Testy Penetracyjne

Atakowanie aplikacji

Arkadiusz Chrobot

Katedra Systemów Informatycznych

20 maja 2024

Plan

Wstęp

Systemy operacyjne i firmware w komputerach i urządzeniach sieciowych są zazwyczaj regularnie zabezpieczane i aktualizowane przez odpowiedzialnych za nie administratorów. Bardziej podatnym celem ataków mogą okazać się aplikacje, ze względu na swoją różnorodność i liczebność. Dotyczy to także oprogramowania użytkowego wytworzonego i użytkowanego wewnętrznie przez firmy i organizacje.

Podatności aplikacji

Wstrzyknięcie kodu

Istnieje kilka odmian i pododmian ataków wykorzystujących luki zabezpieczeń pozwalających na wstrzyknięcie kodu. Najczęściej spotykaną są ataki wstrzyknięcia kodu SQL (ang. SQL *Injection* — SQLi). Wynik takiego ataku może być widoczny bezpośrednio na ekranie lub doprowadzić do zmiany stanu aplikacji, której atakujący nie będzie mógł zaobserwować. W tym ostatnim przypadku może on zastosować wariant tej metody nazywany *ślepym atakiem typu* SQL*i*. Najczęściej ten rodzaj ataku występuje w dwóch odmianach. Pierwsza wykorzystuje wyrażenia logiczne do warunkowej zmiany widoku aplikacji (ang. *boolean blind* SQL*i*), a druga do warunkowego opóźnienia działania oprogramowania (ang. *time-based* SQL*i*).

Podatności aplikacji

Wstrzyknięcie kodu

Niektóre aplikacje korzystają z podprogramów, które umożliwiają im wykonywanie poleceń ▸ powłoki systemowej (np.: funkcje `system()` i `execl()` w języku C lub metoda `Runtime.exec()` w języku Java). Jeśli dane wejściowe dla tych podprogramów lub środowisko wykonania aplikacji mogą być kontrolowane przez intruza, to może on przeprowadzić atak *wstrzyknięcia* ▸ *polecenia* (ang. *command injection*). Ta technika polega na bezpośredniej zmianie argumentów podprogramu realizującego polecenie powłoki tak, aby oprócz określonego przez programistę polecenia wykonał jeszcze podane przez atakującego. Można to osiągnąć łącząc je np. przy pomocy operatora `&&` (logiczne *i* w języku powłoki). Jeżeli programista użył ścieżki względnej do polecenia i nie zabezpieczył zmiennych środowiskowych, które może kontrolować intruz, to atak może być wykonany np. przez modyfikację wartości zmiennej `PATH`. Istnieją też *ślepe* warianty tej metody.

Podatności aplikacji

Wstrzyknięcie kodu

Wstrzyknięcie może też zapytań LDAP, czyli protokołu używanego przez niektóre usługi odpowiedzialne za autoryzację. Korzysta z nich np. Active Directory. ▸ Wstrzyknięcia LDAP umożliwiają intruzowi obejście mechanizmów autoryzacji lub uzyskanie informacji o uprawnieniach użytkowników i grup.

W przypadku aplikacji internetowych często spotykanymi podatnościami są te umożliwiające wstrzyknięcie kodu HTML lub JavaScript. Określa się je mianem *Cross-Site Scripting*, w skrócie XSS. Znane są trzy rodzaje ataków wykorzystujących te luki: *odbity* XSS (ang. *reflected* XSS), *trwały* XSS (ang. *persistent or stored* XSS) oraz bazujący na modelu DOM (ang. DOM-*Based* XSS). Bliżej z tymi rodzajami ataków pozwala zapoznać się ▸ Google XSS Game.

Podatności aplikacji

Wstrzyknięcie kodu

Pierwotną przyczyną powstawania luk umożliwiających wstrzyknięcie kodu jest brak walidacji danych wejściowych, czyli ograniczenia ich do takiego podzbioru, który uniemożliwi przeprowadzenie ataków z ich wykorzystaniem. Najskuteczniejsze najczęściej okazuje się podejście, które akceptuje tylko dopuszczalne wartości wejściowe (ang. *white list approach*), ale ich zbiór może w niektórych przypadkach być trudny do określenia. W takiej sytuacji warto rozważyć zastosowanie podejścia polegającego na odrzucaniu niepożądanych wartości (ang. *black list approach*). Należy także zauważyć, że luki te niekoniecznie muszą dotyczyć pól edycyjnych w interfejsach użytkownika, ale np. w przypadku aplikacji internetowych mogą występować w parametrach przekazywanych w adresach URL. Ta ich odmiana jest często określana jako *zanieczyszczenie parametrów* (ang. *parameter pollution*).

