Cloud Penetration Testing Playbook

cloud security alliance®

The permanent and official location for Cloud Security Alliance Top Threats research is https://cloudsecurityalliance.org/working-groups/top-threats/#_overview

ACKNOWLEDGEMENTS

Lead Authors

Alexander Getsin

Contributors

Asaf Hecht
Michael Roza
Jon-Michael Brook
Shlomi Ohayon
Chris Farris
Greg Jensen
Victor Chin

CSA Global Staff

Victor Chin

Special Acknowledgement

The CSA Top Threats Working Group would like to thank CyberInt for their support in the development of this document.

TABLE OF CONTENTS

INTRODUCTION

Security testing is crucial to the security assurance of cloud environments, systems and services. In this document, we discuss the most predominant form of security testing in relation to cloud environments – penetration testing.

Penetration testing, as defined by NIST, is a specialized type of technical assessment conducted on information systems or individual system components to identify vulnerabilities that could be exploited by adversaries. Such testing can be used to either identify vulnerabilities or determine the degree of resistance organizational information systems have to adversaries within a set of specified constraints (e.g., time, resources, and/or skills)[1]. ENISA's definition is conceptually similar to NIST's[2].

Traditionally, the primary objective of penetration testing is to identify technical security weaknesses and systems resilience. However, a broader application of security testing also serves to assess an organization's implementation of security policy, compliance requirements, the effectiveness of employees' security awareness and ability to identify and respond to security incidents[3]. Hence, penetration testing is essential to any holistic cyber defense effort as it provides visibility into system security, its assurance (or lack thereof), and produces highly actionable mitigations to drive the security of the systems and environments involved.

As cloud services continue to enable new technologies, see massive adoption and become foundational for many businesses, there is a need to extend the scope of penetration testing into public cloud systems and components.

The process described here aims to provide the foundation for a public cloud penetration testing methodology and is designed for current and future technologies that are hosted on public cloud environments or services. In particular, this document focuses on penetration testing of applications and services hosted in the cloud. It addresses the methodological and knowledge gaps in security testing of information systems and applications in public cloud environments.

Target Audience

The target audience of this document are penetration testers and cloud / cloud-based systems security practitioners. However, the first few pages will provide CIOs, CISOs and Senior Management an understanding of what cloud penetration testing is, its scope, its context, its objectives and how it fits within a cybersecurity strategy. Developers and Architects will also find this document useful while designing secure (public cloud based) systems.

This document aims to:

- Raise awareness of the importance and methods of cloud penetration testing in a cybersecurity strategy

[1] https://nvd.nist.gov/800-53/Rev4/control/CA-8#Rev4Statements

[2] https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/vulnerabilities-and-exploits

[3] https://www.fedramp.gov/assets/resources/documents/CSP_Penetration_Test_Guidance.pdf

- Educate readers on the principles and considerations of cloud penetration testing in security assurance
- Provide guidance to penetration testers so as to deliver better informed & comprehensive security testing in public cloud environments

SCOPE OF THIS DOCUMENT

This work focuses on testing systems and services hosted in public cloud environments. This refers to customer-controlled or customer-managed systems and services. For example, a custom virtual machine, managed and controlled by the cloud customer, in an IaaS environment would be in-scope whereas the hypervisor of an IaaS environment that is controlled by the cloud service provider isn't. As for testing hybrid clouds, this document does not cover the hybrid interface and on-premises environment.

This methodology is COMPLEMENTARY to

Subject matter scope – guidance on how to test cloud implementation of applications and systems is provided, but is not provided on testing the application itself. That's what OWASP (Open Web Application Security Project) would be for, for example.

Existing testing and assurance frameworks – while testing program and delivery phases, as well as some test cases not unique to the cloud, are outlined within, this is done purely for context and due diligence and are NOT comprehensive. The cloud-unique test cases and considerations are meant to complement existing frameworks by allowing ease of adoption and integration.

This resource also provides reflection and insights on scoping and legal aspects of public cloud security testing and training opportunities and resources.

CLOUD PENETRATION TESTING SCOPE

Security testing of cloud based systems, environments and services is nuanced and unique to the public cloud.

Testing Scope per the Shared Responsibility Model

Security controls that fall under the complete responsibility of the Cloud Service Provider (CSP) are usually not within the scope of a penetration test that is commissioned by the cloud consumer. For example, in a Software as a Service (SaaS) environment, it is within the penetration tester's scope of authority to exploit excessive permissions given to a particular user to conduct business level attacks (i.e. approve expenditure). However, the tester should not test the implementation of access control (session validation) or the SaaS application input filtering (i.e. SQL injection) in the SaaS application. This is because the test involves compromising underlying infrastructure that is out of the scope of authority of the recipient of the penetration test. Therefore, underlying infrastructure is normally not part of the scope of a penetration test unless explicit permission from the CSP is given.

