Atakowanie kontroli dostępu

W ramach podstawowych mechanizmów bezpieczeństwa aplikacji kontrola dostępu jest logicznie zbudowana w oparciu o uwierzytelnianie i zarządzanie sesją. Do tej pory widziałeś, jak aplikacja może najpierw zweryfikować tożsamość użytkownika, a następnie potwierdzić, że określona sekwencja otrzymanych żądań pochodzi od tego samego użytkownika. Głównym powodem, dla którego aplikacja musi robić te rzeczy - pod względem bezpieczeństwa, jest m.in. najmniej - jest tak, ponieważ potrzebuje sposobu, aby zdecydować, czy powinien zezwolić danemu żądaniu na wykonanie jego próby działania lub uzyskać dostęp do zasobów, o które prosi. Kontrole dostępu są krytycznym mechanizmem obronnym w aplikacji, ponieważ są odpowiedzialne za podejmowanie tych kluczowych decyzji. Kiedy są wadliwe, osoba atakująca często może zagrozić całej aplikacji, przejmując kontrolę nad funkcjami administracyjnymi i uzyskując dostęp do poufnych danych należących do każdego innego użytkownika. Jak zauważono w Części 1, zepsute mechanizmy kontroli dostępu należą do najczęściej spotykanych kategorii luk w zabezpieczeniach aplikacji internetowych, dotykając aż 71 procent aplikacji ostatnio testowanych przez autorów. Bardzo często spotyka się aplikacje, które zadają sobie trud implementacji solidnych mechanizmów uwierzytelniania i zarządzania sesjami, tylko po to, by zmarnować tę inwestycję, zaniedbując zbudowanie na nich skutecznej kontroli dostępu. Jednym z powodów, dla których te słabości są tak powszechne, jest to, że kontrole kontroli dostępu muszą być przeprowadzane dla każdego żądania i każdej operacji na zasobie, którą konkretny użytkownik próbuje wykonać w określonym czasie. I w przeciwieństwie do wielu innych klas kontroli, jest to decyzja projektowa, którą musi podjąć człowiek; nie można go rozwiązać za pomocą technologii. Luki w zabezpieczeniach kontroli dostępu są koncepcyjnie proste: aplikacja pozwala zrobić coś, czego nie powinieneś. Różnice między poszczególnymi wadami tak naprawdę sprowadzają się do różnych sposobów manifestowania się tej podstawowej wady i różnych technik, które należy zastosować, aby ją wykryć. W tej części opisano wszystkie te techniki, pokazując, w jaki sposób można wykorzystać różne rodzaje zachowań w aplikacji do wykonywania nieautoryzowanych działań i uzyskiwania dostępu do chronionych danych.

Typowe luki w zabezpieczeniach

Kontrolę dostępu można podzielić na trzy szerokie kategorie: pionową, poziomą i zależną od kontekstu. Kontrola dostępu pionowego umożliwia różnym typom użytkowników dostęp do różnych części funkcjonalności aplikacji. W najprostszym przypadku zazwyczaj wiąże się to z podziałem na zwykłych użytkowników i administratorów. W bardziej skomplikowanych przypadkach. Pionowe kontrole dostępu mogą obejmować szczegółowe role użytkowników przyznające dostęp do określonych funkcji, przy czym każdemu użytkownikowi przydziela się pojedynczą rolę lub kombinację różnych ról. Kontrola dostępu poziomego umożliwia użytkownikom dostęp do pewnego podzbioru szerszego zakresu zasobów tego samego typu. Na przykład aplikacja poczty internetowej może umożliwiać odczytywanie wiadomości e-mail tylko z Twojego konta, bank internetowy może zezwalać na przesyłanie pieniędzy tylko z Twojego konta, a aplikacja do zarządzania przepływem pracy może umożliwiać aktualizowanie przydzielonych Ci zadań, ale czytać tylko zadania przydzielone innym osobom. Zależna od kontekstu kontrola dostępu gwarantuje, że dostęp użytkowników jest ograniczony do tego, co jest dozwolone w bieżącym stanie aplikacji. Na przykład, jeśli użytkownik śledzi wiele etapów w ramach procesu, zależne od kontekstu kontrole dostępu mogą uniemożliwić użytkownikowi dostęp do etapów poza ustaloną kolejnością. W wielu przypadkach pionowe i poziome kontrole dostępu są ze sobą powiązane. Na przykład aplikacja do planowania zasobów przedsiębiorstwa może pozwalać każdemu pracownikowi odpowiedzialnemu za księgowość na płacenie faktur za określoną jednostkę organizacyjną i za żadną inną. Z drugiej strony, kierownik ds. rozrachunków z dostawcami może mieć prawo do płacenia faktur za dowolną jednostkę. Podobnie urzędnicy mogą płacić faktury na niewielkie kwoty, ale większe faktury muszą być opłacane przez kierownika. Dyrektor finansowy może mieć wgląd w płatności i

paragony za faktury dla każdej jednostki organizacyjnej w firmie, ale może nie mieć uprawnień do opłacania faktur. Kontrola dostępu nie działa, jeśli dowolny użytkownik może uzyskać dostęp do funkcji lub zasobów, do których nie jest upoważniony. Istnieją trzy główne typy ataków na kontrolę dostępu, odpowiadające trzem kategoriom kontroli:

* Pionowa eskalacja uprawnień ma miejsce, gdy użytkownik może wykonywać funkcje, na które nie pozwala mu przypisana mu rola. Na przykład, jeśli zwykły użytkownik może wykonywać funkcje administracyjne lub urzędnik może płacić faktury dowolnej wielkości, kontrola dostępu jest zepsuta.

* Pozioma eskalacja uprawnień ma miejsce, gdy użytkownik może przeglądać lub modyfikować zasoby, do których nie jest uprawniony. Na przykład, jeśli możesz używać aplikacji poczty internetowej do odczytywania wiadomości e-mail innych osób lub jeśli urzędnik ds. Płatności może przetwarzać faktury dla jednostki organizacyjnej innej niż jego własna, kontrola dostępu nie działa.

* Wykorzystanie logiki biznesowej ma miejsce, gdy użytkownik może wykorzystać lukę w maszynie stanu aplikacji, aby uzyskać dostęp do kluczowego zasobu. Na przykład użytkownik może być w stanie ominąć krok płatności w sekwencji realizacji zakupu.

Często spotyka się przypadki, w których luka w poziomym podziale uprawnień aplikacji może natychmiast doprowadzić do ataku z eskalacją pionową. Na przykład, jeśli użytkownik znajdzie sposób na ustawienie hasła innego użytkownika, może zaatakować konto administratora i przejąć kontrolę nad aplikacją. W dotychczas opisanych przypadkach zepsuta kontrola dostępu umożliwia użytkownikom, którzy uwierzytelnili się w aplikacji w określonym kontekście użytkownika, wykonywanie czynności lub dostęp do danych, do których ten kontekst ich nie upoważnia. Jednak w najpoważniejszych przypadkach naruszenia kontroli dostępu całkowicie nieupoważnieni użytkownicy mogą uzyskać dostęp do funkcji lub danych, do których dostęp mają tylko uprzywilejowani uwierzytelnieni użytkownicy.

Całkowicie niezabezpieczona funkcjonalność

W wielu przypadkach złamanych kontroli dostępu każdy, kto zna odpowiedni adres URL, może uzyskać dostęp do wrażliwych funkcji i zasobów. Na przykład w przypadku wielu aplikacji każdy, kto odwiedzi określony adres URL, może w pełni korzystać z jego funkcji administracyjnych:

https://wahh-app.com/admin/

W takiej sytuacji aplikacja zazwyczaj wymusza kontrolę dostępu tylko w następującym zakresie: użytkownicy, którzy zalogowali się jako administratorzy, widzą link do tego adresu URL w swoim interfejsie użytkownika, a inni użytkownicy nie. Ta kosmetyczna różnica jest jedynym mechanizmem „chroniącym” wrażliwą funkcjonalność przed nieautoryzowanym użyciem. Czasami adres URL, który zapewnia dostęp do zaawansowanych funkcji, może być trudniejszy do odgadnięcia, a nawet dość tajemniczy:

https://wahh-app.com/menus/secure/ff457/DoAdminMenu2.jsp

Tutaj dostęp do funkcji administracyjnych jest chroniony przy założeniu, że atakujący nie będzie znał ani nie odkryje tego adresu URL. Osobie z zewnątrz trudniej jest skompromitować aplikację, ponieważ jest mniej prawdopodobne, że odgadnie adres URL, za pomocą którego może to zrobić. W niektórych aplikacjach, w których poufne funkcje są ukryte za adresami URL które nie są łatwe do odgadnięcia, osoba atakująca może często być w stanie je zidentyfikować poprzez dokładną kontrolę kodu po stronie klienta. Wiele aplikacji używa języka JavaScript do dynamicznego tworzenia interfejsu użytkownika w kliencie. Zwykle działa to poprzez ustawienie różnych flag dotyczących statusu

użytkownika, a następnie dodanie poszczególnych elementów do interfejsu użytkownika na ich podstawie:

```
var isAdmin = false;

...

if (isAdmin)

{

adminMenu.addItem("/menus/secure/ff457/addNewPortalUser2.jsp",

"create a new user");

}
```

Tutaj osoba atakująca może po prostu przejrzeć JavaScript, aby zidentyfikować adresy URL funkcji administracyjnych i spróbować uzyskać do nich dostęp. W innych przypadkach komentarze HTML mogą zawierać odniesienia lub wskazówki dotyczące adresów URL, które nie są powiązane z zawartością na ekranie. W części 4 omówiono różne techniki, za pomocą których osoba atakująca może zbierać informacje o zawartości ukrytej w aplikacji.

PWSZECHNY MIT

„Żaden użytkownik o niskich uprawnieniach nie będzie znał tego adresu URL. Nigdzie w aplikacji nie odwołujemy się do niego".

Brak jakiejkolwiek prawdziwej kontroli dostępu nadal stanowi poważną lukę, niezależnie od tego, jak łatwo byłoby odgadnąć adres URL. Adresy URL nie mają statusu tajemnicy ani w samej aplikacji, ani w rękach jej użytkowników. Są one wyświetlane na ekranie i pojawiają się w historii przeglądarki oraz dziennikach serwerów WWW i serwerów proxy. Użytkownicy mogą je zapisywać, dodawać do zakładek lub przesyłać e-mailem. Zwykle nie są one okresowo zmieniane, jak powinno być w przypadku haseł. Kiedy użytkownicy zmieniają role służbowe, a ich dostęp do funkcji administracyjnych musi zostać cofnięty, nie ma możliwości usunięcia ich wiedzy o określonym adresie URL.