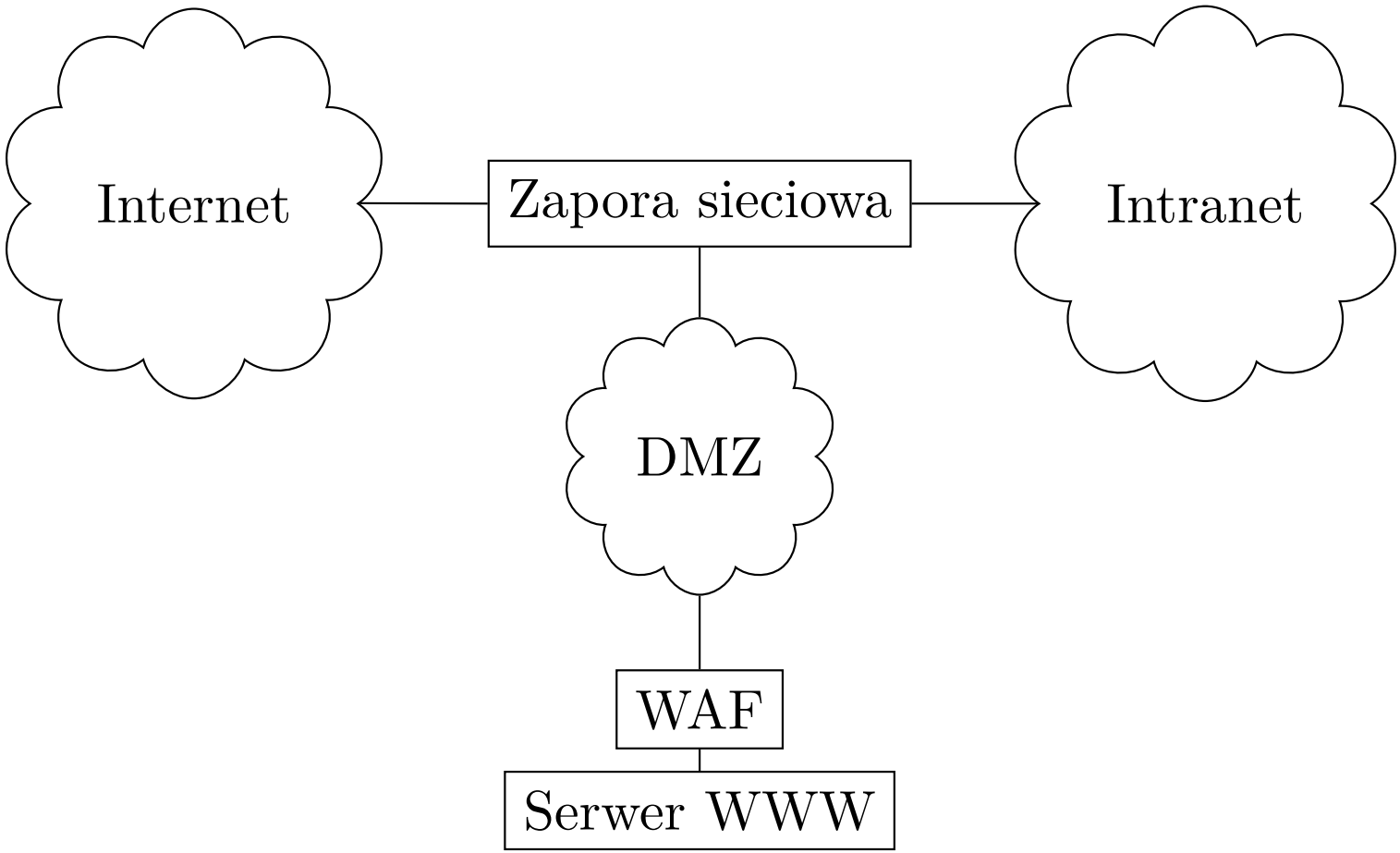
Podatności aplikacji

Wstrzyknięcie kodu

W przypadku złożonych aplikacji wyeliminowanie wszystkich wystąpień podatności umożliwiających wstrzyknięcie kodu może okazać się trudne. W takim przypadku warto rozważyć użycie dodatkowego środka zapobiegawczego jakim są zapory typu WAF (ang. *Web Application Firewall*). W przeciwieństwie do typowych zapór sieciowych (ang. *network firewalls*) sprawdzają one jedynie komunikaty protokołu HTTP w poszukiwaniu potencjalnie niebezpiecznych danych. Jeśli je znajdą, to zablokują je zanim trafią do aplikacji. Rysunek 1 przedstawia schematycznie umiejscowienie zapory WAF w systemie. Powinna ona być zainstalowana przez serwerem WWW.

Podatności aplikacji

WAF



Rysunek: Lokalizacja zapory typu WAF

Podatności aplikacji

Atakowanie uwierzytelniania

Najpopularniejszą metodą uwierzytelniania jest stosowanie haseł. Niestety, jest ona także najprostszą do pokonania, gdyż bazuje na wiedzy, która powinna być dla intruza niedostępna. Jeśli on jednak ją posiądzie, to może podawać się za uprawnionego użytkownika. Metod pozyskania haseł jest kilka:

1. użycie socjotechniki;
2. podsłuchiwanie niezaszyfrowanego ruchu sieciowego;
3. użycie hasła użytkownika, które wyciekło w wyniku innego ataku;
4. zastosowanie metod siłowych lub słownikowych.

Warto zauważyć, że sprzęt sieciowy dosyć często ma domyślne dane uwierzytelniające, które nie są zmieniane przez jego użytkowników, a które są ▸ publicznie dostępne lub typowe (np. *admin/admin*). Niekiedy te dane mogą być wprost zaszyte w firmware tych urządzeń.