Cloud penetration testing would not challenge the underlying technologies' design and code integrity but rather consider and leverage it. For example,

- Leveraging flaws, common misconfigurations and known vulnerabilities in cloud services, technology and providers **is within the scope** of a cloud-based application/asset test, but
- Forensics, reverse engineering and research of the cloud service or fabric **is not**.

The Shared Responsibility Model

Security in the cloud is tested, not security of the cloud. For example, in an IaaS environment, as per Fig 1, User Access/Identity, Data, Application, and Operating System layers are in-scope. All other components below the red line are under the control and management of the CSP and are, thus, out of scope. The same logic applies to PaaS and SaaS environments with the scope of the penetration test dependent on the service model. The degree of testing and the scope of testing depends on the various services provided by the cloud service provider(s). Nevertheless, the accidental discovery of any flaws and vulnerabilities of the cloud service should still be reported.

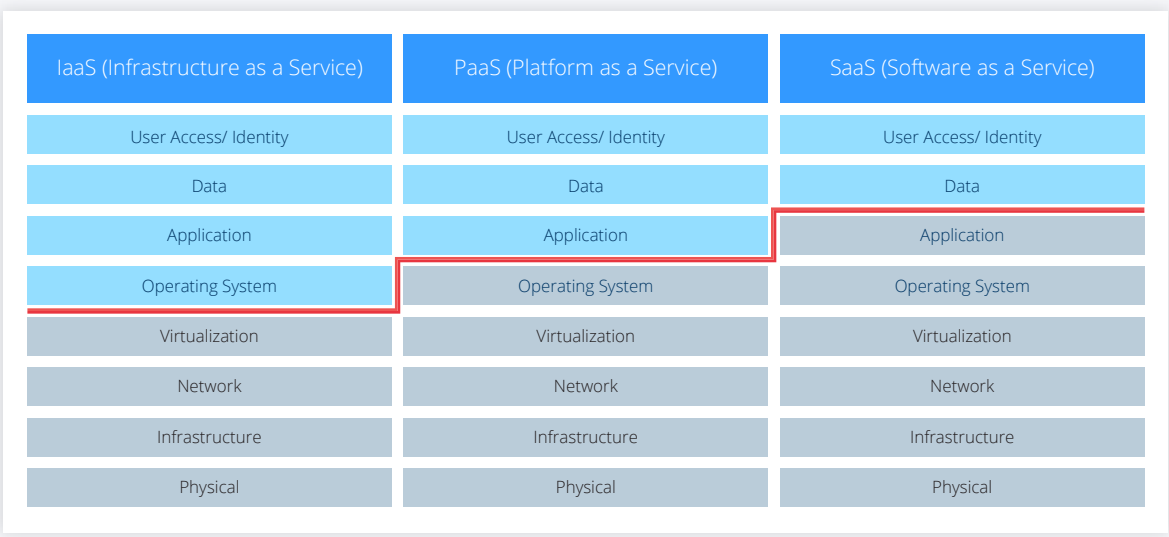


Fig 1. Shared Security Responsibility model

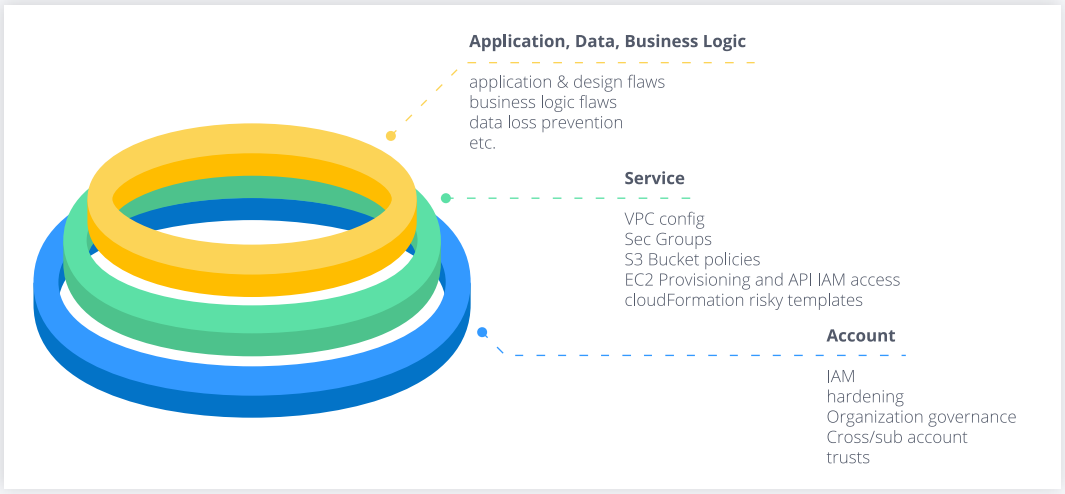
The scope for testing and application of test cases also differs from service model to service model (Fig 1). In an SaaS application, a relatively small scope exists: data and user access/identity controls. In PaaS, the application layer (and some platform configuration) is in scope; all layers lower are excluded. The same principle is applicable to IaaS; the client responsibility is expanded and so does the scope of potential security testing. Implementing or moving a workload to the cloud changes the scope for potential security tests (and breaches!). Additional scope for testing is introduced (e.g. the cloud management plane), but some is still offloaded to the CSP (virtualization, hardware, and sometimes OS).