Bezpośredni dostęp do metod

Specyficzny przypadek niezabezpieczonej funkcjonalności może wystąpić, gdy aplikacje ujawniają adresy URL lub parametry, które w rzeczywistości są zdalnymi wywołaniami metod API, zwykle tymi, które są udostępniane przez interfejs Java. Dzieje się tak często, gdy kod po stronie serwera jest przenoszony do komponentu rozszerzenia przeglądarki i tworzone są kody pośredniczące metod, dzięki czemu kod może nadal wywoływać metody po stronie serwera, których wymaga do działania. Poza tą sytuacją można zidentyfikować niektóre przypadki bezpośredniego dostępu do metod, w których adresy URL lub parametry używają standardowych konwencji nazewnictwa języka Java, takich jak getBalance i isExpired. Zasadniczo żądania określające interfejs API po stronie serwera do wykonania nie muszą być mniej bezpieczne niż żądania określające skrypt po stronie serwera lub inny zasób. W praktyce jednak ten typ mechanizmu często zawiera luki. Często klient wchodzi w bezpośrednią interakcję z metodami API po stronie serwera i omija normalne kontrole aplikacji dotyczące dostępu lub nieoczekiwanych wektorów danych wejściowych. Istnieje również szansa, że istnieje inna funkcjonalność, którą można wywołać w ten sposób i która nie jest chroniona przez żadne kontrole, przy założeniu, że nigdy nie mogłaby zostać bezpośrednio wywołana przez klientów aplikacji internetowych. Często istnieje potrzeba zapewnienia użytkownikom dostępu do określonych metod, ale zamiast tego otrzymują oni dostęp do wszystkich metod. Dzieje się tak dlatego, że programista nie

jest w pełni świadomy, który podzbiór metod ma być proxy i zapewnia dostęp do wszystkich metod, albo dlatego, że interfejs API używany do mapowania ich na serwer HTTP zapewnia domyślnie dostęp do wszystkich metod. Poniższy przykład przedstawia wywołanie metody getCurrentUserRoles z poziomu interfejsu securityCheck:

http://wahh-app.com/public/securityCheck/getCurrentUserRoles

W tym przykładzie oprócz testowania kontroli dostępu w metodzie getCurrentUserRoles należy sprawdzić, czy istnieją inne metody o podobnych nazwach, takie jak getAllUserRoles, getAllRoles, getAllUsers i getCurrentUserPermissions. Dalsze rozważania specyficzne dla testowania bezpośredniego dostępu do metod są opisane w dalszej części tej części.

Funkcje oparte na identyfikatorze

Kiedy funkcja aplikacji jest używana do uzyskania dostępu do określonego zasobu, często zdarza się, że identyfikator żądanego zasobu jest przekazywany do serwera w parametrze żądania, w ciągu zapytania URL lub w treści zapytania. Żądanie POST. Na przykład aplikacja może użyć następującego adresu URL do wyświetlenia określonego dokumentu należącego do określonego użytkownika:

https://wahh-app.com/ViewDocument.php?docid=1280149120

Gdy użytkownik będący właścicielem dokumentu jest zalogowany, łącze do tego adresu URL jest wyświetlane na stronie Moje dokumenty użytkownika. Inni użytkownicy nie widzą linku. Jeśli jednak kontrola dostępu zostanie naruszona, każdy użytkownik, który poprosi o odpowiedni adres URL, może wyświetlić dokument dokładnie w taki sam sposób, jak upoważniony użytkownik.

WSKAZÓWKA: Ten typ luki często pojawia się, gdy główna aplikacja łączy się z zewnętrznym systemem lub komponentem zaplecza. Współdzielenie modelu bezpieczeństwa opartego na sesjach między różnymi systemami, które mogą być oparte na różnych technologiach, może być trudne. W obliczu tego problemu programiści często wybierają skróty i odchodzą od tego modelu, korzystając z przesłanego przez klienta parametru do podejmowania decyzji kontroli dostępu.

W tym przykładzie atakujący chcący uzyskać nieautoryzowany dostęp musi znać nie tylko nazwę strony aplikacji ViewDocument.php), ale także identyfikator dokumentu, który chce przeglądać. Czasami identyfikatory zasobów są generowane w wysoce nieprzewidywalny sposób; na przykład mogą to być losowo wybrane identyfikatory GUID. W innych przypadkach można je łatwo odgadnąć; na przykład mogą to być liczby generowane sekwencyjnie. Jednak aplikacja jest podatna na ataki w obu przypadkach. Jak opisano wcześniej, adresy URL nie mają statusu tajemnicy i to samo dotyczy identyfikatorów zasobów. Często osoba atakująca, która chce odkryć identyfikatory zasobów innych użytkowników, może znaleźć w aplikacji miejsce, które je ujawnia, na przykład dzienniki dostępu. Nawet jeśli nie można łatwo odgadnąć identyfikatorów zasobów aplikacji, aplikacja nadal jest podatna na ataki, jeśli nie kontroluje właściwie dostępu do tych zasobów. W przypadkach, gdy identyfikatory są łatwe do przewidzenia, problem jest jeszcze poważniejszy i łatwiejszy do wykorzystania.

WSKAZÓWKA: Dzienniki aplikacji to często kopalnia informacji. Mogą zawierać liczne elementy danych, które można wykorzystać jako identyfikatory do badania funkcjonalności, do których uzyskuje się dostęp w ten sposób. Identyfikatory często spotykane w dziennikach aplikacji obejmują nazwy użytkowników, numery identyfikacyjne użytkowników, numery kont, identyfikatory dokumentów, grupy i role użytkowników oraz adresy e-mail.

UWAGA: Oprócz tego, że jest używany jako odniesienie do zasobów opartych na danych w aplikacji, ten rodzaj identyfikatora jest często używany do odwoływania się do funkcji samej aplikacji. Jak

widzieliśmy w części 4, aplikacja może udostępniać różne funkcje za pośrednictwem pojedynczej strony, która akceptuje nazwę funkcji lub identyfikator jako parametr. Ponownie w tej sytuacji kontrola dostępu może nie sięgać głębiej niż obecność lub brak określonych adresów URL w interfejsach różnych typów użytkowników. Jeśli osoba atakująca może określić identyfikator wrażliwej funkcji, może uzyskać do niej dostęp w taki sam sposób, jak bardziej uprzywilejowany użytkownik.

Funkcje wieloetapowe

Wiele rodzajów funkcji w aplikacji jest realizowanych w kilku etapach, obejmujących wysyłanie wielu żądań od klienta do serwera. Na przykład funkcja dodawania nowego użytkownika może obejmować wybranie tej opcji z menu obsługi użytkownika, wybranie działu i roli użytkownika z rozwijanych list, a następnie wprowadzenie nowej nazwy użytkownika, początkowego hasła i innych informacji. Często spotyka się aplikacje, w których podjęto wysiłki w celu ochrony tego rodzaju wrażliwych funkcji przed nieupoważnionym dostępem, ale w których zastosowane mechanizmy kontroli dostępu są zepsute z powodu błędnych założeń dotyczących sposobu wykorzystania tych funkcji. W poprzednim przykładzie, gdy użytkownik próbuje załadować menu obsługi użytkownika i wybiera opcję dodania nowego użytkownika, aplikacja może zweryfikować, czy użytkownik ma wymagane uprawnienia i zablokować dostęp, jeśli użytkownik ich nie posiada. Jeśli jednak atakujący przejdzie bezpośrednio do etapu określania działu użytkownika i innych szczegółów, może nie być skutecznej kontroli dostępu. Twórcy nieświadomie założyli, że każdy użytkownik, który dotrze do dalszych etapów procesu, musi posiadać odpowiednie uprawnienia, ponieważ zostało to zweryfikowane na wcześniejszych etapach. W rezultacie każdy użytkownik aplikacji może dodać nowe konto użytkownika administracyjnego, a tym samym przejąć pełną kontrolę nad aplikacją, uzyskując dostęp do wielu innych funkcji, których kontrola dostępu jest z natury niezawodna. Autorzy napotkali tego typu luki nawet w najbardziej krytycznych pod względem bezpieczeństwa aplikacjach internetowych — tych wdrożonych przez banki internetowe. Dokonywanie transferu środków w aplikacji bankowej zazwyczaj obejmuje wiele etapów, częściowo po to, aby zapobiec przypadkowym pomyłkom podczas żądania przelewu. Ten wieloetapowy proces obejmuje przechwytywanie różnych elementów danych od użytkownika na każdym etapie. Dane te są dokładnie sprawdzane przy pierwszym przesłaniu, a następnie zazwyczaj przekazywane są do każdego kolejnego etapu, przy użyciu ukrytych pól w formacie HTML. Jeśli jednak aplikacja nie zweryfikuje ponownie wszystkich tych danych na ostatnim etapie, atakujący może potencjalnie ominąć sprawdzanie serwera. Na przykład aplikacja może zweryfikować, czy wybrany do przelewu rachunek źródłowy należy do bieżącego użytkownika, a następnie zapytać o szczegóły dotyczące rachunku docelowego i kwoty przelewu. Jeśli użytkownik przechwyci końcowe żądanie POST tego procesu i zmodyfikuje numer konta źródłowego, może wykonać poziomą eskalację uprawnień i przelać środki z konta należącego do innego użytkownika.

Pliki statyczne

W większości przypadków użytkownicy uzyskują dostęp do chronionych funkcji i zasobów, wysyłając żądania do dynamicznych stron, które są wykonywane na serwerze. Obowiązkiem każdej takiej strony jest przeprowadzenie odpowiednich kontroli kontroli dostępu i potwierdzenie, że użytkownik ma odpowiednie uprawnienia do wykonania czynności, którą próbuje wykonać. Jednak w niektórych przypadkach żądania dotyczące chronionych zasobów są kierowane bezpośrednio do samych zasobów statycznych, które znajdują się w katalogu głównym serwera. Na przykład wydawca online może zezwolić użytkownikom na przeglądanie jego katalogu książek i kupowanie e-booków do pobrania. Po dokonaniu płatności użytkownik jest kierowany do adresu URL pobierania, takiego jak ten:

https://wahh-books.com/download/9780636628104.pdf

Ponieważ jest to całkowicie statyczny zasób, jeśli jest hostowany na tradycyjnym serwerze WWW, jego zawartość jest po prostu zwracana bezpośrednio przez serwer i nie jest wykonywany żaden kod na poziomie aplikacji. W związku z tym zasób nie może zaimplementować żadnej logiki w celu sprawdzenia, czy żądający użytkownik ma wymagane uprawnienia. Gdy w ten sposób uzyskuje się dostęp do zasobów statycznych, jest wysoce prawdopodobne, że żadna skuteczna kontrola dostępu ich nie chroni i że każdy, kto zna schemat nazewnictwa adresów URL, może to wykorzystać, aby uzyskać dostęp do dowolnych zasobów. W tym przypadku nazwa dokumentu wygląda podejrzanie jak numer ISBN, który umożliwiłby atakującemu szybkie pobranie każdego e-booka wyprodukowanego przez wydawcę! Niektóre rodzaje funkcji są szczególnie podatne na tego rodzaju problemy, w tym witryny finansowe zapewniające dostęp do statycznych dokumentów dotyczących firm, takich jak raporty roczne, dostawcy oprogramowania udostępniający pliki binarne do pobrania oraz funkcje administracyjne, które zapewniają dostęp do statycznych plików dziennika i innych gromadzonych poufnych danych w aplikacji.