Podatności aplikacji

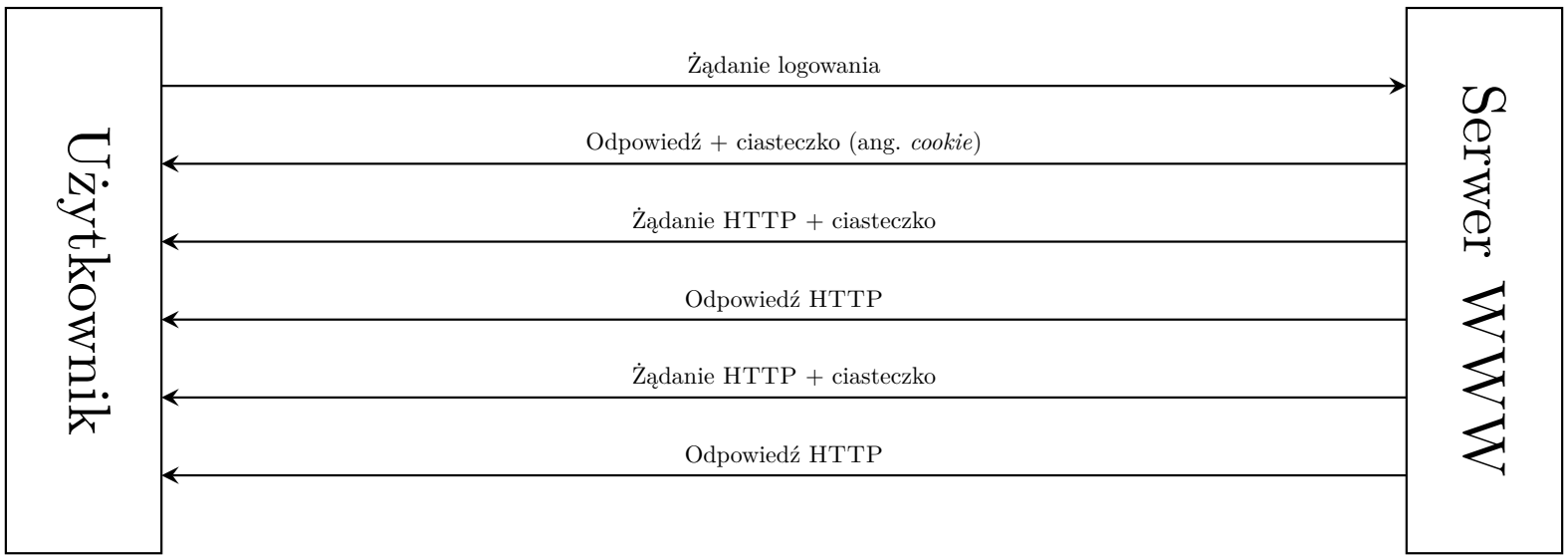
Atakowanie sesji

Przejęcie sesji (ang. *session hijacking*) polega na przechwyceniu poświadczeń użytkownika, który przeszedł proces uwierzytelnienia. W przypadku aplikacji internetowych takie poświadczenie ma formę *ciasteczka* (ang. *cookie*), czyli informacji dostarczanej przez serwer przeglądarce, która następnie ją przechowuje i wysyła wraz z każdym zapytaniem do tego serwera (Rysunek 2). *Kradzież* takiego ciasteczka pozwala atakującemu podszywać się pod prawowitego użytkownika i uzyskać przejąć jego sesję z serwerem (Rysunek 3). Jest to zatem forma ataku powtórzeniowego. Istnieje kilka sposobów pozyskania ciasteczka:

- jeśli komunikacja nie jest szyfrowana, to intruz może podsłuchiwać ruch sieciowy i wykraść ciasteczko, gdy jest transmitowane;
- atakujący może zainstalować złośliwe oprogramowanie, w formie np. dodatku w przeglądarce użytkownika;
- atak typu *man-in-middle*.