If the cloud customer gets authorization from the cloud service provider to test cloud components under the control and management of the cloud service provider, additional complexity and scope will be introduced to the scoping of the cloud pentesting service.

The Scope of Security in Public Cloud

Although client-side application testing would be included in IaaS and PaaS testing, this document does not detail testing of the application layer (e.g. SQL injection, XSS vulnerabilities) and the OS layer (e.g. virtual machines) as that has been sufficiently covered by OWASP and other resources. Thus, this document is concerned only with the following domains:

- *Account security* as it relates to user identity and access (e.g. Identity and Access management, logging, account level hardening practices, Cloud Federation and Single Sign-On interface and more)
- *Cloud service security* as it relates to data structures and cloud infrastructure that can be configured by the cloud customer (e.g. s3 bucket policies, vpc network access restrictions, risky cloud formation templates and on)
- *Application / business logic* that is under the control of the end user (e.g. Application and design flaws, business logic flaws, code scripting flaws, data loss prevention, etc.)



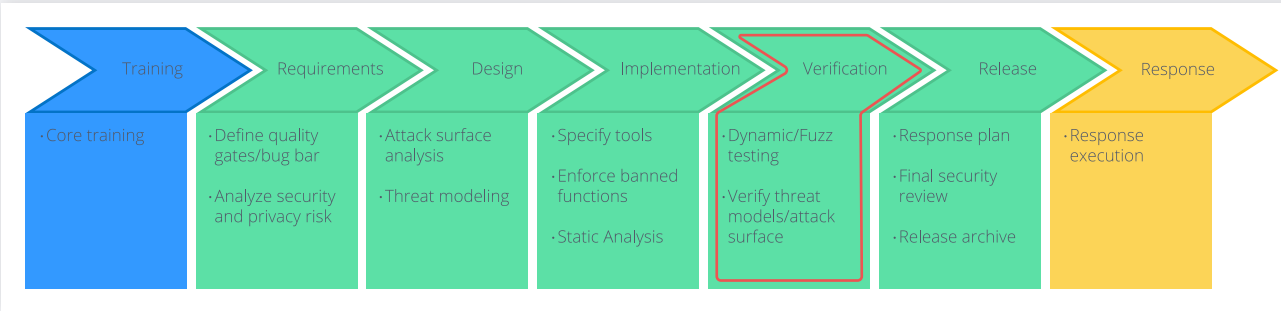
The scoping of a penetration test can and should consider all three domains, even if any are excluded from the scope of a penetration test as the three domains can substantially impact each other.
For example:

- a misconfigured account user (neglect to formulate restricted role based access controls) can and will contribute to the severity and likelihood of an application breach
- an application in AWS could be badly designed, implemented or configured with high cloud privileges (e.g. Assume/Pass Role Cloud Administrator), the breach of which is considered a complete compromise.
- neglecting account level service use and billing thresholds and controls could make the application highly vulnerable to DOS, resource starvation or billing abuse.

CLOUD PENETRATION TESTING IN CONTEXT

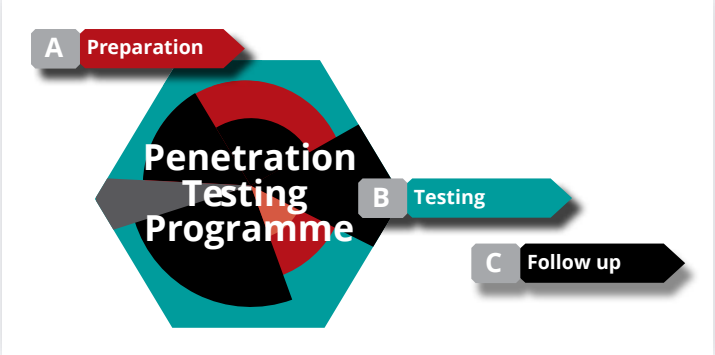
Penetration testing, also referred to as dynamic testing is commonly performed after code has been developed and deployed, even if not in production. It's critical to remember that penetration testing is not necessarily the best or most efficient form of testing. Its appropriateness can be determined based on context and purpose. For example, when security assurance of a live cloud-based product or feature is required, penetration testing by the methodology described in this is advised, but, when the feature is just being designed - threat modelling practices are the best fit.