Błędna konfiguracja platformy

Niektóre aplikacje używają elementów sterujących na serwerze WWW lub warstwie platformy aplikacji do kontrolowania dostępu. Zazwyczaj dostęp do określonych ścieżek URL jest ograniczony w zależności od roli użytkownika w aplikacji. Na przykład dostęp do ścieżki /admin może zostać zabroniony użytkownikom, którzy nie należą do grupy Administratorzy. Zasadniczo jest to całkowicie uzasadniony sposób kontrolowania dostępu. Jednak błędy popełnione w konfiguracji kontroli na poziomie platformy mogą często umożliwić nieautoryzowany dostęp. Konfiguracja na poziomie platformy ma zwykle postać reguł podobnych do zasad polityki zapory sieciowej, które zezwalają na dostęp lub odmawiają go w oparciu o następujące elementy:

* Metoda żądania HTTP

* Ścieżka adresu URL

* Rola użytkownika

Jak opisano w rozdziale 3, pierwotnym celem metody GET jest pobieranie informacji, a celem metody POST jest wykonywanie działań zmieniających dane lub stan aplikacji. Jeśli nie zadba się o opracowanie reguł, które dokładnie zezwalają na dostęp w oparciu o prawidłowe metody HTTP i ścieżki URL, może to prowadzić do nieautoryzowanego dostępu. Na przykład, jeśli funkcja administracyjna do tworzenia nowego użytkownika używa metody POST, platforma może mieć regułę odmowy, która zabrania metody POST i zezwala na wszystkie inne metody. Jeśli jednak kod na poziomie aplikacji nie weryfikuje, czy wszystkie żądania dotyczące tej funkcji faktycznie korzystają z metody POST, osoba atakująca może być w stanie obejść kontrolę, przesyłając to samo żądanie za pomocą metody GET. Ponieważ większość interfejsów API na poziomie aplikacji do pobierania parametrów żądania jest niezależna od metody żądania, osoba atakująca może po prostu podać wymagane parametry w ciągu URl.query żądania GET do nieautoryzowanego korzystanie z funkcji. Na pierwszy rzut oka bardziej zaskakujące jest to, że aplikacje nadal mogą być podatne na ataki, nawet jeśli reguła na poziomie platformy odmawia dostępu zarówno do metod GET, jak i POST. Dzieje się tak, ponieważ żądania korzystające z innych metod HTTP mogą ostatecznie być obsługiwane przez ten sam kod aplikacji, który obsługuje żądania GET i POST. Jednym z przykładów jest metoda HEAD. Zgodnie ze specyfikacją serwery powinny odpowiadać na żądanie HEAD z tymi samymi nagłówkami, których użyłyby w odpowiedzi na odpowiednie żądanie GET, ale bez treści wiadomości. Dlatego większość platform poprawnie obsługuje żądania HEAD, wykonując odpowiednią procedurę obsługi GET i po prostu zwraca wygenerowane nagłówki HTTP. Żądania GET często mogą być wykorzystywane do wykonywania wrażliwych działań, ponieważ albo sama aplikacja używa do tego celu żądań GET (wbrew specyfikacji), albo dlatego, że nie weryfikuje, czy używana jest

metoda POST. Jeśli osoba atakująca może użyć żądania HEAD w celu dodania konta użytkownika administracyjnego, może żyć bez otrzymywania treści wiadomości w odpowiedzi. W niektórych przypadkach platformy obsługują żądania korzystające z nierozpoznanych metod HTTP, po prostu przekazując je do procedury obsługi żądań GET. W tej sytuacji kontrole na poziomie platformy, które po prostu odrzucają określone metody HTTP, można ominąć, określając w żądaniu dowolną nieprawidłową metodę HTTP. Część 18 zawiera konkretny przykład tego typu luki w zabezpieczeniach produktu platformy aplikacji internetowych.

Niebezpieczne metody kontroli dostępu

Niektóre aplikacje wykorzystują zasadniczo niepewny model kontroli dostępu, w którym decyzje dotyczące kontroli dostępu są podejmowane na podstawie parametrów żądania przesłanych przez klienta lub innych warunków pozostających pod kontrolą atakującego.

Kontrola dostępu oparta na parametrach

W niektórych wersjach tego modelu aplikacja określa rolę użytkownika lub poziom dostępu w momencie logowania i od tego momentu przekazuje te informacje przez klienta w ukrytym polu formularza, pliku cookie lub zadanym parametrze ciągu zapytania. Podczas przetwarzania każdego kolejnego żądania aplikacja odczytuje ten parametr żądania i odpowiednio decyduje, jaki dostęp przyznać użytkownikowi. Na przykład administrator korzystający z aplikacji może zobaczyć następujące adresy URL:

https://wahh-app.com/login/home.jsp?admin=true

Adresy URL widoczne dla zwykłych użytkowników zawierają inny parametr lub nie zawierają go wcale. Każdy użytkownik, który zna parametr przypisany administratorom, może po prostu ustawić go we własnych żądaniach i tym samym uzyskać dostęp do funkcji administracyjnych. Ten rodzaj kontroli dostępu może czasami być trudny do wykrycia bez rzeczywistego korzystania z aplikacji jako użytkownik o wysokich uprawnieniach i identyfikowania żądań. Opisane w rozdziale 4 techniki wykrywania ukrytych parametrów żądania mogą być skuteczne w wykrywaniu mechanizmu podczas pracy tylko jako zwykły użytkownik.

Kontrola dostępu oparta na referencjach

W innych niebezpiecznych modelach kontroli dostępu aplikacja używa nagłówka HTTP Referer jako podstawy do podejmowania decyzji dotyczących kontroli dostępu. Na przykład aplikacja może ściśle kontrolować dostęp do głównego menu administracyjnego na podstawie uprawnień użytkownika. Ale gdy użytkownik złoży wniosek o pojedynczą funkcję administracyjną, aplikacja może po prostu sprawdzić, czy to żądanie było odesłane ze strony menu administracyjnego. Może zakładać, że użytkownik musiał uzyskać dostęp do tej strony, a zatem ma wymagane uprawnienia. Ten model jest oczywiście zasadniczo zepsuty, ponieważ nagłówek Referer jest całkowicie pod kontrolą użytkownika i można go ustawić na dowolną wartość.

Kontrola dostępu oparta na lokalizacji

Wiele firm ma wymagania prawne lub biznesowe, aby ograniczyć dostęp do zasobów w zależności od lokalizacji geograficznej użytkownika. Nie ograniczają się one do sektora finansowego, ale obejmują serwisy informacyjne i inne. W takich sytuacjach firma może zastosować różne metody lokalizacji użytkownika, z których najpowszechniejszą jest geolokalizacja aktualnego adresu IP użytkownika. Kontrola dostępu oparta na lokalizacji jest stosunkowo łatwa do obejścia przez atakującego. Oto kilka typowych metod ich obejścia:

* Korzystanie z internetowego serwera proxy, który znajduje się w wymaganej lokalizacji

* Korzystanie z VPN, który kończy się w wymaganej lokalizacji

* Korzystanie z urządzenia mobilnego obsługującego roaming danych

* Bezpośrednia manipulacja mechanizmami po stronie klienta dla geolokalizacji

Atakowanie kontroli dostępu

Przed rozpoczęciem sondowania aplikacji w celu wykrycia rzeczywistych luk w zabezpieczeniach kontroli dostępu należy poświęcić chwilę na przejrzenie wyników ćwiczeń z mapowania aplikacji. Musisz zrozumieć, jakie są rzeczywiste wymagania aplikacji w zakresie kontroli dostępu, a zatem na czym prawdopodobnie najbardziej owocne będzie skupienie uwagi.

KROKI HACKOWANIA

Oto kilka pytań, które należy wziąć pod uwagę podczas sprawdzania kontroli dostępu do aplikacji:

1. Czy funkcje aplikacji zapewniają poszczególnym użytkownikom dostęp do określonego podzbioru danych, które do nich należą?

2. Czy istnieją różne poziomy użytkowników, takie jak menedżerowie, przełożeni, goście itd., którzy mają dostęp do różnych funkcji?

3. Czy administratorzy używają funkcji wbudowanych w tę samą aplikację do jej konfigurowania i monitorowania?

4. Jakie zidentyfikowałeś funkcje lub zasoby danych w aplikacji, które najprawdopodobniej umożliwiłyby Ci eskalację obecnych uprawnień?

5. Czy są jakieś identyfikatory (za pomocą parametrów adresu URL treści wiadomości POST), które sygnalizują, że parametr jest używany do śledzenia poziomów dostępu?

Testowanie z różnymi kontami użytkowników

Najłatwiejszym i najskuteczniejszym sposobem sprawdzenia skuteczności kontroli dostępu do aplikacji jest uzyskanie dostępu do aplikacji przy użyciu różnych kont. W ten sposób możesz określić, czy zasoby i funkcje, do których jedno konto może uzyskać legalny dostęp, mogą być dostępne nielegalnie przez inne.

Testowanie z różnymi kontami użytkowników

Najłatwiejszym i najskuteczniejszym sposobem sprawdzenia skuteczności kontroli dostępu do aplikacji jest uzyskanie dostępu do aplikacji przy użyciu różnych kont. W ten sposób możesz określić, czy zasoby i funkcje, do których jedno konto może uzyskać legalny dostęp, mogą być dostępne nielegalnie przez inne.

KROKI HACKOWANIA

1. Jeśli aplikacja segreguje dostęp użytkowników do różnych poziomów funkcjonalności, najpierw użyj potężnego konta, aby zlokalizować wszystkie dostępne funkcje. Następnie spróbuj uzyskać do niego dostęp za pomocą konta o niższych uprawnieniach, aby przetestować eskalację uprawnień w pionie.