Podatności aplikacji

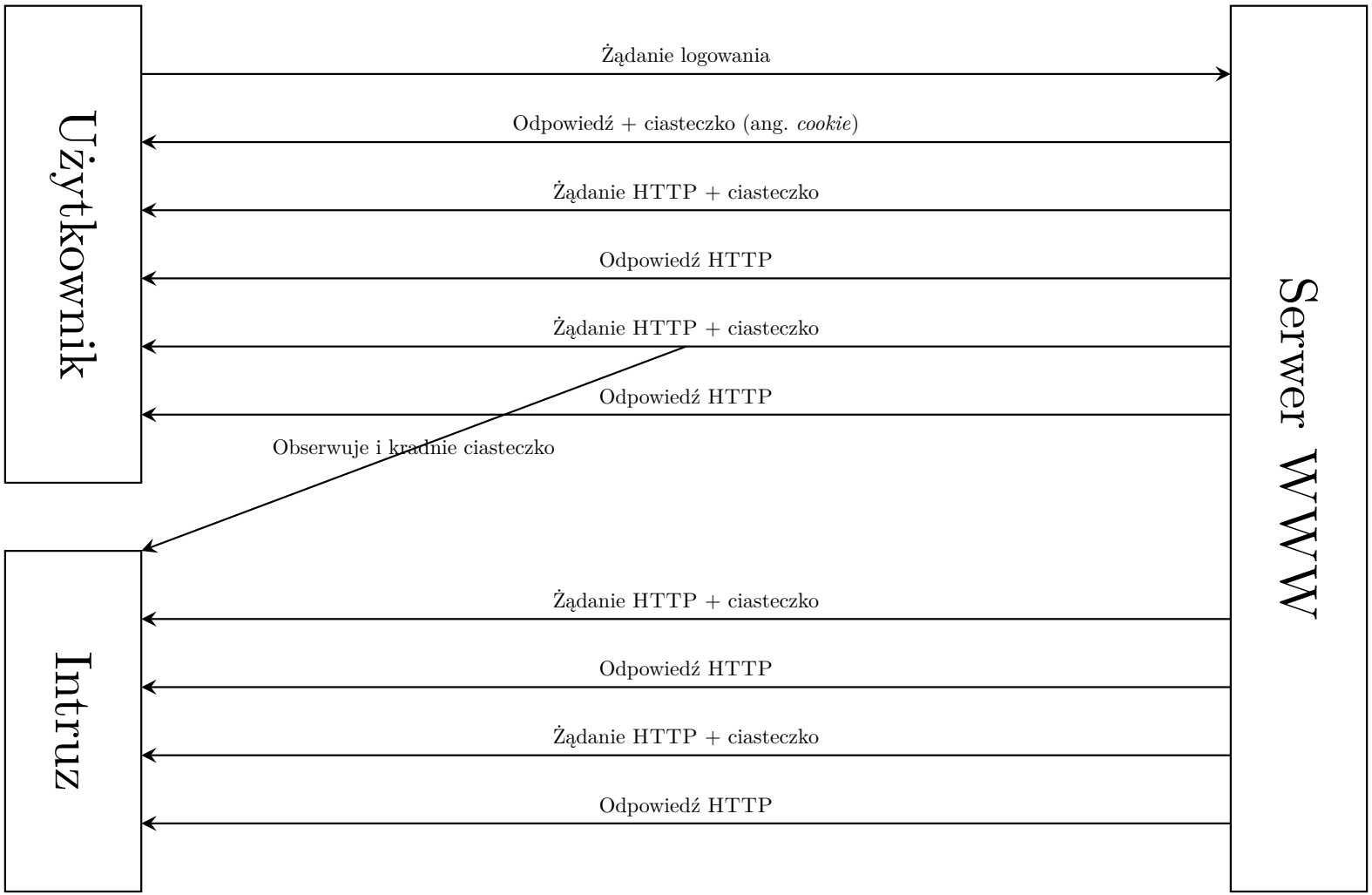
Uwierzytelnianie sesji



Rysunek: Uwierzytelnianie sesji za pomocą ciasteczek

Podatności aplikacji

Przejmowanie sesji



Rysunek: Przejmowanie sesji za pomocą ciasteczek

Podatności aplikacji

Atakowanie sesji

Niektóre aplikacje internetowe przypisują użytkownikom na stałe ten sam identyfikator sesji (ang. *session* ID), zamiast nadawać nowy przy tworzeniu nowej sesji. Umożliwia to intruzowi przeprowadzenie ataku *utrwalonej sesji*. Ta metoda wymaga trzech kroków:

1. Uzyskanie identyfikatora sesji, który może być zapisany w ciasteczku, argumencie URL lub można go znaleźć w logu.
2. Zmuszenie prawowitego użytkownika do uwierzytelnienia się w aplikacji, np. za pomocą wyskakującego okna (ang. *popping window*) lub linku w wiadomości e-mail. Intruz nie musi śledzić procesu uwierzytelnienia.
3. Użycie identyfikatora sesji, by uzyskać dostęp do zdalnego serwera.

Przeciwdziałać kradzieży ciasteczek można ustawiając ich atrybuty `Secure` i `HttpOnly`. Dodatkowo można sprawdzić poprawność konfiguracji TLS np. w serwisach ▸ SSL Labs i ▸ SSL Tools.

Podatności aplikacji

Niezweryfikowane przekierowania

Niezweryfikowane przekierowania (ang. *unvalidated redirects*) lub *niebezpieczne przekierowania* URL (ang. *insecure* URL *redirect*) są innym typem podatności w aplikacjach internetowych, które może wykorzystać intruz. Niektóre serwery pozwalają przeglądarką przesłać adres docelowy do aplikacji, a potem przekierować użytkownika pod ten adres, w ramach zakończenia transakcji. Jest to wygodne rozwiązanie, bo pozwala zmienić stronę docelową bez konieczności modyfikacji kodu aplikacji. Jednakże, może ono stanowić lukę bezpieczeństwa, bo intruz może np. opublikować link do aplikacji z przekierowaniem prowadzącym do strony, którą on kontroluje. Można temu zapobiec sprawdzając, czy przekierowanie prowadzi do jednej ze stron, które znajdują się na liście dozwolonych lub np. ograniczając przekierowania tylko do zaufanych domen. Tego typu metoda nazywa się *weryfikowanymi przekierowaniami* (ang. *validated redirects*).