The Microsoft Secure Development Lifecycle places Security Testing in the Verification Phase, including Dynamic Testing and verification of threat model/attack surface.



Assurance Program

Penetration testing can also be – and often is – carried out as part of a security program. CREST advocates their best practice Penetration Testing Programme -



The CREST program aims to assist with effectively managing penetration testing carried out in or delivered for a consumer organization. It outlines the steps and good practices for utilizing to greatest benefit a good penetration testing service. As such, there would appear to be some 'duplicate scope'; however, it must be said that the **preparation phase** for a recipient of a penetration test is different from the preparation phase of the security service provider (i.e. organization providing penetration testing services). The methodology described in this document is written from the perspective of the security service provider. This perspective aligns with and meets some 'client'/'recipient' program requirements, such as the **testing phase**, in particular the need to '*use an effective testing methodology*'.

The **follow up phase** provides guidance to consumers of security testing on taking action based on the deliverables, as well as assessing the effectiveness of the testing, which is not in the scope of our work within this document. Otherwise, if cloud systems or environments are in scope, then this playbook accounts for it; non-cloud scope still requires its own methodology and reference respectively.

The cloud penetration testing playbook is adapted from CREST Preparation (Part 3) and Testing (Part 4) and suggests five main phases. They are *Preparation*, *Threat Modelling*, *Reconnaissance and Research*, *Testing* and *Report Writing*. This document details the unique cloud security test cases and considerations in each of these phases.

CLOUD PENETRATION TESTING OBJECTIVES

The objective of penetration testing is to identify vulnerabilities in code, configuration and otherwise insecure implementation and to advise on effective mitigations.

It is important to remember that the test cases below consider only unique, cloud-inherited test cases and flaws. The cloud is a canvas with many possible deployments and uses (e.g. workload, storage or container). Depending on what is being hosted and its implementation, the resulting vulnerability or security weakness will differ. Naturally, such components will require their own testing guidelines.

The model we choose to guide the offensive and defensive efforts by priming exploration of what can go wrong is **STRIDE,** a threat model developed at Microsoft for identifying computer security threats. We have decided to group suggested test cases by the STRIDE model since the terminology and format are widely known and used.

- **Spoofing** - Impersonating, masquerading or otherwise falsely assuming an identity, characteristic or claim about oneself. In cloud testing, spoofing often takes the form of stealing cloud environment credentials to leverage their identity's privileges.
- **Tampering** - Sabotage, modification or forgery of records, process or product in a harmful way, or otherwise in a fashion that serves an attacker's other objective or attack chain. In cloud testing tampering often takes the form of altering cloud logs, changing hosted images, and tampering with API, repositories or data.
- **Repudiation** - Creating a situation of dispute, lack or compromise of the authenticity of a record or data. In cloud testing repudiation often takes the form of deleting or turning off cloud logs or leveraging cloud services and mechanisms to mask an action or occurrence.
- **Information disclosure** - The breach of privacy or leak of information to unauthorized persons or to the public domain. In cloud testing information disclosure often takes the form of leak of data from misconfigured public cloud data stores.
- **Denial of service** - The act of making a system, feature or resource unavailable for intended users. In cloud testing, denial of service often takes the form of destruction or encryption of cloud resources, disablement of accounts, credentials or users..
- **Elevation of Privileges** - The act of leveraging a vulnerability or configuration to enable or achieve an elevation of access or privilege beyond what was intended. In cloud testing, elevation of privileges often takes the form leveraging misconfigured IAM permissions that allow escalation or permissions employed by compromised or targeted services and systems.

CLOUD PENETRATION TEST CASES AND CONCERNS

Activities in black are traditional activities already included in standard penetration testing engagements and frameworks. Activities in blue are relevant to cloud environments and should be considered for testing. In most cases, an example or further reference is provided in brackets.