2. Jeśli aplikacja segreguje dostęp użytkowników do różnych zasobów (takich jak dokumenty), użyj dwóch różnych kont na poziomie użytkownika, aby sprawdzić, czy kontrola dostępu jest skuteczna lub

czy możliwa jest pozioma eskalacja uprawnień. Na przykład znajdź dokument, do którego jeden użytkownik może legalnie uzyskać dostęp, a inny nie, i spróbuj uzyskać do niego dostęp za pomocą konta drugiego użytkownika — albo żądając odpowiedniego adresu URL, albo przesyłając te same parametry POST z sesji drugiego użytkownika.

Dokładne testowanie kontroli dostępu aplikacji jest procesem czasochłonnym. Na szczęście niektóre narzędzia mogą pomóc w zautomatyzowaniu niektórych prac, dzięki czemu testowanie będzie szybsze i bardziej niezawodne. Pozwoli ci to skoncentrować się na tych częściach zadania, które do skutecznego wykonania wymagają ludzkiej inteligencji. Burp Suite umożliwia mapowanie zawartości aplikacji przy użyciu dwóch różnych kontekstów użytkownika. Następnie możesz porównać wyniki, aby dokładnie zobaczyć, gdzie treści, do których każdy użytkownik uzyskuje dostęp, są takie same lub różne.

KROKI HACKOWANIA

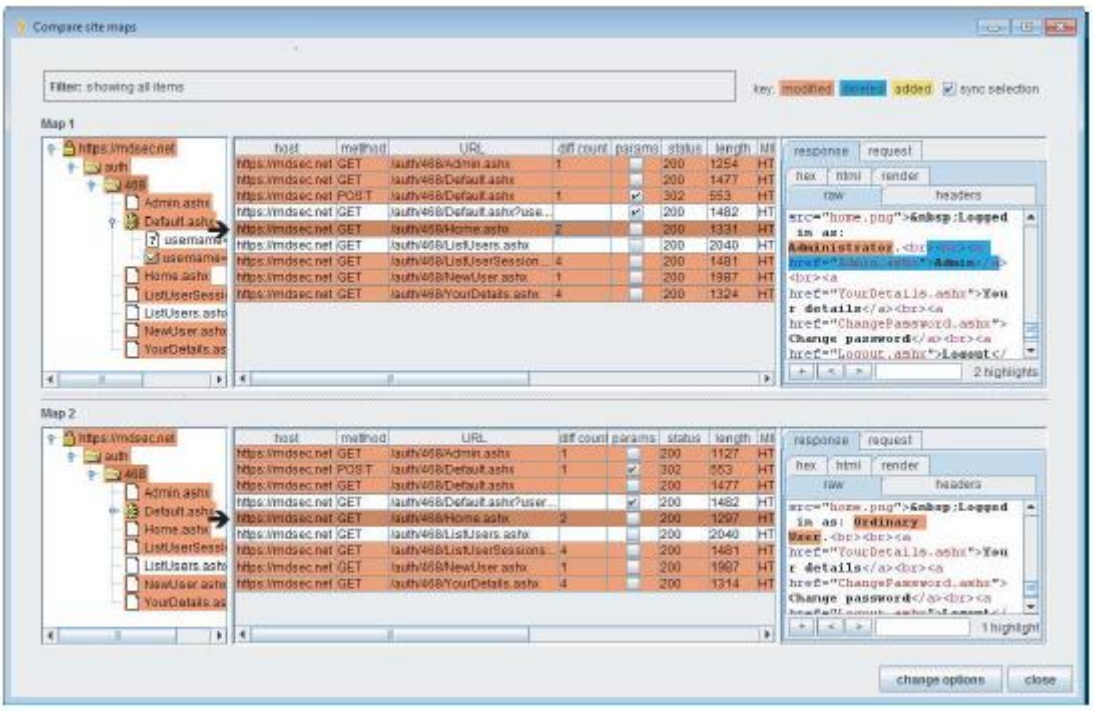
1. Po skonfigurowaniu Burp jako proxy i wyłączeniu przechwytywania przeglądaj całą zawartość aplikacji w jednym kontekście użytkownika. Jeśli testujesz pionową kontrolę dostępu, użyj do tego konta o wyższych uprawnieniach.

2. Przejrzyj zawartość mapy witryny Burp, aby upewnić się, że zidentyfikowałeś wszystkie funkcje, które chcesz przetestować. Następnie użyj menu kontekstowego, aby wybrać funkcję „porównaj mapy witryn".

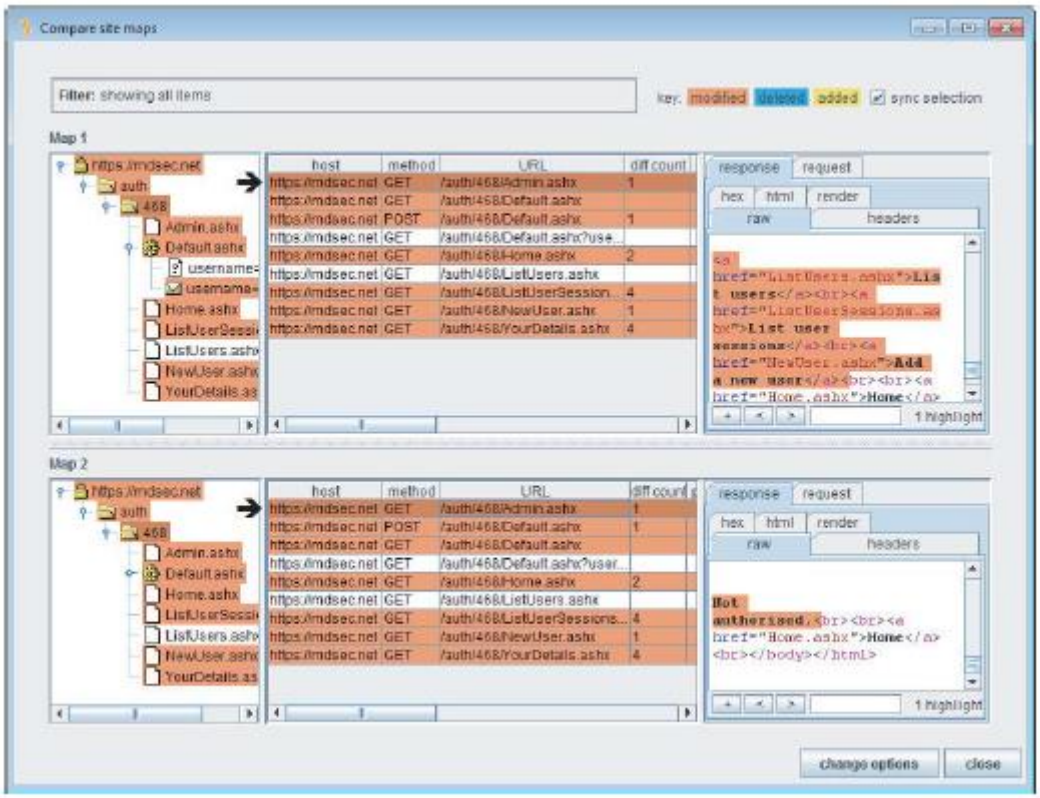
3. Aby wybrać drugą mapę witryny do porównania, możesz albo załadować ją z pliku stanu Burp, albo zlecić Burpowi dynamiczne żądanie pierwszej mapy witryny w kontekście nowej sesji. Aby przetestować poziomą kontrolę dostępu między użytkownikami tego samego typu, możesz po prostu załadować plik stanu, który zapisałeś wcześniej, po zmapowaniu aplikacji jako innego użytkownika. W przypadku testowania pionowej kontroli dostępu preferowane jest ponowne zażądanie mapy witryny o wysokich uprawnieniach jako użytkownik o niskich uprawnieniach, ponieważ zapewnia to pełne pokrycie odpowiednich funkcji.

4. Aby zażądać pierwszej mapy witryny w innej sesji, należy skonfigurować funkcję obsługi sesji Burp ze szczegółami sesji użytkownika o niskich uprawnieniach (na przykład rejestrując makro logowania lub dostarczając określony plik cookie do wykorzystania w upraszanie). Ta funkcja jest opisana bardziej szczegółowo w części 14. Może być również konieczne zdefiniowanie odpowiednich reguł zakresu, aby uniemożliwić Burp żądanie jakiejkolwiek funkcji wylogowania

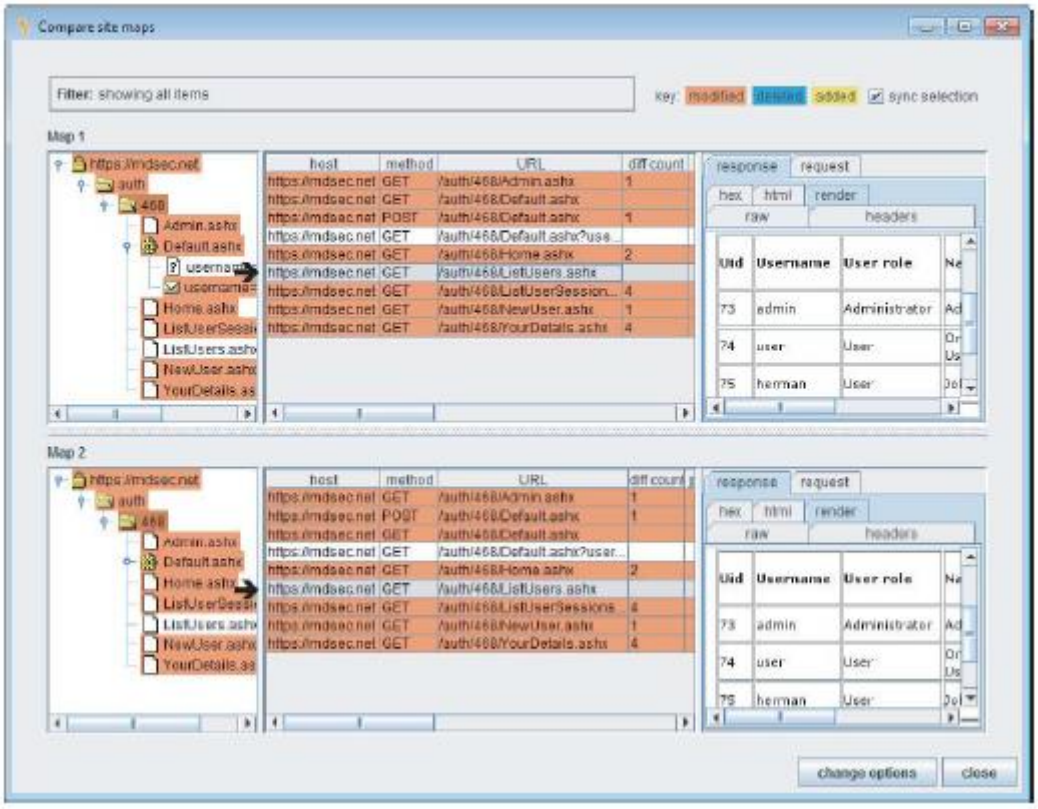
Rysunek przedstawia wyniki prostego porównania mapy witryny.



