemowych. W swoich aplikacjach blokuj wczytywania stron w ramkach (iFrame). Użytkownicy powinni dysponować wydajnym programem antywirusowym, zabezpieczającym przed złośliwym oprogramowaniem oraz powinni je jak najczęściej aktualizować.

Aby zapobiec atakom typu XSS, należy filtrować dane od użytkownika oraz bazy danych lub je odpowiednio formatować, jeśli są wstawiane w kontent naszej strony.

Jeśli przechowujemy wrażliwe dane w aplikacji, powinniśmy udostępniać naszą aplikację poprzez protokół https. W przeciwnym wypadku będą możliwe ataki typu Man in the middle na naszą aplikację.

Identyfikator sesji możemy przekazywać zarówno w adresie URL, jak i w ciasteczku. Pierwszy z wymienionych sposobów zdecydowanie nie jest dobrym pomysłem, ponieważ:

- Każda osoba używająca tego samego komputera będzie mogła poznać identyfikator sesji z historii stron przeglądarki.
- Adresy URL często zapisywane są przez serwery proxy, dzienniki zdarzeń, itp. W wyniku ich analizy można przeczytać identyfikatory sesji.
- Kiedy użytkownik przechodzi na inną stronę, adres URL wraz z identyfikatorem sesji dostępny jest w nagłówku HTTP Referer.

- Użytkownik, przekazując adres URL innym osobom, nieumyślnie udostępnia także identyfikator sesji.
- Naturalnym wyborem do przekazywania identyfikatora sesji są więc ciasteczka.
- W konfiguracji powinniśmy też ustawić odpowiednie parametry dotyczące samego ciasteczka sesji (lifetime, path, secure, httponly).
- Identyfikator sesji powinien być odpowiednio długi i losowy, trudny do odgadnięcia oraz trudny do odtworzenia.

Sesje, które nie wygasają w odpowiednio krótkim odstępie czasu, dają atakującemu o wiele więcej możliwości na atak. Mechanizm obsługi sesji musi w pełni kontrolować czas, po jakim sesja traci ważność albo jest usuwana z magazynu przechowującego. Sesja powinna być przerwana, gdy nie stwierdzono żadnej aktywności przez pewien okres czasu, np. 30 minut, gdy nastąpi błąd bezpieczeństwa, lub gdy użytkownik sam przerwie sesję poprzez wylogowanie z systemu.

3. Podsumowanie

Pomimo tego. że wiele organizacji boryka się z cyberprzestępczością, wprowadzanie oraz przestrzeganie odpowiednich procedur bezpieczeństwa jest rzadkością.

Firmy powinny korzystać zawsze z najnowszego oprogramowania, a zespół IT powinien pilnować by były zawsze aktualne. Organizacje powinny korzystać z technologii pozwalających na wykrywanie szkodliwego oprogramowania, a również uniemożliwić pracownikom instalację oprogramowania z niezaufanych źródeł. Kopie bezpieczeństwa powinny być tworzone stosunkowo często, ponieważ w przypadku utraty danych bądź dostępu skutkiem ataku będzie można w stosunkowo prosty i bezbolesny sposób przywrócić utracony dostęp.

Często niedoceniane są wszelkie audyty bezpieczeństwa, które dają jasną ocenę aktualnego bezpieczeństwa sieci oraz pozwalają na identyfikację błędów w zabezpieczeniach.

Bibliografia

1. Francois Mouton, Alastair Nottingham, Louise Leenen, H. S. Venter, „FINITE STATE MACHINE FOR THE SOCIAL ENGINEERING ATTACK DETECTION MODEL: SEADM", SAIEE AFRICA RESEARCH JOURNAL, Volume 109, Issue 2, Pages 133-147, JUN 2018
2. Zhenjun Zhang, Xingqun Zhan, „Statistical Analysis of Spoofing Detection based on TDOA", IEEJ TRANSACTIONS ON ELECTRICAL AND ELECTRONIC ENGINEERING,nVolume 13, Issue 6, Pages 840-850, JUN 2018
3. Hossen Mustafa, Wenyuan Xu, Ahmad-Reza Sadeghi, Steffen Schulz, „End-to-End Detection of Caller ID Spoofing Attacks", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, Volume 15, Issue 3, Pages 423-436, MAY-JUN 2018
4. Karis D'silva, J. Vanajakshi, K. N. Manjunath, Srikanth Prabhu, „An Effective Method for Preventing SQL Injection Attack and Session Hijacking", 2017 2ND IEEE INTERNATIONAL CONFERENCE ON RECENT TRENDS IN ELECTRONICS, INFORMATION & COMMUNICATION TECHNOLOGY (RTEICT), Pages 697-701, 2017
5. Bharti Nagpal, Naresh Chauhan, Nanhay Singh, „A Survey on the Detection of SQL Injection Attacks and Their Countermeasures", JOURNAL OF INFORMATION PROCESSING SYSTEMS, Volume 13, Issue 4, Pages 689-702, AUG 2017
6. Shashank Gupta, B. B. Gupta, Pooja Chaudhary, „Hunting for DOM-Based XSS vulnerabilities in mobile cloud-based online social network", FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE, Volume 79, Pages 319-336, FEB 2018
7. Materiały szkoleniowe EC-Council do egzaminu CEH v9 (Certified Ethical Hacker)

Streszczenie

Każda organizacja niezależnie od jej wielkości musi posiadać system bezpieczeństwa. Zrozumienie wartości danych wrażliwych oraz ochronę tychże informacji ułatwiają nam dziś między innymi regulacje branżowe i prawne. W praktyce stosuje się dedykowane narzędzia oraz rozwiązania, które najczęściej są zaszyte w komercyjnych usługach IT. W artykule autorzy przedstawili krótki przegląd ataków, których celem jest kompromitacja danych poufnych bądź zaprzestanie świadczenia usług, oraz metody zapobiegania im.

Abstract

Every organization, regardless of its size, must have a security system. Understanding the value of sensitive data and the protection of this information are facilitated today by, inter alia, industry and legal regulations. In practice, dedicated tools and solutions are most often stitched in commercial IT services. In the article, the authors presented a brief overview of attacks aimed at compromising confidential data or ceasing to provide services, as well as methods of preventing them.

Keywords: internet services, network attacks, compromise of confidential data