1. Preparation

a. Sign off on the Non-Disclosure, Liability and Testing Agreements with the Client
b. Define and agree on the purpose and scope of the penetration test
    i. Identify testing constraints
    ii. Identify targets and environments in scope
        1. Is the cloud account in scope?
        2. Are cloud Supply Chain services and partners in scope?
        3. How are various tenants considered in the scope? Are they excluded? Are they purposefully targeted? What is considered a separate tenant?
        4. Have the CSPs pentesting approval/constraints been considered?
        5. A detailed assessment of the Attack vectors and Risks in Public Cloud are understood
c. Establish cloud penetration testing approval per Cloud Service Provider and the client, per the appropriate (and often public) security testing procedure
d. Produce/Receive requirements specification
    i. Consider cloud compliance, guidance and frameworks (CSA CCM for example)
e. Tailor, agree and sign off on the penetration testing tools, tactics and procedures (TTP), as well as the methodology
    i. Non-cloud TTPs like OWASP Application testing guide
    ii. Cloud reconnaissance, phishing, account hijacking / password reset TTPs
    iii. Cloud audit tools for identification of best post-exploitation - Azurite, ScoutSuite
    iv. Acceptable and minimum test cases for spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege
    v. Acceptable action on objectives, constitutes and evidence to prove success of testing and meeting objectives
    vi. Implement management control and operation processes
    vii. Appoint points of contact
    viii. Submit and manage change requests
    ix. Resolve testing operational issues
    x. Isolating, restricting and resolve system impact from testing

2. Threat Modelling

a. Refactor client concerns, purpose and specifications into threat models
b. Perform Threat Modelling on the scope
    i. Consider relevant cloud service provider specific, deployment and consumption models threats

ii. Consider industry standard/best practice on cloud threats (top threats)
    1. Treacherous 12, Attack Trees

3. Reconnaissance and Research

a. Conduct standard reconnaissance (records, web, network, IP fingerprinting,osint, people, social media)
    i. Leverage DNS records (N, MX, NS, SPF, TXT, CName, A) to determine cloud providers and services of a targeted domain/organization and potentially mismanaged or hijackable ones
    ii. Leverage identity federation servers reconnaissance via google dorking and DNS records for adfs, auth, fs, okta, ping, sso, sts, oauth, openID, saml, ws, technologies and service providers etc.
    iii. Look for cloud credentials in code and text repositories (such as API keys, federations service private certificates and storage account keys/sas, Azure publish setting file certificates)
    iv. Gather and enumerate cloud users and administrative credentials from compromised credentials dumps and via OSINT
    v. Identify cloud administration, operation, user and chain of supply personnel targets via Linkedin, company website
    vi. Identify cloud services, assets and nameserver records via certificate transparency logs and DNS records (such as company bucket.s3.amazonaws.com)
    vii. Identify scope and related cloud storage instances, accounts and services
    viii. Look for profile, setting and configuration files for cloud accounts and systems (like Azure Publish Settings files, app.config or aws/azure .config files)
    ix. Enumerate account, user and/or role via service API calls (such as AWS enumeration with a known or common resource identifier within the account or blindly with UpdateAssumeRolePolicy)
    x. Conduct post exploitation environment / account reconnaissance to determine account id, aliases, account organization structure and cloud model, compromised user IAM, (and) other users
    xi. Identify different cloud model accounts (such as public cloud AWS vs AWS Government Cloud), different account types (like Azure ARM vs Azure ASM accounts vs Azure storage accounts)
    xii. Analyze mobile applications and native applications code for cloud service / account secrets, users, roles, resource names (arn, AWS key, Azure storage account name, AWS bucket name)
    xiii. Conduct post exploitation environment / account reconnaissance to determine high value systems, assets and users
b. Conduct research
    i. Identified assets reconnaissance for
        1. known vulnerabilities
        2. common misconfigurations
        3. exploitation tools and methods
        4. Review cloud technology and service provider for security bulletin; they may produce vectors for unpatch compromise (AWS Bulletin)
    ii. Attribute recon findings to Threat Models