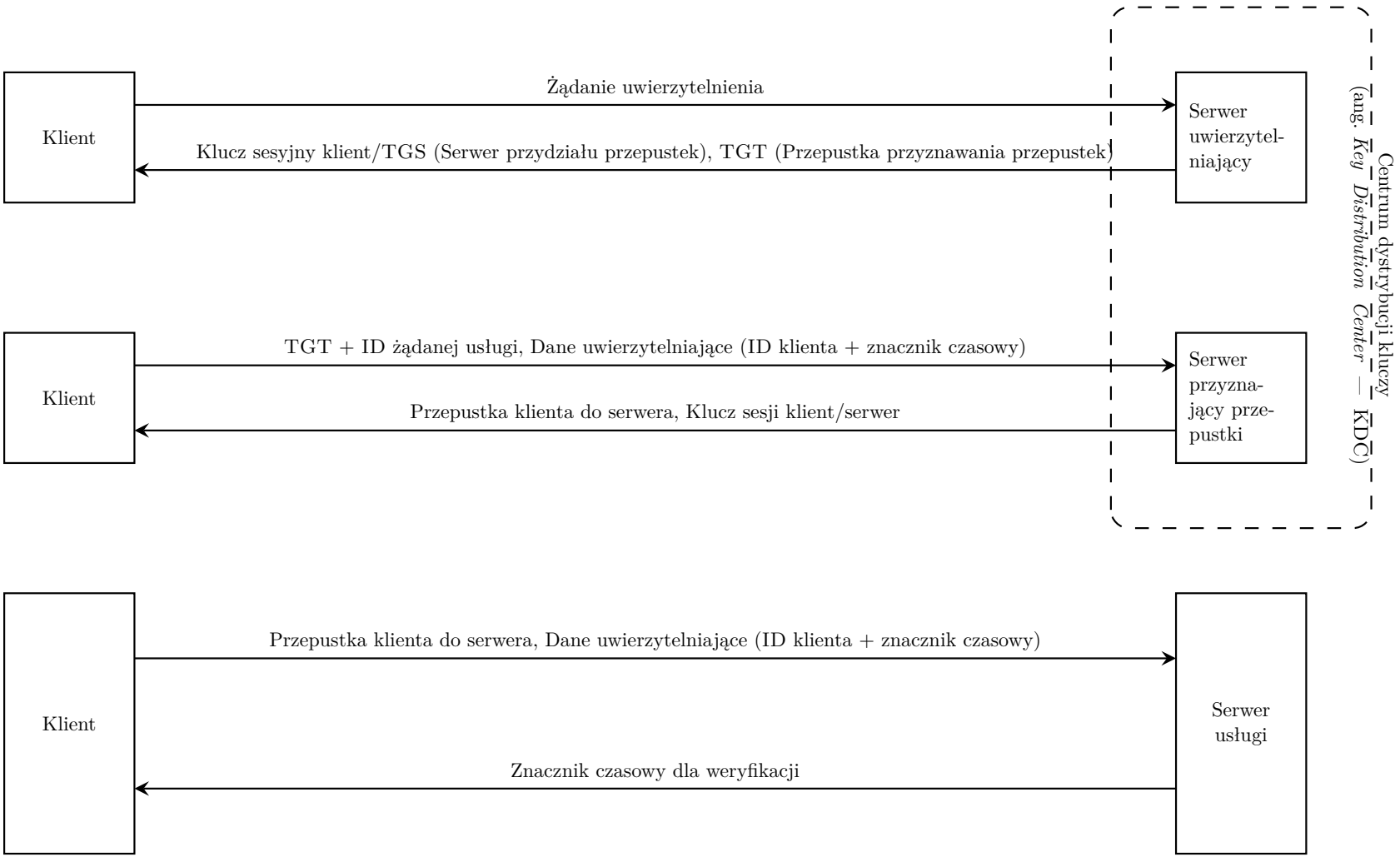
Podatności aplikacji

Ataki na usługę Kerberos

Kerberos jest popularnym i sprawdzonym protokołem uwierzytelniania opracowanym w MIT. Korzysta z niego np. Active Directory. Proces uwierzytelnienia z użyciem tego protokołu przedstawia schematycznie Rysunek 4. Główną rolę odgrywa w nim *Centrum dystrybucji kluczy* (KDC), składające się z *serwera uwierzytelniającego* i *serwera przyznającego przepustki* (ang. *Ticket Granting Servers* — TGS). Pierwszy uwierzytelnia klienta i przydziela mu *przepustkę przydzielającą przepustki* (ang. *Ticket Granting Ticket* —TGT). Drugi na podstawie tej przepustki, identyfikatora usługi żądanej przez klienta i danych uwierzytelniających wydaje klientowi przepustkę, która jest jego poświadczeniem dla serwera dostarczającego żądaną usługę.

