Introduction

Introduction

Hello, my name is Jonah. I am a senior penetration tester. Prior to this, I was a penetration tester in KPMG. <and a SOC operator in the SAF during my national service>

I'm also a student, and I'm undertaking a computer science degree under the University of Wollongong.

On top of the certI've recently attained OSEP yesterday and I'll be starting on OSED next week. That's all for me.

I was accredited with a CVE on multiple XSS via Unicode transformation and I'm currently in touch with organizations such as Microsoft and Oracle to address vulnerabilities. That's all for me.

Why did you decide to leave Privasec?

- **There is a lack of forecasted engagements in the pipeline, engagements that come are on an ad hoc basis**
- **And thus, I've been placed on various non-technical tasks such as creating articles for publicity, and creating checklists for new services such as source code review and red teaming to bring up the technical skillsets of the team and the firm as a whole**
- **Although it is meaningful and it helps the team, I feel that I am not presented with enough opportunities for growth in the firm**
- **Thus, I decided that it would be in my best interest to seek for a firm which has a more conducive environment for myself to advance with, in which my technical knowledge is constantly challenged so as to broaden my areas of knowledge of what I know and what I don't know and to keep my technical skillsets sharpened.**

What's your day to day like?

- **In my previous position, I've performed a full suite assessment of an application which includes source code review, web, and network penetration testing.**
- **Also, I recently did a PUBLIC SPEAKING ON RED TEAMING and the Adversarial Attack Simulation Exercises (AASE) which is the guideline for red teaming for finance sectors in Singapore.**
- **However, due to the lack of forecasted engagements in the pipeline, engagements that come are on an ad hoc basis.**
- **Hence currently, I've been placed on various non-technical tasks such as creating articles for publicity, and creating checklists for different services such as source code review and red teaming to bring up the technical skillsets of the team and the firm as a whole**
- **However, in my previous firm, I was primarily engaged in Web and Network penetration tests, thick client assessments, and red teaming for high stake clients in various sectors.**

Difference between red team and penetration test?

- **A penetration test focuses on assessing an organization's infrastructure, from networks, to web applications, to mobile devices in an effort to identify as many vulnerabilities as possible.**

- **On the other hand, the objective of a Red Team is to target the 'crown jewels' of an organisation, and this will be carried out without prejudice to any attack vector, in other words, anything goes.**
- **Attacks are multi-layered and can zero in on errors across people, processes and technologies; an unsuspecting employee? A broken lock? Or outdated technologies or even the lack of.**
- **What this means is that it challenges the organization's security in depth, as a whole.**
- **Rather than putting a priority on finding as many vulnerabilities as possible, a red team attempts to test how well an organization's security team reacts and responds to various threats, simulating an attack by real threat actors in the wild.**

Methodologies

- **MITRE Attack framework**
  - **Reconnaissance**
  - **Resource development**
  - **Initial Access**
  - **Execution**
  - **Privilege escalation**
  - **Lateral Movement**
- **OWASP PTES (Penetration Testing Execution Standard)**
  - **Pre-engagement Interactions**
  - **Intelligence Gathering**
  - **Threat Modelling (Microsoft STRIDE and DREAD)**
    - **We make test cases based on threat intel reports that tells attacks targeting specific industries, from Mandiant. For example, we modeled a red teaming engagement based of Fireeye's threat report on Ryuk, with zerologon and ransomware being the prime focus.**
  - **Vulnerability Analysis**
  - **Exploitation**
  - **Post Exploitation**
  - **Reporting**
- **OWASP Top 10 and WSTG**
  - **A01:2021-Broken Access Control**
  - **A02:2021-Cryptographic Failures**
  - **A03:2021-Injection**
  - **A04:2021-Insecure Design**
  - **A05:2021-Security Misconfiguration**
  - **A06:2021-Vulnerable and Outdated Components**
  - **A07:2021-Identification and Authentication Failures**
  - **A08:2021-Software and Data Integrity Failures**
  - **A09:2021-Security Logging and Monitoring Failures**
  - **A10:2021-Server-Side Request Forgery**

Network Penetration Testing

0) Common vulnerabilities

- **Insecure protocols in use (FTP, Telnet, TFTP)**
- **Insecure software in use (Outdated software with known vulnerabilities)**
- **Outdated or unpatched software (usage of softwares with known vulnerabilities and public exploits, or using end of life operating systems like Windows server 2003)**
- **Security misconfigurations allowing weak passwords**

1) Phases

There are generally 5 phases of a penetration test:

- **Reconnaissance**
- **Vulnerability Assessment**
- **Exploitation**
- **Post-exploitation**
- **Privilege Escalation**

1) Reconnaissance

Overview

The objective is to gather more information of the targets. I will use a combination of both automated and manual scanning tools such as Nessus and Nmap, and packet analysis tools such as Wireshark to perform Host, Port, and Service discovery.