4. Testing

a. Validating baseline security requirements
b. Employ security test cases, guides and checklists relevant to domain & technologies
   - web? mobile? mative? serverside? API?
   - c# mvc? objective c IOS? Python redhat? c++ winforms
c. Test for Spoofing of user identity and other entities
   i. Steal hardcoded serverless workloads function (a workload implemented as a function) credentials and secrets (like hardcodedAzure function code or by pulling a lambda deploy package)
   ii. Attempt load balancer MiTM for session hijacking (elb) by cloud service configuration or load balancer instance compromise
   iii. Attempt domain transfer to another registrar for domains not transfer prohibited (Route53, aka domain hijacking)
   iv. Steal environmental variables and local file credentials to leverage and impersonate user identity (such as ~/.aws, instance metadata, shell variables, azure ServiceBusExplorer.exe utility .Config file, ecs task definitions orAzure ARM Profile Tokens)
   v. Steal credentials from metadata of proxy or http forwarding servers (credentials in AWS meta)
   vi. Steal cloud workload credentials (AWS metadata sts or Azure Linux Agent (waagent) folder credentials)
   vii. Compromise default privileged service and user accounts in legacy cloud environments and services (likeAzure old ASM co-administrator accounts or Azure Storage Account keys)
   viii. Steal cloud console or server certificates (like Azure asm certificates)
   ix. Steal cloud unique credentials (like AWS sts temporary service token or azure Shared Access Signature (SAS) tokens)
   x. Steal credentials from or leverage privilege to operation of a cloud key service (aws kms, azure key vault)
   xi. Perform spear phishing against cloud users, administrators and chain of supply persons and companies
   xii. Leverage compromised or misconfigured cloud email service for business email compromise and further phishing (for example, if SES is configured to allow sending from @company.com, then IAM permissions of ses:* can send an SES email that will appear to originate from internally)
   xiii. Stealing cookies, secrets, passwords , kerberos tickets, Identity Tokens
d. Test for Tampering
   i. Alter data in datastore for fraudulent transactions or static website compromise (s3, rds, redshift)
   ii. Alter a serverless function, logic app or otherwise a business logic implementation for action on objective or escalation (AWS lambda orAzure logic apps)
   iii. Alter billing threshold and alerts (AWS expenditure, fluctuation custom threshold and cloudwatch alerts)
   iv. Change application, website or otherwise code integrity for resource abuse, persistence, exfiltration or other (AWS s3 static websites orAzure websites)
   v. Create or alter a DNS Record record set in a trusted zone and/or certificates for the resource record set to divert traffic, create phishing sites and abuse the brand (AWS ACM,

AWS Route53, Azure DNS Service)

vi. Alter data in local sql or mysql databases

e. Test for Repudiation
   i. Operate in regions where logging is not enabled or disable global logging (like CloudTrail)
   ii. Alter log files in a non-validated log store or disable validation (like cloud trail log validation)
   iii. Disable network traffic analysis / logging (VPC flowlogs)
   iv. Disable cloud alerting to prevent detection and response (like cloudwatch alerts, GuardDuty, Security Hub, or Azure Security Center)
   v. Disable data store access logging to prevent detection and response (cloudtrain data access, s3 access logging, redshift user activity)
   vi. Alter log retention or damage the integrity of logs (s3 lifecycle, kms decryption cmk key deletion/role privilege lockout)
   vii. Change local windows / Linux logs
f. Test for Information disclosure (privacy breach or data leak)
   i. Leverage misconfigured and default security groups and access lists for exfiltration of data to ANY internet IP address (vpc acl, instance sgs)
   ii. Attempt DB caching and in memory caching data leak (elasticache) using service endpoint and mitm
   iii. Create new big data jobs to processes and output sensitive data to an accessible data store(emr, s3)
   iv. Exfiltrate data from publicly accessible datastore services (s3, rds, rds snapshots, redshift cluster, elastic search domains) or private stores with cli / dumps (s3 aws cli get, dynamodump), and/or configure them accordingly for exfiltration).
   v. Employ cloud email and sms distribution services to exfiltrate data (ses, sns)
   vi. Access misconfigured message queues to access potentially queued sensitive data (AWS SQS)
   vii. Steal and leverage virtual machine metadata (such as VPC, subnet, account, iam roles, role credentials)
   viii. Steal meta information from metadata of proxy or http forwarding servers (credentials in AWS meta)
   ix. Steal virtual machine images and snapshots from storage accounts; analyze them for sensitive data (likeAzure vm vhd snapshots from storage accounts, public or private AWS EBS Volume Snapshots and AMIs)
   x. Fingerprint server and application versions and frameworks, detect sensitive PII in application logs
   xi. Attempt MiTM for data theft
g. Test for Denial of service (D.o.S)
   i. Destroy / encrypt data in datastores not backed up or destruction protected (s3, rds)
   ii. Deny servicers or operability of servers and clients by flooding email or sms messages from a cloud environment (AWS sns, ses)
   iii. Destroy cloud services configuration, datastores and/or accounts (sufficient to use --dry-run AWS cli flag or prove you have the privileges to)
   iv. Deny access to a KMS (CMK for AWS) key by deleting all the IAM users or roles that have access to the key (use --dry-run AWS cli flag or prove you have the privileges to)
   v. Perform a volume-based denial of service or application denial of service attack on an application; practice extreme diligence and caution per CSP and client policies and agreement