Podatności aplikacji

Ataki na usługę Kerberos



Rysunek: Uwierzytelnianie za pomocą Kerberos

Podatności aplikacji

Ataki na usługę Kerberos

Metody ataku na Kerberos obejmują między innymi:

- Przejęcie konta administracyjnego KDC.
- Ponownie użycie przepustki Kerberos (ang. *pss-the-ticket*), co pozwala intruzowi podszywać się pod klienta, przez okres jej ważności.
- Ponownie użycie związanego z przepustką tajnego klucza (ang. *pass--the-key*), celem uzyskania nowej przepustki.
- Ataki skierowane na TGT — przepustki przyznające przepustki (TGT), są szczególnie cenne dla atakujących, bo mają one wydłużony okres obowiązywania i pozwalają na pełny dostęp do systemów obsługiwanych przez Kerberso, w tym umożliwiają pozyskiwanie nowych przepustek, modyfikację kont, a nawet ich usuwanie.

Narzędziem umożliwiającym atakowanie Kerberos jest Mimikatz, dostępny także z poziomu Metasploit.

Podatności aplikacji

Atakowanie autoryzacji

Niektóre aplikacje internetowe są tak zaprojektowane, aby bezpośrednio pozyskiwać informacje z bazy danych, na podstawie argumentu przekazanego przez użytkownika w URL lub za pomocą metody POST. Jest to dobre rozwiązanie, o ile aplikacja stosuje dodatkowe mechanizmy uwierzytelniania i autoryzacji. W przeciwnym przypadku użytkownik po zalogowaniu może uzyskać dostęp do danych, które nie są przeznaczone dla niego, zmieniając tylko wartość opisanego argumentu. Nazywa się ta podatność *niezabezpieczonymi, bezpośrednimi odwołaniami do obiektu* (ang. *Insecure Direct Object References*). Dodatkowym mechanizmem zabezpieczającym powinna być losowość wartości odwołania (argumentu) dla każdego obiektu, aby nie można łatwo jej ustalić.

Podatności aplikacji

Atakowanie autoryzacji

Podatność *przeglądania katalogów* (ang. *directory traversal*) pozwala intruzowi na uzyskanie dostępu do innych katalogów i plików na serwerze, niż te, z których powinna korzystać aplikacja. Może on uzyskać taki dostęp dodając do URL operator `..`, który pozwala przejść do katalogu nadrzędnego. Pewną odmianą tego ataku jest sprawdzanie URLi, kończących się takimi nazwami jak `admin`, `business`, itp. celem poszukiwania ukrytych aplikacji. Ten atak można przeprowadzić za pomocą narzędzia OWASP *DirBuster*. Jest to stosunkowo stare oprogramowanie, ale nadal może się okazać przydatne.

Podatności aplikacji

Atakowanie autoryzacji

Atak typu *zawarcie pliku* (ang. *file inclusion*) pozwala na zdalne wykonanie (na serwerze) pliku, którego nazwa jest zawarta w URL przesyłanym do aplikacji. Są dwie odmiany tego ataku:

zawarcie pliku lokalnego (ang. *local file inclusion*) Wymaga od intruza wcześniejszego umieszczenia wykonywalnego pliku na serwerze, a potem podania jego lokalizacji w URL.

zawarcie pliku zdalnego (ang. *remote file inclusion*) W tym przypadku jedynym warunkiem jest, aby plik wykonywalny był dostępny w Internecie. Jeśli jest on spełniony, to intruz musi jedynie przekazać jego adres w URL przesyłanym do aplikacji.

Plik wykonywalny używany zazwyczaj w takich atakach zawiera implementację powłoki webowej (ang. *web shell*), co pozwala intruzowi na dosyć swobodny dostęp do serwera.

Podatności aplikacji

Atakowanie autoryzacji

Aplikacje internetowe zazwyczaj mają własnych użytkowników i własne poziomy uprzywilejowania, zatem intruzi mogą próbować przeprowadzać ataki mające na celu *podniesienie uprawnień* (ang. *privilege escalation*).

Podatności aplikacji

Fałszowanie żądań

Ataki związane z *fałszowaniem żądań* (ang. *request forgery*) wykorzystują fakt, że serwer najczęściej ma zaufanie do żądań wysyłanych przez klienta. Jest to swego rodzaju odwrotna sytuacja niż w przypadku ataków typu *Cross-Site Scripting*. Do tych ataków zaliczane są:

Cross-Site Request Forgery —XSRF lub CSRF Wymagaja, aby użytkownik miał otwarte w przeglądarce dwie różne strony. Jeśli druga z nich zawiera złośliwi kod umieszczony w niej przez intruza, to może on przesłać żądanie do pierwszej aplikacji, które ona następnie przekieruje do serwera. ▸ Obronę przed takimi atakami stanowią tajne tokeny anty-CSFR, nagłówki `Referrer` oraz atrybuty `SameSite` ciasteczek.

Server-Side Request Forgery (SSRF) ten ▸ atak jest podobny do CSRF, ale możliwy jest w aplikacjach, w których serwer pobiera informacje ze strony, której URL jest dostarczany przez użytkownika.

Podatności aplikacji

Przechwycenie kliknięcia

Atak typu *przechwycenie kliknięcia* (ang.*clickjacking*) dotyczy zarówno aplikacji internetowych, jak i mobilnych. Polega na umieszczeniu w interfejsie strony lub aplikacji elementu, po którego kliknięciu zostaną w aplikacji wykonane operacje niepożądane przez użytkownika, np. osłabienie jej zabezpieczeń. Taki element, może np. być przezroczysty i nałożony na inny, będący prawidłową częścią stony.