Jego kolorowa analiza różnic między mapami witryn pokazuje elementy, które zostały dodane, usunięte lub zmodyfikowane między dwiema mapami. W przypadku zmodyfikowanych elementów tabela zawiera kolumnę „liczba różnic”, która jest liczbą edycji wymaganych do zmodyfikowania elementu z pierwszej mapy na element z drugiej mapy. Ponadto po wybraniu elementu odpowiedzi są również kolorowane, aby pokazać lokalizacje tych zmian w odpowiedziach. Interpretacja wyników porównania map witryn wymaga ludzkiej inteligencji oraz zrozumienia znaczenia i kontekstu określonych funkcji aplikacji. Na przykład na rysunku 8.1 przedstawiono odpowiedzi, które są zwracane każdemu użytkownikowi, gdy przegląda on swoją stronę główną. Dwie odpowiedzi pokazują inny opis zalogowanego użytkownika, a użytkownik administracyjny ma dodatkową pozycję menu. Różnic tych należy się spodziewać i są one neutralne pod względem skuteczności kontroli dostępu do aplikacji, ponieważ dotyczą tylko interfejsu użytkownika. Rysunek przedstawia odpowiedź zwróconą, gdy każdy użytkownik zażąda strony administratora najwyższego poziomu.



Tutaj administrator widzi menu dostępnych opcji, podczas gdy zwykły użytkownik widzi komunikat „brak autoryzacji”. Różnice te wskazują, że kontrola dostępu jest stosowana prawidłowo. Rysunek przedstawia odpowiedź zwróconą, gdy każdy użytkownik zażąda funkcji administratora „lista użytkowników”.



Tutaj odpowiedzi są identyczne, co wskazuje, że aplikacja jest podatna na ataki, ponieważ zwykły użytkownik nie powinien mieć dostępu do tej funkcji i nie ma do niej żadnego odnośnika w swoim interfejsie użytkownika. Samo zbadanie drzewa mapy witryny i przyjrzenie się liczbie różnic między elementami nie wystarczy do oceny skuteczności kontroli dostępu do aplikacji. Dwie identyczne odpowiedzi mogą wskazywać na lukę (na przykład w funkcji administracyjnej, która ujawnia poufne informacje) lub mogą być nieszkodliwe (na przykład w niezabezpieczonej funkcji wyszukiwania). I odwrotnie, dwie różne odpowiedzi mogą nadal oznaczać, że istnieje luka w zabezpieczeniach (na przykład w funkcji administracyjnej, która zwraca inną zawartość za każdym razem, gdy uzyskuje się do niej dostęp) lub mogą być nieszkodliwe (na przykład na stronie wyświetlającej informacje o profilu aktualnie zalogowanego użytkownika). w użytkowniku). Z tych powodów w pełni zautomatyzowane narzędzia są generalnie nieskuteczne w identyfikowaniu luk w zabezpieczeniach kontroli dostępu. Wykorzystując funkcjonalność Burp do porównywania map witryn, możesz maksymalnie zautomatyzować ten proces, dając Ci wszystkie potrzebne informacje w gotowej formie i pozwalając wykorzystać swoją wiedzę na temat funkcjonalności aplikacji do zidentyfikowania rzeczywistych luk w zabezpieczeniach.

Testowanie procesów wieloetapowych

Podejście opisane w poprzedniej sekcji — porównywanie zawartości aplikacji podczas uzyskiwania dostępu w różnych kontekstach użytkownika — jest nieskuteczne podczas testowania niektórych procesów wieloetapowych. Tutaj, aby wykonać akcję, użytkownik zazwyczaj musi wykonać kilka żądań we właściwej kolejności, a aplikacja buduje pewien stan na temat działań użytkownika, gdy to robi. Zwykłe zażądanie każdego elementu na mapie witryny może nie spowodować poprawnej replikacji procesu, więc próba działania może zakończyć się niepowodzeniem z powodów innych niż stosowanie kontroli dostępu. Rozważmy na przykład funkcję administracyjną, aby dodać nowego użytkownika aplikacji. Może to obejmować kilka kroków, w tym załadowanie formularza w celu dodania użytkownika, przesłanie formularza z danymi nowego użytkownika, przejrzenie tych danych i potwierdzenie akcji. W niektórych przypadkach aplikacja może chronić dostęp do formularza początkowego, ale nie chronią strony obsługującej przesyłanie formularza ani strony potwierdzającej. Cały proces może wiązać się z licznymi żądaniami, w tym przekierowaniami, przy czym parametry przesłane na wcześniejszych etapach są później retransmitowane przez stronę klienta. Każdy etap tego procesu należy przetestować indywidualnie, aby potwierdzić, czy kontrole dostępu są stosowane prawidłowo.

KROKI HACKOWANIA

1. Gdy działanie jest wykonywane w sposób wieloetapowy i obejmuje kilka różnych żądań od klienta do serwera, przetestuj każde żądanie indywidualnie, aby ustalić, czy zastosowano do niego kontrolę dostępu. Pamiętaj, aby uwzględnić każde żądanie, w tym przesłane formularze, następujące przekierowania i wszelkie żądania niesparametryzowane.

2. Spróbuj znaleźć miejsca, w których aplikacja skutecznie zakłada, że jeśli dotarłeś do określonego punktu, musiałeś dotrzeć w legalny sposób. Spróbuj dotrzeć do tego punktu w inny sposób, korzystając z konta o niższych uprawnieniach, aby wykryć, czy możliwe są ataki polegające na eskalacji uprawnień.

3. Jednym ze sposobów ręcznego przeprowadzenia tego testu jest kilkakrotne przejście przez chroniony proces wieloetapowy w przeglądarce i użycie serwera proxy do zamiany tokena sesji dostarczanego w różnych żądaniach na token mniej uprzywilejowanego użytkownika.

4. Często możesz radykalnie przyspieszyć ten proces, korzystając z funkcji „żądanie w przeglądarce” pakietu Burp Suite:

A. Użyj konta o wyższych uprawnieniach, aby przejść przez cały wieloetapowy proces.

B. Zaloguj się do aplikacji przy użyciu konta o niższych uprawnieniach (lub żadnego).

C. W historii Burp Proxy znajdź sekwencję żądań, które zostały wykonane, gdy proces wieloetapowy był wykonywany jako bardziej uprzywilejowany użytkownik. Dla każdego żądania w sekwencji wybierz pozycję menu kontekstowego „żądanie w przeglądarce w bieżącej sesji przeglądarki". Wklej podany adres URL do przeglądarki, która jest zalogowana jako użytkownik o niższych uprawnieniach.

D. Jeśli aplikacja Ci na to pozwala, wykonaj resztę wieloetapowego procesu w normalny sposób, korzystając z przeglądarki.

E. Wyświetl wynik zarówno w przeglądarce, jak iw historii serwera proxy, aby określić, czy pomyślnie wykonał akcję uprzywilejowaną.

Po wybraniu funkcji Burp „żądanie w przeglądarce w bieżącej sesji przeglądarki" dla określonego żądania, Burp podaje unikalny adres URL kierujący na wewnętrzny serwer sieciowy Burp, który wklejasz w pasku adresu przeglądarki. Gdy przeglądarka żąda tego adresu URL, Burp zwraca przekierowanie do pierwotnie określonego adresu URL. Gdy Twoja przeglądarka podąża za przekierowaniem, Burp zastępuje żądanie tym, które pierwotnie określiłeś, pozostawiając nienaruszony nagłówek Cookie. Jeśli testujesz różne konteksty użytkownika, możesz przyspieszyć ten proces. Zaloguj się do kilku różnych przeglądarek jako różni użytkownicy i wklej adres URL do każdej przeglądarki, aby zobaczyć, jak obsługiwane jest żądanie użytkownika, który jest zalogowany przy użyciu tej przeglądarki. (Pamiętaj, że ponieważ pliki cookie są zazwyczaj współużytkowane przez różne okna tej samej przeglądarki, zwykle do przeprowadzenia tego testu konieczne będzie użycie różnych przeglądarek lub przeglądarek na różnych komputerach). WSKAZÓWKA: Podczas testowania procesów wieloetapowych w różnych kontekstach użytkownika , czasami warto przejrzeć sekwencje próśb wysyłanych przez różnych użytkowników obok siebie, aby zidentyfikować subtelne różnice, które mogą zasługiwać na dalsze zbadanie. Jeśli korzystasz z oddzielnych przeglądarek, aby uzyskać dostęp do aplikacji jako różni użytkownicy, możesz utworzyć różne odbiorniki proxy w Burp do użytku przez każdą przeglądarkę (musisz zaktualizować konfigurację proxy w każdej przeglądarce, aby wskazywała odpowiedni odbiornik). Następnie dla każdej przeglądarki użyj menu kontekstowego w historii proxy, aby otworzyć nowe okno historii i ustaw filtr wyświetlania, aby wyświetlał tylko żądania od odpowiedniego odbiornika proxy.

Testowanie z ograniczonym dostępem

Jeśli masz tylko jedno konto na poziomie użytkownika, za pomocą którego możesz uzyskać dostęp do aplikacji (lub nie masz go wcale), należy wykonać dodatkową pracę, aby przetestować skuteczność kontroli dostępu. W rzeczywistości, aby przeprowadzić w pełni kompleksowy test, w każdym przypadku należy wykonać dalsze prace. Mogą istnieć słabo chronione funkcje, które nie są jawnie połączone z interfejsem żadnego użytkownika aplikacji. Na przykład być może stara funkcjonalność nie została jeszcze usunięta lub nowa funkcjonalność została wdrożona, ale nie została jeszcze opublikowana dla użytkowników.

KROKI HACKOWANIA

1. Użyj technik wykrywania treści opisanych w rozdziale 4, aby zidentyfikować jak najwięcej funkcji aplikacji. Wykonanie tego ćwiczenia jako użytkownik o niskich uprawnieniach jest często wystarczające zarówno do wyliczenia, jak i uzyskania bezpośredniego dostępu do poufnych funkcji.