h. Test for Elevation of privilege
   i. Trigger Cloud Orchestration Automation with higher privileges (for instance, Cloud formation stack with highly privileged roles assumed)
   ii. Run or deploy a workload with an assigned/passed service or role, export instance credentials for those privileges (such as ec2 passed role and meta credentials)
   iii. Leverage policy write capability to change or create an unrestricted policy assigned to a user (like am:CreatePolicyVersion)
   iv. Change the default policy for a user or new users to include additional privileges (like set-default-policy-version)
   v. Create or reset a login, access key or temporary credential belonging to a high privilege user (like iam:CreateAccessKey, sts or iam:UpdateLoginProfile)
   vi. Attach or update a policy to a role, group or asset you have access to (like iam:AttachGroupPolicy, iam:PutUserPolicy, sts:AssumeRole)
   vii. Leverage developer and alternative consoles to execute privileges on their behalf (AWS Glue Console endpoint with pass role, Azure machine learning studio)
   viii. Leverage data or code pipelines to execute operations on behalf of their assumed roles (AWS data pipeline ShellCommandActivity, inject python code into a pickle celery sqs queue)
   ix. Pass roles and assign high instance privileges to virtual machines, which can be then be controlled and used for AWS API calls (such as create-instance-profile and iam:passrole)
   x. Steal application or code management credentials using descriptive privileges (like Get-AzureWebsite -Name webappname)
   xi. Export service and other account type keys (likeAzure Get-AzureStorageKey -StorageAccountName “Storage_Account_Name”)
   xii. Add users, assets or accounts to existing roles or groups with higher privileges (leverage privileges such as iam:AddUserToGroup)
   xiii. Process hooking, process injection, windows access token manipulation, leveraging misconfigured sudo capabilities

i. **Test for Other Cases and Objectives** (Test for non MS Threat Model Stride cases and actions of objectives)
   i. Lateral Movement
   ii. Leverage misconfigured security groups and access lists for lateral movement between assets in the Cloud(EC2, RDS, other), from account to account (AWS cross account assume role)
   iii. Create an additional interface / assign and IP address in target network / subnet on a compromised machine (like assigning a secondary private IPv4 address or interface to an AWS ec2)
   iv. Create jobs or serverless actions to add root certificates and ssh private keys to machines and users (such as AWS lambda)
   v. Steal virtual machine images from storage accounts, analyze them for passwords, keys and certificates to access live systems (likeAzure vm vhd snapshots from storage accounts)
   vi. Gain OS level access to Instances/VMs via workload management service privileges (AWS SSM or Azure Agent)
   vii. Exploit applications and services on local network systems; leverage file shares, scripting frameworks like powershell, OS orchestration like WMI and administration frameworks like configuration manager

j. Persistence
    i. Assign a public IP to a compromised / internal resource (AWS cli / console - elastic IP)
    ii. Establish an alternative cloud native / service control interface (such as AWS glue console, workspaces or Azure cloud shell/serial console)
    iii. Configure account / user recovery details for persistence, including backup contact methods (like AWS alternative contacts)
    iv. Edit custom machine images and templates to include persistence mechanism (like reverse shells in AWS custom AMIs)
    v. Establish inter-cloud vendor, cross account connectivity persistence with an account under your control (such as a VPC endpoint that allows all traffic from your accounts to an internal network, in a compromised account)
    vi. Publish an internal resource via a configured load balancer (like ssh, rdp or 80 via an ELB load balancer)
    vii. Employ a workload / alert to maintain persistence by and inform on compromise / discovery (AWS lambda functions, cloudwatch, ec2)
    viii. Invokable function and or API with administrative permissions and otherwise IAM/KMS affecting privileges. Call it to get the API Key/Secret/Token. (such as AWS lambda & API gateway)
    ix. Employ a privileged event source driven workload as a backdoor, shell or persistence mechanism (lambda to add rules to security groups, keys to users, listen on a log, ec2, elb or other events to pipe shell commands, trigger events on SQS commands from the internet)
    x. Create systems management commands or abuse instance metadata for scheduled and triggered command and control (AWS systems manager, modify EC2 UserData to trigger a reverse shell)
    xi. Perform remote code execution with cloud native systems management utilities (aws systems management)
    xii. Implement a startup script for virtual machines (like Azure startup scripts)
    xiii. Add credentials to existing users and services (such as AWS security credentials access key)
    xiv. Create shadow administrative users or roles with obscure but escalation able privileges (like AWS CreatePolicyVersion and SetDefaultPolicyVersion privileges)
    xv. Create local instance users with remote control privileges (ssh / rdp)

5. Report

a. Report key findings
    ii. Refer to Industry standard and vendor best practices cloud security practices and configurations (Cloud Security Alliance CCM Controls, AWS Well Architected Framework)
    iii. Collect and report evidence in cloud accounts, aliases, metadata, keys, amis
b. Follow up
    i. Crest Follow-up phase items such as implement monitoring plan and assess testing effectiveness

LEGAL

Penetration testing needs to comply with all applicable local and national laws. A formal, written and signed client authorization should be obtained before any penetration testing services are rendered.