Niebezpieczne praktyki programistyczne

Niebezpieczne praktyki programistyczne (ang. *unsecure coding practices*) niekoniecznie prowadzą do powstawania podatności, ale mogą ułatwiać zadanie intruzowi. Twórcy aplikacji internetowych powinni wziąć pod uwagę następujące kwestie:

Komentarze Komentarze objaśniają kod źródłowy i są na ogół pożyteczne, ale te, które trafiają do stron WWW są widoczne również dla atakujących. Należy więc zastanowić się nad ich usunięciem przed instalacją aplikacji na środowisku produkcyjnym. W przypadku skryptów JavaScript warto rozważyć zastosowanie narzędzi zaciemniających (ang. *obfuscate*) kod.

Obsługa wyjątków Programiści powinni przewidzieć jakie niespodziewane sytuacje mogą się pojawić w czasie działania aplikacji i je obsłużyć. Informacje o wyjątkach, które prezentowane są użytkownikowi nie powinny być obszerne. Szczegóły aplikacja powinna zapisywać do logu.

Niebezpieczne praktyki programistyczne

Zaszyte dane uwierzytelniające (ang. *hard-coded credentials*) — Kod źródłowy aplikacji nie powinien zawierać żadnych domyślnych kont z niezmiennymi hasłami. Takie rozwiązania są znane jako *tylne furtki* (ang. *backdoor*) i mogą być stosunkowo łatwo wykorzystane przez intruzów.

Sytuacje hazardowe (ang. *race conditions*) — to problemy wywołane przez zależności czasowe w mechanizmach zabezpieczających. Najczęściej przybierają formę *time-of-checkdywizto--time-of-use* (TOCTTOU lub TOC/TOU) i polegają na sprawdzaniu uprawnień do zasobu na długo przed jego faktycznym zażądaniem lub użyciem.

Niebezpieczne praktyki programistyczne

Niezabezpieczone API obecnie w użyciu są głównie dwie wersje interfejsu programistycznego aplikacji internetowej: SOAP i REST API. Pierwsza jest pochodną mechanizmu zdalnych wywołań procedur (XML RPC) i również bazuje na XML. Jest także coraz mniej popularna. Druga bazuje na protokole HTTP. Zabezpieczenie API aplikacji internetowej powinno obejmować szyfrowanie komunikacji i stosowanie kluczy dla niepublicznych API.

Niepodpisany kod Zarówno aplikacje biurkowe, jak i mobilne dostępne w repozytoriach lub nawet w wersji instalacyjnej są najczęściej podpisywane kryptograficznie przez twórców. Warto rozważyć zastosowanie takiego rozwiązania również dla aplikacji internetowych, szczególnie tych o zastosowaniach krytycznych.

Narzędzia

Steganografia

Przeprowadzanie testów penetracyjnych aplikacji, tak jak testowanie bezpieczeństwa innych elementów systemu, wymaga użycia odpowiednich narzędzi. Mogą to być narzędzia związane ze steganografią, czyli ukrywaniem innej treści w plikach obrazów lub audio, poprzez umieszczenie jej np. w najmniej znaczących bitach składowych koloru lub poprzez modyfikację tylko określonych pikseli. Ta technika może posłużyć pentesterom np. do przesyłania plików wykonywalnych z oprogramowaniem, które może być wykryte i zablokowane np. przez programy antywirusowe, albo do przesyłania ze spenetrowanego systemu zdobytych informacji w sposób nieprzyciągający uwagi. Należy jednak pamiętać, że informacja jaka może być przemycana w plikach z obrazami i nagraniami ma ograniczoną wielkość, więc w praktyce rzadko korzysta się z tej metody.

Narzędzia

Steganografia

Przykładami narzędzi umożliwiających korzystanie z steganografii opartej na obrazach są *OpenStego* (Windows) i `steghide` (Linux, MacOS). To ostatnie stosuje algorytm oparty na grafach, aby tak umieścić informację w obrazie, żeby nie mogła być wykryta przy pomocy podstawowych testów statystycznych. Z kolei *Coagula* (Windows) umożliwia ukrywanie treści w plikach audio, a *stegsnow* w plikach tekstowych, przy pomocy białych znaków (ang. *whitespaces*).
Do analizy i ujawniania utajnionych poprzez steganografię treści służy stosujący metodę siłową `stegcrack` (Linux), przeznaczony dla obrazów. W przypadku plików audio można zastosować *Sonic Visualiser* (wszystkie systemy). Wyszukiwarka obrazów TinEye także może być pomocna w procesie analizy obrazów z ukrytą treścią, gdyż pozwala znaleźć ich oryginalne wersje.

Narzędzia

SAST

Narzędzia SAST (ang. *Static Application Security Testing*) wykonują analizę kodu źródłowego aplikacji, bez jego uruchamiania i wskazują miejsca występowania potencjalnych luk bezpieczeństwa. Są one dostępne zarówno jako pełnopłatne, komercyjne oprogramowania, jak również jako bezpłatne aplikacje typu *open source*. Do pierwszej kategorii zaliczają się *Coverity* firmy *Synopsys* oraz *Klocwork* firmy *Perfoce*. Z kolei w drugiej kategorii należy wymienić *SonarQube* (dostępny również w wersji płatnej) oraz *FindBugs*.