2. W przypadku zidentyfikowania stron aplikacji, które mogą przedstawiać różne funkcje lub łącza zwykłym użytkownikom i użytkownikom administracyjnym (na przykład Panel sterowania lub Moja strona główna), spróbuj dodać parametry, takie jak admin=true, do ciągu zapytania adresu URL i treści Żądania POST. Pomoże Ci to określić, czy odkryje lub zapewni dostęp do dodatkowych funkcji, do których kontekst użytkownika nie ma normalnego dostępu.

3. Sprawdź, czy aplikacja używa nagłówka Referer jako podstawy do podejmowania decyzji dotyczących kontroli dostępu. W przypadku kluczowych funkcji aplikacji, do których masz uprawnienia dostępu, spróbuj usunąć lub zmodyfikować nagłówek Referer i ustalić, czy żądanie nadal jest skuteczne. Jeśli nie, aplikacja może ufać nagłówkowi strony odsyłającej w niebezpieczny sposób. Jeśli skanujesz żądania za pomocą aktywnego skanera Burp, Burp próbuje usunąć nagłówek Referer z każdego żądania i informuje, czy wydaje się, że ma to systematyczny i istotny wpływ na odpowiedź aplikacji.

4. Przejrzyj wszystkie kody HTML i skrypty po stronie klienta, aby znaleźć odniesienia do ukrytych funkcji lub funkcji, którymi można manipulować po stronie klienta, takich jak interfejsy użytkownika oparte na skryptach. Ponadto zdekompiluj wszystkie komponenty rozszerzenia przeglądarki zgodnie z opisem w części 5, aby wykryć wszelkie odniesienia do funkcji po stronie serwera.

Po wyliczeniu wszystkich dostępnych funkcji należy sprawdzić, czy segregacja dostępu do zasobów dla poszczególnych użytkowników jest prawidłowo wymuszana. W każdym przypadku, gdy aplikacja zapewnia użytkownikom dostęp do podzbioru szerszego zakresu zasobów tego samego typu (takich jak dokumenty, zamówienia, e-maile i dane osobowe), może zaistnieć możliwość uzyskania przez jednego użytkownika nieautoryzowanego dostępu do innych zasobów.

KROKI HACKOWANIA

1. Jeśli aplikacja używa wszelkiego rodzaju identyfikatorów (identyfikatorów dokumentów, numerów kont, numerów zamówień) w celu określenia, którego zasobu żąda użytkownik, spróbuj wykryć identyfikatory zasobów, do których nie masz autoryzowanego dostępu.

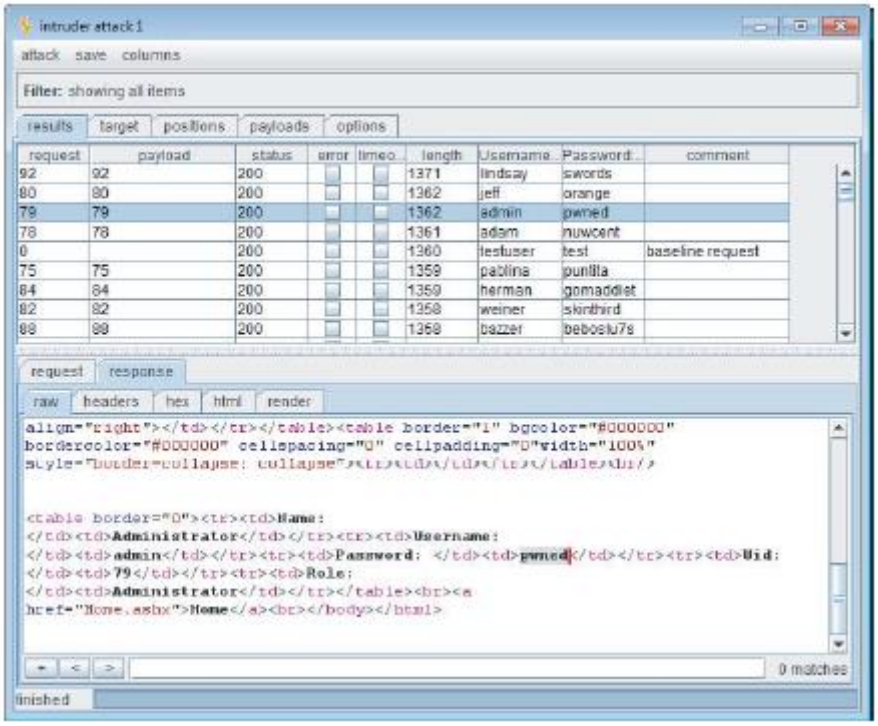
2. Jeżeli możliwe jest wygenerowanie serii takich identyfikatorów w krótkich odstępach czasu (na przykład poprzez utworzenie wielu nowych dokumentów lub zleceń), należy zastosować techniki dla tokenów sesyjnych, aby spróbować wykryć wszelkie przewidywalne sekwencje w identyfikatorach, które wytwarza aplikacja.

3. Jeśli nie jest możliwe wygenerowanie nowych identyfikatorów, możesz ograniczyć się do analizy identyfikatorów, które już odkryłeś, lub nawet użyć zwykłego zgadywania. Jeśli identyfikator ma postać GUID, jest mało prawdopodobne, że jakiekolwiek próby oparte na zgadywaniu zakończą się sukcesem. Jeśli jednak jest to stosunkowo niewielka liczba, wypróbuj inne liczby z bliskiej odległości lub liczby losowe o tej samej liczbie cyfr.

4. Jeśli okaże się, że kontrola dostępu jest złamana, a identyfikatory zasobów są przewidywalne, można przeprowadzić zautomatyzowany atak w celu zebrania wrażliwych zasobów i informacji z aplikacji. Użyj technik opisanych w rozdziale 14, aby zaprojektować zautomatyzowany atak na zamówienie w celu odzyskania potrzebnych danych.

Katastrofalna luka w zabezpieczeniach tego rodzaju występuje, gdy strona Informacje o koncie wyświetla dane osobowe użytkownika wraz z jego nazwą użytkownika i hasłem. Chociaż hasło jest zwykle maskowane na ekranie, to jednak jest przesyłane w całości do przeglądarki. W tym miejscu często można szybko przejrzeć pełen zakres identyfikatorów kont, aby zebrać dane logowania

wszystkich użytkowników, w tym administratorów. Rysunek przedstawia użycie Burp Intruder do przeprowadzenia udanego ataku tego rodzaju.



WSKAZÓWKA: po wykryciu luki w zabezpieczeniach kontroli dostępu natychmiastowym atakiem jest próba dalszego zwiększenia uprawnień przez przejęcie konta użytkownika z uprawnieniami administratora. Możesz użyć różnych sztuczek, aby zlokalizować konto administracyjne. Korzystając z luki w kontroli dostępu, takiej jak ta zilustrowana, możesz zebrać setki danych uwierzytelniających użytkowników i nie cieszyć się ręcznym logowaniem jak każdy użytkownik, dopóki nie znajdziesz administratora. Jednak gdy konta są identyfikowane za pomocą identyfikatora numerycznego, często zdarza się, że administratorzy mają przypisane najniższe numery kont. Zalogowanie się jako kilku pierwszych użytkowników zarejestrowanych w aplikacji często identyfikuje administratora. Jeśli to podejście zawiedzie, skuteczną metodą jest znalezienie w aplikacji funkcji, w której dostęp jest odpowiednio segregowany poziomo, takiej jak główna strona główna prezentowana każdemu użytkownikowi. Napisz skrypt, aby zalogować się przy użyciu każdego zestawu przechwyconych poświadczeń, a następnie spróbuj uzyskać dostęp do własnej strony głównej. Jest prawdopodobne, że użytkownicy administracyjni mogą przeglądać stronę główną każdego użytkownika, więc natychmiast wykryjesz, kiedy używane jest konto administracyjne.

Testowanie bezpośredniego dostępu do metod

Gdy aplikacja korzysta z żądań, które dają bezpośredni dostęp do metod API po stronie serwera, wszelkie słabe punkty kontroli dostępu w tych metodach są zwykle identyfikowane przy użyciu opisanej już metodologii. Należy jednak również sprawdzić, czy istnieją dodatkowe interfejsy API, które mogą nie być odpowiednio chronione. Na przykład aplet można wywołać za pomocą następującego żądania:

POST /svc HTTP/1.1

Accept-Encoding: gzip, deflate

Host: wahh-app

Content-Length: 37

servlet=com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug

Ponieważ jest to dobrze znany serwlet, być może możesz uzyskać dostęp do innych serwletów w celu wykonania nieautoryzowanych działań.

KROKI HACKOWANIA

1. Zidentyfikuj parametry, które są zgodne z konwencjami nazewnictwa języka Java (np. get, set, add, update, is lub has, po których następuje słowo pisane wielką literą) lub jawnie określ strukturę pakietu (np. com.companyname .xxx.yyy .Nazwa klasy). Zanotuj wszystkie metody, do których się odwołujesz, jakie możesz znaleźć.

2. Poszukaj metody, która zawiera listę dostępnych interfejsów lub metod. Sprawdź historię swojego serwera proxy, aby zobaczyć, czy został on wywołany w ramach normalnej komunikacji aplikacji. Jeśli nie, spróbuj odgadnąć, korzystając z obserwowanej konwencji nazewnictwa.

3. Skonsultuj się z zasobami publicznymi, takimi jak wyszukiwarki i fora internetowe, aby określić inne metody, które mogą być dostępne.

4. Użyj technik opisanych w rozdziale 4, aby odgadnąć inne nazwy metod.

5. Próba uzyskania dostępu do wszystkich metod zebranych przy użyciu różnych typów kont użytkowników, w tym dostępu nieuwierzytelnionego.

6. Jeśli nie znasz liczby lub typów argumentów oczekiwanych przez niektóre metody, poszukaj metod, które z mniejszym prawdopodobieństwem przyjmują argumenty, takich jak listInterfaces i getAllUsersInRoles.

Testowanie kontroli zasobów statycznych

W przypadkach, gdy zasoby statyczne, które chroni aplikacja, są ostatecznie dostępne bezpośrednio za pośrednictwem adresów URL do samych plików zasobów, należy sprawdzić, czy nieupoważnieni użytkownicy mogą po prostu bezpośrednio zażądać tych adresów URL.

KROKI HACKOWNIA

1. Przejdź przez normalny proces uzyskiwania dostępu do chronionego zasobu statycznego, aby uzyskać przykład adresu URL, za pomocą którego jest on ostatecznie pobierany.

2. Używając innego kontekstu użytkownika (na przykład mniej uprzywilejowanego użytkownika lub konta, które nie dokonało wymaganego zakupu), spróbuj uzyskać dostęp do zasobu bezpośrednio za pomocą wskazanego adresu URL.