Testers and clients should also consider any legal supply chain requirements. Third-party providers and services that fall within the scope of the test may have their own testing guidelines, procedures, restraints and requirements. For example,

- Amazon Web Services no longer mandates a testing permit issued by them. However, there are still several constraints which are mentioned here.
- Tests that may impact European citizens' personal Identifiable information (PII) must consider that any such data must be handled per the GDPR guidelines (anonymizing, safely handling in transit, breach reporting). That may often be overlooked since the GDPR mandates such testing in the first place.

TRAINING AND RESOURCES

Training

We advocate the study of CSA Security Guidance for Critical Areas of Focus in Cloud Computing. In addition, the following resources might also be useful.

Labs & Resources

Few offensive cloud security hands-on training opportunities are available outside of setting your own or on the job experience; however, the following are advisable:

- FLAWS - challenges you to learn by moving through a series of levels, about common mistakes and "gotchas" when using AWS.
- CloudGoat Rhino Security Labs' "Vulnerable by Design" AWS infrastructure setup tool

Tools

Many open tools for assessing and testing security of and within cloud environments are available. A few honorable mentions include

- NCC Groups open source cloud auditing tools (ScoutSuite and more) - a multi cloud auditing suite
- LazyS3 - a tool for enumerating AWS s3 buckets
- CloudBurst - a collection of tools inclusive to enumeration of (Azure) services, data stores, credentials harvesting and more
- Nimbusland - a tool for resolving cloud IP address spaces
- pacu - a post exploitation AWS toolkit
- Shodan - the search engine for Internet-connected devices can assist with identification, research into and reconnaissance of public cloud-based systems and assets.

A more comprehensive register of open cloud security tools is available at ToniBlyx - link.

CONCLUSION

Since its conception almost 10 years ago, cloud computing has gone from being a new technological paradigm with many critics and doubters to a widely accepted form of information technology. While cloud computing usage has become normalized, security best practices and processes still have to be continuously updated, developed and refined to ensure its success. This document presents a collective effort to provide guidance for the penetration testing of systems in public cloud environments. Penetration testers will be able to use the test objectives in this document to test the security of public cloud systems and environments. Legal and other associated concerns are also discussed and aim to educate key decision makers on the complexities of penetration testing in a multi-stakeholder and layered information technology stack. The CSA Top Threats Working Group hopes that providing advice regarding such key topics will help mature cloud penetration testing and in the process create a more secure cloud computing environment for everyone.

REFERENCES

1. https://www.crest-approved.org/wp-content/uploads/CREST-Penetration-Testing-Guide.pdf
2. https://www.owasp.org/index.php/OWASP_Testing_Guide_v3_Table_of_Contents
3. Penetration Testing Execution Standard (PTES)
4. http://www.vulnerabilityassessment.co.uk/Penetration%20Test.html
5. https://www.youtube.com/watch?v=ge6gJkb3nXE A Penetration Tester's Guide to the Azure Cloud
6. https://labs.mwrinfosecurity.com/assets/BlogFiles/mwri-a-penetration-testers-guide-to-the-azure-cloud-v1.2.pdf
7. https://vimeo.com/214855977
8. Gone in 60 miliseconds aws
9. https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation/
10. https://github.com/dagrz/aws_pwn/blob/master/miscellanea/Kiwicon%202016%20-%20Hacking%20AWS%20End%20to%20End.pdf
11. Daniel Grzelak AWS Account Backdoor, Daniel Grzelak AWS account post compromise
12. https://medium.com/@cloud_haxor/enumerate-aws-account-ids-and-iam-resources-c374843cfdf4
13. DEF CON 25 - Gerald Steere, Sean Metcalf - Hacking the Cloud
14. https://www.cyberark.com/threat-research-blog/cloud-shadow-admin-threat-10-permissions-protect/
15. https://gbhackers.com/cloud-computing-penetration-testing-checklist-important-considerations/
16. https://www.cloudconformity.com/conformity-rules/
17. https://attack.mitre.org/wiki/Lateral_Movement
18. https://www.blackhat.com/docs/us-16/materials/us-16-Amiga-Account-Jumping-Post-Infection-Persistency-And-Lateral-Movement-In-AWS-wp.pdf
19. http://www.chrisfarris.com/post/lateral-movement-aws/
20. BSidesSLC 2017 -- Bryce Kunz -- Pwned Cloud Society
21. Blue Cloud of Death - Red Teaming Azure
22. https://www.microsoft.com/en-us/securityengineering/sdl/howto
23. AWS ELB neglecting internal server TLS certificates
24. http://www.irongeek.com/i.php?page=videos/derbycon8/track-3-16-cloud-forensics-putting-the-bits-back-together-brandon-sherman-