Narzędzia

DAST

Istnieje kilka kategorii narzędzi DAST (ang. *Dynamic Application Security Testing*), czyli takich, które umożliwiają testowanie wykonującego się oprogramowania. W przypadku aplikacji internetowych jednymi z najbardziej przydatnych są *pośrednicy przechwytujący* (ang. *interception proxies*). Zgodnie z nazwą umożliwiają one przechwytywanie żądań przeglądarki WWW, lub innego klienta, do serwera i ich modyfikację. Mogą również przechwytywać odpowiedzi. Do najpopularniejszych należą ▸ OWASP ZAP i ▸ Burp Suite. Oba narzędzia mają wbudowaną przeglądarkę WWW, ale mogą także współpracować z zewnętrznymi klientami i obsługują protokół HTTPS. Ponadto posiadają one wiele przydatnych dodatków. Niektóre z nich są darmowe, jak *JavaSerialKiller* dla *BurpSuite*, służący do testowania niebezpiecznej deserializacji w języku Java.

Narzędzia
DAST

Narzędzie OWASP ZAP jest aplikacją typu *open source*, dostępną bezpłatnie. Umożliwia ono, oprócz przechwytywania żądań i odpowiedzi, aktywne i pasywne skanowanie aplikacji internetowej, wyszukiwanie plików i katalogów (wbudowane narzędzie DirBuster), testowanie rozmyte, indeksowanie struktury witryn, i wiele innych czynności. Posiada również mechanizm zabezpieczający, uniemożliwiający przypadkowe wykonanie operacji mogących mieć destruktywne skutki oraz opcję HUD ułatwiającą opanowanie jego obsługi. OWASP ZAP daje możliwość prowadzenia zarówno testów manualnych, jak i automatycznych.

Narzędzia
DAST

Burp Suite jest oprogramowaniem dostępnym zarówno w wersji bezpłatnej (*Community*), jak i w pełni odpłatnej (*Professional*). Wersja bezpłatna jest okrojona, ale wystarczająca w większości testów penetracyjnych. Przechwycone żądanie może być przesłane do zintegrowanych w tym oprogramowaniu narzędzi, umożliwiających jego modyfikację (*Intercepter*), powtórzenie (*Repeater*), wielokrotne wysłanie w zmienionej formie (*Intruder*), badanie losowości parametrów (ang. *Sequencer*) oraz rozkodowanie/zakodowanie jego części (*Decoder*). W wersji bezpłatnej Burp Suite możliwości automatyzacji testów są ograniczone.
Warto zauważyć, że w testach penetracyjnych mogą być przydatne także narzędzia dla programistów wbudowane w większość popularnych przeglądarek. Dodatkowo dla przeglądarki Firefox opracowano dodatek o nazwie ▸ Tamper Data.

Narzędzia
DAST

Zarówno w przypadku aplikacji internetowych, jak i innego oprogramowania można zastosować narzędzia nazywane *fuzzerami* (ang. *fuzzers*), które umożliwiają *testowanie rozmyte*. Polega ono na automatycznym i szybkim tworzeniu wielu wariantów danych wejściowych, które potencjalnie mogą prowadzić do ujawnienie istnienia podatności w testowanym oprogramowaniu. Kolejne wersje danych wejściowych mogą być tworzone w sposób pseudolosowy, albo przy użyciu bardziej zaawansowanych metod, np. algorytmów genetycznych. Przykładami takich narzędzi są *Peach Fuzzer* (komercyjny) i *American Fuzzy Lop* — AFL (darmowy).

Narzędzia
DAST

W analizie wykonywalnej postaci oprogramowania mogą być przydatne debuggery, od tych ogólnego przeznaczenia, takich jak GNU *Debugger* — GDB, *OllyDbg*, *WinDbg*, czy debugger wbudowany w framework *Covenant*, po specjalizowane narzędzia przeznaczone do prac związanych z bezpieczeństwem, takich jak ▸ *Immunity Debugger*, ▸ IDA i ▸ *Ghidra*.

Narzędzia
DAST

Z innych narzędzi, które zaliczają się do kategorii DAST, warto wymienić skanery (ang. *scanners*), umożliwiające poszukiwanie określonych podatności w działającym oprogramowaniu. Przykładem takiego narzędzia może być *Gobuster*, który umożliwia wykrycie nieupublicznionych plików i katalogów w aplikacjach internetowych, poddomen DNS, otwartych kubełów w środowiskach chmurowych Amazon i Google, serwerów TFTP oraz nazw wirtualnych hostów.
W przypadku aplikacji mobilnych przydatne mogą być takie narzędzia, jak *Dozer*, pozwalający atakować i weryfikować bezpieczeństwo aplikacji przeznaczonych dla systemu Android oraz APKX i APK *Studio*, pozwalające na dekompilację pakietów aplikacji dla tego systemu operacyjnego.

Pytania

?

Koniec

Dziękuję Państwu za uwagę!