3. Jeśli ten atak się powiedzie, spróbuj zrozumieć schemat nazewnictwa używany dla chronionych plików statycznych. Jeśli to możliwe, skonstruuj zautomatyzowany atak w celu wyszukania treści, które mogą być przydatne lub mogą zawierać poufne dane

Ograniczenia testowania metod HTTP

Chociaż może nie być gotowego sposobu na wykrycie, czy kontrola dostępu aplikacji wykorzystuje kontrole na poziomie platformy w stosunku do metod HTTP, możesz wykonać kilka prostych kroków, aby zidentyfikować wszelkie luki w zabezpieczeniach.

KROKI HACKOWANIA

1. Korzystając z konta z wysokimi uprawnieniami, zidentyfikuj niektóre uprzywilejowane żądania, które wykonują poufne działania, takie jak dodanie nowego użytkownika lub zmiana roli zabezpieczeń użytkownika.

2. Jeśli te żądania nie są chronione żadnymi tokenami anty-CSRF lub podobnymi funkcjami, użyj konta z wysokimi uprawnieniami, aby ustalić, czy aplikacja nadal wykonuje żądaną akcję, jeśli metoda HTTP zostanie zmodyfikowana. Przetestuj następujące metody HTTP:

* POST

* GET

* HEAD

* Dowolna nieprawidłowa metoda HTTP

3. Jeśli aplikacja honoruje żądania przy użyciu metod HTTP innych niż metoda oryginalna, przetestuj kontrolę dostępu do tych żądań, korzystając z opisanej już standardowej metodologii, używając kont z niższymi uprawnieniami.

Zabezpieczanie kontroli dostępu

Kontrola dostępu to jeden z najłatwiejszych do zrozumienia obszarów bezpieczeństwa aplikacji internetowych, chociaż podczas ich wdrażania należy ostrożnie stosować dobrze poinformowaną i dogłębną metodologię. Po pierwsze, należy unikać kilku oczywistych pułapek. Wynikają one zwykle z nieznajomości podstawowych wymagań skutecznej kontroli dostępu lub błędne założenia dotyczące rodzajów żądań, które użytkownicy będą składać i przed którymi aplikacja musi się bronić:

* Nie polegaj na nieznajomości adresów URL aplikacji lub identyfikatorów używanych do określania zasobów aplikacji, takich jak numery kont i identyfikatory dokumentów. Załóż, że użytkownicy znają każdy adres URL i identyfikator aplikacji oraz upewnij się, że same mechanizmy kontroli dostępu do aplikacji są wystarczające, aby zapobiec nieautoryzowanemu dostępowi.

* Nie ufaj żadnym przesłanym przez użytkownika parametrom oznaczającym prawa dostępu (takim jak admin=true).

* Nie zakładaj, że użytkownicy będą uzyskiwać dostęp do stron aplikacji w zamierzonej kolejności. Nie zakładaj, że ponieważ użytkownicy nie mogą uzyskać dostępu do strony Edytuj użytkowników, nie mogą przejść do strony Edytuj użytkownika X, do której prowadzi łącze.

* Nie ufaj użytkownikowi, że nie będzie manipulował żadnymi danymi przesyłanymi przez klienta. Jeśli niektóre dane przesłane przez użytkownika zostały zweryfikowane, a następnie przesłane przez klienta, nie należy polegać na retransmitowanej wartości bez ponownej walidacji.

Poniżej przedstawiono najlepsze praktyki wdrażania skutecznych kontroli dostępu w aplikacjach internetowych:

* Wyraźnie oceniaj i dokumentuj wymagania dotyczące kontroli dostępu dla każdej jednostki funkcjonalności aplikacji. Musi to obejmować zarówno to, kto może legalnie korzystać z funkcji, jak i zasoby, do których poszczególni użytkownicy mogą uzyskiwać dostęp za pośrednictwem tej funkcji.

* Kieruj wszystkimi decyzjami dotyczącymi kontroli dostępu z poziomu sesji użytkownika.

* Użyj centralnego komponentu aplikacji do sprawdzania kontroli dostępu.

* Przetwarzaj każde żądanie klienta za pośrednictwem tego komponentu, aby sprawdzić, czy użytkownik składający żądanie ma dostęp do żądanych funkcji i zasobów.

* Użyj technik programistycznych, aby upewnić się, że nie ma wyjątków od poprzedniego punktu. Skutecznym podejściem jest nakazanie, aby każda strona aplikacji zawierała interfejs, do którego wysyłane są zapytania z centralnego mechanizmu kontroli dostępu. Jeśli zmusisz programistów do jawnego kodowania logiki kontroli dostępu na każdej stronie, nie ma usprawiedliwienia dla pominięć.

* W przypadku szczególnie wrażliwych funkcji, takich jak strony administracyjne, można dodatkowo ograniczyć dostęp za pomocą adresu IP, aby mieć pewność, że dostęp do funkcji będą mieli tylko użytkownicy z określonego zakresu sieci, niezależnie od ich statusu logowania.

* Jeśli zawartość statyczna musi być chroniona, istnieją dwie metody zapewnienia kontroli dostępu. Po pierwsze, dostęp do plików statycznych można uzyskać pośrednio, przekazując nazwę pliku do dynamicznej strony po stronie serwera, która implementuje odpowiednią logikę kontroli dostępu. Po drugie, bezpośredni dostęp do plików statycznych można kontrolować za pomocą uwierzytelniania HTTP lub innych funkcji serwera aplikacji w celu zawinięcia przychodzącego żądania i sprawdzenia uprawnień zasobu przed przyznaniem dostępu.

* Identyfikatory określające, do którego zasobu użytkownik chce uzyskać dostęp, są podatne na manipulacje za każdym razem, gdy są przesyłane przez klienta. Serwer powinien ufać tylko integralności danych po stronie serwera. Za każdym razem, gdy te identyfikatory są przesyłane przez klienta, należy je ponownie zweryfikować, aby upewnić się, że użytkownik ma uprawnienia dostępu do żądanego zasobu.

* W przypadku funkcji aplikacji o znaczeniu krytycznym dla bezpieczeństwa, takich jak tworzenie nowego odbiorcy rachunku w aplikacji bankowej, należy rozważyć wdrożenie ponownego uwierzytelniania transakcji i podwójnej autoryzacji, aby zapewnić dodatkową pewność, że funkcja nie jest używana przez nieupoważnioną osobę. Łagodzi to również konsekwencje innych możliwych ataków, takich jak przejmowanie sesji.

* Rejestruj każde zdarzenie, w którym uzyskano dostęp do poufnych danych lub wykonano poufną akcję. Te dzienniki umożliwią wykrywanie i badanie potencjalnych naruszeń kontroli dostępu.

Twórcy aplikacji internetowych często wdrażają funkcje kontroli dostępu fragmentarycznie. Dodają kod do poszczególnych stron w przypadkach, gdy wymagana jest pewna kontrola dostępu, i często wycinają i wklejają ten sam kod między stronami, aby wdrożyć podobne wymagania. Takie podejście niesie ze sobą nieodłączne ryzyko defektów w wynikowym mechanizmie kontroli dostępu. Wiele przypadków jest pomijanych, gdy wymagane są kontrole, kontrole zaprojektowane dla jednego obszaru mogą nie działać w zamierzony sposób w innym obszarze, a modyfikacje dokonane w innym miejscu aplikacji mogą zepsuć istniejące kontrole, naruszając przyjęte przez nie założenia. W przeciwieństwie do tego podejścia, wcześniej opisana metoda wykorzystania centralnego komponentu aplikacji do egzekwowania kontroli dostępu ma wiele zalet:

* Zwiększa przejrzystość kontroli dostępu w aplikacji, umożliwiając różnym programistom szybkie zrozumienie kontroli zaimplementowanych przez innych.

* Sprawia, że konserwacja jest bardziej wydajna i niezawodna. Większość zmian należy zastosować tylko raz, do pojedynczego współdzielonego komponentu i nie trzeba ich wycinać i wklejać w wielu miejscach.

* Poprawia zdolności adaptacyjne. Tam, gdzie pojawiają się nowe wymagania dotyczące kontroli dostępu, można je łatwo odzwierciedlić w istniejącym interfejsie API zaimplementowanym na każdej stronie aplikacji.

* Skutkuje to mniejszą liczbą błędów i pominięć, niż gdyby kod kontroli dostępu był wdrażany fragmentarycznie w całej aplikacji.

Wielowarstwowy model uprawnień

Kwestie związane z dostępem dotyczą nie tylko samej aplikacji internetowej, ale także innych warstw infrastruktury, które leżą poniżej — w szczególności serwera aplikacji, bazy danych i systemu operacyjnego. Podejście do bezpieczeństwa obejmujące kompleksową ochronę wymaga wdrożenia kontroli dostępu na każdej z tych warstw w celu stworzenia kilku warstw ochrony. Zapewnia to większą pewność przed zagrożeniem nieautoryzowanym dostępem, ponieważ jeśli atakującemu uda się złamać zabezpieczenia w jednej warstwie, atak może zostać zablokowany przez zabezpieczenia w innej warstwie. Oprócz implementacji skutecznych kontroli dostępu w samej aplikacji internetowej, jak już opisano, wielowarstwowe podejście można zastosować na różne sposoby do komponentów leżących u podstaw aplikacji:

* Serwer aplikacji może służyć do kontrolowania dostępu do całych ścieżek URL na podstawie ról użytkowników zdefiniowanych na poziomie serwera aplikacji.

* Aplikacja może korzystać z innego konta bazy danych podczas wykonywania działań różnych użytkowników. W przypadku użytkowników, którzy powinni tylko wyszukiwać dane (nie aktualizować ich), należy użyć konta z uprawnieniami tylko do odczytu.

* Precyzyjną kontrolę nad dostępem do różnych tabel bazy danych można zaimplementować w samej bazie danych, korzystając z tabeli uprawnień

* Konta systemu operacyjnego używane do uruchamiania każdego komponentu w infrastrukturze mogą być ograniczone do najniższych uprawnień, jakich dany komponent faktycznie wymaga.

W złożonych aplikacjach o krytycznym znaczeniu dla bezpieczeństwa tego rodzaju warstwowe zabezpieczenia można opracować za pomocą macierzy definiującej różne role użytkowników w aplikacji i różne uprawnienia na każdym poziomie, które należy przypisać każdej roli. Rysunek przedstawia częściowy przykład macierzy uprawnień dla złożonej aplikacji.

	Application Server	Application Roles	Database Privileges													
User type	URL path	User role	Search	Create Application	Edit Application	Purge Application	View Applications	Policy Updates	Rate Adjustment	View User Accounts	Create Users	View Company Ac	Edit Company Ac	Create Company	View Audit Log	Delegate privilege
Administrator	/*	Site Administrator	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Support	✓		✓		✓	✓		✓	✓	✓	✓	✓		
Site Supervisor	/admin/* /myQuotes/* /help/*	Back Office – New business		✓			✓									
		Back Office – Referrals		✓	✓	✓		✓	✓							
		Back Office – Helpdesk	✓				✓			✓		✓			✓	✓
Company Administrator	/myQuotes/* /help/*	Customer – Administrator		✓	✓	✓	✓				✓	✓	✓			✓
		Customer – New Business		✓		✓	✓									
		Customer – Support	✓				✓			✓						
Normal User	/myQuotes/dash.jsp /myQuotes/apply.jsp /myQuotes/search.jsp /help/*	User – Applications	✓	✓			✓									
		User – Referrals														
		User – Helpdesk														
		Unregistered (Read Only)	✓				✓									
Audit	(none)	Syslog Server Account													✓	

W ramach tego rodzaju modelu bezpieczeństwa można zobaczyć, jak można zastosować różne przydatne koncepcje kontroli dostępu:

* Kontrola programowa — macierz poszczególnych uprawnień do bazy danych jest przechowywana w tabeli w bazie danych i jest stosowana programowo w celu egzekwowania decyzji dotyczących kontroli dostępu. Klasyfikacja ról użytkowników zapewnia skrót do stosowania pewnych kontroli kontroli dostępu, co jest również stosowane programowo. Kontrolki programistyczne mogą być niezwykle szczegółowe i mogą wbudowywać dowolnie złożoną logikę w proces podejmowania decyzji dotyczących kontroli dostępu w aplikacji.

* Uznaniowa kontrola dostępu (DAC) — administratorzy mogą delegować swoje uprawnienia innym użytkownikom w odniesieniu do określonych posiadanych zasobów, stosując uznaniową kontrolę dostępu. Jest to zamknięty model DAC, w którym dostęp jest zabroniony, chyba że zostanie wyraźnie przyznany. Administratorzy mogą również blokować lub wygasać indywidualne konta użytkowników. Jest to otwarty model DAC, w którym dostęp jest dozwolony, o ile nie zostanie wyraźnie cofnięty. Różni użytkownicy aplikacji mają uprawnienia do tworzenia kont użytkowników, ponownie stosując uznaniową kontrolę dostępu.

* Kontrola dostępu oparta na rolach (RBAC) — nazwane role zawierają różne zestawy określonych uprawnień, a każdy użytkownik jest przypisany do jednej z tych ról. Służy to jako skrót do przypisywania i wymuszania różnych uprawnień i jest niezbędny do zarządzania kontrolą dostępu w złożonych aplikacjach. Używanie ról do przeprowadzania kontroli dostępu z góry na żądania użytkowników umożliwia szybkie odrzucanie wielu nieautoryzowanych żądań przy minimalnej ilości przetwarzania. Przykładem takiego podejścia jest ochrona ścieżek URL, do których mogą uzyskiwać dostęp określone typy użytkowników. Projektując mechanizmy kontroli dostępu oparte na rolach, należy zrównoważyć liczbę ról, tak aby pozostały one użytecznym narzędziem pomagającym zarządzać uprawnieniami w aplikacji. Jeśli zostanie utworzonych zbyt wiele szczegółowych ról, liczba różnych ról staje się nieporęczna i trudno jest nimi dokładnie zarządzać. Jeśli zostanie utworzonych zbyt mało ról, powstałe role będą zgrubnym narzędziem do zarządzania dostępem. Jest prawdopodobne, że poszczególni użytkownicy otrzymają uprawnienia, które nie są bezwzględnie niezbędne do wykonywania ich funkcji. Jeśli kontrole na poziomie platformy są używane do ograniczania dostępu do różnych ról aplikacji na

podstawie metody HTTP i adresu URL, należy je zaprojektować przy użyciu modelu domyślnej odmowy, zgodnie z najlepszą praktyką w przypadku reguł zapory. Powinno to obejmować różne określone reguły, które przypisują określone metody HTTP i adresy URL do określonych ról, a ostateczna reguła powinna odrzucać każde żądanie, które nie pasuje do poprzedniej reguły.

* Kontrola deklaratywna - aplikacja używa ograniczonych kont bazy danych podczas uzyskiwania dostępu do bazy danych. Wykorzystuje różne konta dla różnych grup użytkowników, przy czym każde konto ma najniższy poziom uprawnień niezbędny do wykonywania działań, które grupa może wykonywać. Kontrolki deklaratywne tego rodzaju są deklarowane spoza aplikacji. Jest to przydatne zastosowanie zasad ochrony w głąb, ponieważ uprawnienia są nakładane na aplikację przez inny komponent. Nawet jeśli użytkownik znajdzie sposób na naruszenie kontroli dostępu zaimplementowanych w warstwie aplikacji w celu wykonania poufnej czynności, takiej jak dodanie nowego użytkownika, nie może tego zrobić. Konto bazy danych, którego używa, nie ma wymaganych uprawnień w bazie danych. Inne sposoby stosowania deklaratywnej kontroli dostępu istnieją na poziomie serwera aplikacji, poprzez pliki deskryptorów wdrażania, które są stosowane podczas wdrażania aplikacji. Mogą to być jednak stosunkowo proste instrumenty i nie zawsze dobrze skalują się, aby zarządzać precyzyjnymi uprawnieniami w dużej aplikacji.

KROKI HACKOWANIA

Jeśli atakujesz aplikację, która wykorzystuje wielopoziomowy model uprawnień tego rodzaju, prawdopodobnie uda się obronić przed wieloma najbardziej oczywistymi błędami, które są często popełniane podczas stosowania kontroli dostępu. Może się okazać, że obejście kontroli zaimplementowanych w aplikacji nie zaprowadzi Cię zbyt daleko ze względu na ochronę na innych warstwach. Mając to na uwadze, nadal masz do dyspozycji kilka potencjalnych linii ataku. Co najważniejsze, zrozumienie ograniczeń każdego rodzaju kontroli pod względem ochrony, której nie oferuje, pomoże zidentyfikować luki w zabezpieczeniach, które najprawdopodobniej będą miały na nią wpływ:

* Kontrole programistyczne w warstwie aplikacji mogą być podatne na ataki oparte na wstrzykiwaniu.

* Role zdefiniowane w warstwie serwera aplikacji są często zdefiniowane zgrubnie i mogą być niekompletne.

* Tam, gdzie komponenty aplikacji działają przy użyciu kont systemu operacyjnego o niskich uprawnieniach, zazwyczaj mogą odczytywać wiele rodzajów potencjalnie wrażliwych danych w systemie plików hosta. Wszelkie luki w zabezpieczeniach umożliwiające dowolny dostęp do plików mogą nadal być użytecznie wykorzystywane, nawet jeśli tylko w celu odczytania wrażliwych danych.

* Zazwyczaj luki w samym oprogramowaniu serwera aplikacji

pozwalają obejść wszystkie kontrole dostępu zaimplementowane w warstwie aplikacji, ale nadal możesz mieć ograniczony dostęp do bazy danych i systemu operacyjnego.

* Pojedyncza luka w zabezpieczeniach kontroli dostępu, którą można wykorzystać w odpowiedniej lokalizacji, może nadal stanowić punkt wyjścia do poważnej eskalacji uprawnień. Na przykład, jeśli odkryjesz sposób modyfikowania roli powiązanej z Twoim kontem, może się okazać, że ponowne zalogowanie się na to konto zapewni Ci lepszy dostęp zarówno w warstwie aplikacji, jak i bazy danych.

Streszczenie

Wady kontroli dostępu mogą objawiać się na różne sposoby. W niektórych przypadkach mogą być nieciekawe, umożliwiając nielegalny dostęp do nieszkodliwej funkcji, której nie można wykorzystać do

dalszej eskalacji uprawnień. W innych przypadkach znalezienie słabego punktu w kontroli dostępu może szybko doprowadzić do całkowitego skompromitowania aplikacji. Błędy w kontroli dostępu mogą wynikać z różnych źródeł. Zły projekt aplikacji może utrudnić lub uniemożliwić sprawdzenie, czy nie ma do niej dostępu nieuprawniony użytkownik, zwykłe przeoczenie może pozostawić bez ochrony tylko jedną lub dwie funkcje, a błędne założenia dotyczące zachowania użytkowników mogą pozostawić aplikację bez obrony, gdy te założenia zostaną naruszone. W wielu przypadkach znalezienie przerwy w kontroli dostępu jest niemal trywialne. Wystarczy poprosić o wspólny administracyjny adres URL i uzyskać bezpośredni dostęp do funkcji. W innych przypadkach może to być bardzo trudne, a subtelne defekty mogą czaić się głęboko w logice aplikacji, szczególnie w złożonych aplikacjach o wysokim poziomie bezpieczeństwa. Najważniejszą lekcją podczas atakowania kontroli dostępu jest szukanie wszędzie. Jeśli masz trudności z robieniem postępów, bądź cierpliwy i testuj każdy krok każdej funkcji aplikacji. Błąd, który pozwala na posiadanie całej aplikacji, może być tuż za rogiem.

Pytania

1. Aplikacja może używać nagłówka HTTP Referer do kontrolowania dostępu bez wyraźnego wskazania tego w swoim normalnym zachowaniu. Jak możesz sprawdzić tę słabość?

2. Logujesz się do aplikacji i zostajesz przekierowany do następującego adresu URL: https://wahh-app.com/MyAccount.php?uid=1241126841

Wygląda na to, że aplikacja przekazuje identyfikator użytkownika do strony MyAccount.php. Jedynym znanym Ci identyfikatorem jest Twój własny. Jak sprawdzić, czy aplikacja używa tego parametru do egzekwowania kontroli dostępu w niebezpieczny sposób?

3. Aplikacja internetowa w Internecie wymusza kontrolę dostępu, sprawdzając źródłowe adresy IP użytkowników. Dlaczego to zachowanie jest potencjalnie wadliwe?

4. Jedynym celem aplikacji jest zapewnienie przeszukiwalnego repozytorium informacji do użytku przez członków społeczeństwa. Nie ma mechanizmów uwierzytelniania ani obsługi sesji. Jakie kontrole dostępu należy zaimplementować w aplikacji?

5. Podczas przeglądania aplikacji natkniesz się na kilka wrażliwych zasobów, które należy chronić przed nieautoryzowanym dostępem i które mają rozszerzenie pliku .xls. Dlaczego od razu powinny przykuć twoją uwagę?