After identifying the open services, I will then proceed to enumeration. The goal is to map out the attack surface and identify potential entry points and useful information for further leverage such as banner grabbing for running service versions. (and enumeration for domain users and groups)

Common tools

- **Wireshark**
- **Nessus and Nmap**
- **The Impacket framework**
    a. **rpcdump.py @192.168.0.100 | grep MS-RPRN**
- **Check for Netbios Null sessions**
    a. **Might be able to connect to IPC$ hidden drive**
    b. **Use rpcclient to enumerate for system password policies, use RID cycling to enumerate for domain users and groups**
- **FFUF/Dirb/Gobuster for Web fuzzing**
- **Enum4linux**

Questions

- **Difference between authenticated and unauthenticated scans**
    - **An unauthenticated scan can only examine publicly visible services.**
    - **It will have no visibility to internal running services, patch audits and host configuration issues.**
    - **As an authenticated user generally has access to more resources like network shares, an authenticated scan will be able to identify these resources and perform relevant checks on it appropriately.**
    - **An authenticated scan, if in the context of a domain-joined user, can also perform relevant checks for Active Directory.**
    - **WMI is required for authenticated scans.**
    - **If no admin rights, Nessus has to fall back to perform a patch audit through the registry.**
- **TCP port scan vs UDP port scan**
    - **In a TCP port scan, a scanner identifies an open port via the TCP 3-way handshake. To elaborate, the scanner sends a SYN packet to the port, if a port is open, it will respond with a SYN-ACK response.**
    - **Otherwise, it will either send a RST response which indicates that a port is closed or no response will be returned to which the scanner will not be able to verify the state of the port (aka filtered state)**
    - **In a UDP port scan, since there is no handshake, the scanner will be unable to identify the state of the port, whether it is open or closed. Hence, it can only send a relevant request in hopes that the service responds to it accordingly i.e., sending a UDP DNS request to UDP 53. If the service responds, the port is open. Else, it is considered as filtered.**
- **Difference between Nessus and Nmap**

    - **Both performs host, port, and service discovery.**
    - **Nmap to an extent covers vulnerability discovery with the use of nse scripts**
    - **However, Nessus's vulnerability database is more comprehensive, and as such it contains more test cases. Thus, it more accurately identifies vulnerabilities in services.**
- **Why use nmap then?**
    - **As with all tools, nmap is just another tool in the shed. We can opt for other tools such as masscan.**
    - **The objective is to use 2 different scanners, so when cross-referenced, we can sieve out false-positives that arise from one scanner.**
    - **This way, we can achieve more coverage and adopt a more holistic approach to gathering information.**

2) Vulnerability Assessment

Overview

- **After gathering useful pieces of information on the running services, I will proceed to run service-specific test cases. Such test cases can be as simple as default credentials, anonymous FTP and SMB null sessions.**
- **I will also search for relevant vulnerabilities and its respective exploits through vulnerability databases such as cvedetail, and exploitdb.**
- **I will then manually use tools or auxiliary scanners to confirm the existence of the vulnerability by attempting to elicit a vulnerable response from the service.**

Test cases

- **Default credentials in Services**
    a. **FTP**
    b. **Database service such as MySQL/MSSQL**
    c. **Tomcat manager**
        i. **manager:s3cr3t**
- **FTP**
    a. **Anonymous FTP enabled. Might be able to read sensitive files, or upload documents. Use grep to sieve through files with strings like user or password.**
    b. **Insecure versions of FTP like ProFTPD 1.3.5 mod_copy.**
- **SMB**
    a. **Check for SMB version, if is version 1 or 3.1.1 (SMBGhost)**
        i. **SMBGhost is affected by a compression buffer overflow, leading to arbitrary command execution.**
    b. **SMB signing disabled**
        i. **We can use crackmapexec to check for relayable targets without SMB signing.**
        ii. **If there are relayable targets, and NetBIOS Name Server and Local-Link Multicast Name Resolution (NBNS and LLMNR Poisoning) is enabled, we can spoof an authoritative source for name resolution on a victim network by responding to LLMNR.**
        iii. **Use Responder in conjunction with mitm6 to identify if NBNS LLMNR is enabled.**
        iv. **Responder.py -I eth0 -wrf**
    c. **Accessible Shares (SMB/NFS)**
        i. **We can check if there are any accessible shares. We can recursively sieve through contents for sensitive information with tools such as smbmap.**
- **DNS**
    a. **Exploits such as SigRed (SIG resource record BOF)**
- **SNMP**
    a. **SNMP 1,2,2c does not encrypt traffic**
    b. **Can enumerate for host information via default community strings**
        i. **Public**
        ii. **Private**
        iii. **Community**
    c. **snmpwalk**
- **SMTP**
    a. **EXPN VRFY user enumeration**

- **RPC**
    a. **Printnightmare**
- **Kerberos**
    a. **AS-REP roasting**
        i. **Roast for users set with the "Do not require Kerberos pre-authentication" property**
        ii. **Run it against gathered users accounts from RID cycling through rpcclient**
        iii. **Retrieve AS-REP hashes with Impacket-GetNPUsers.py and crack it with hashcat**
- **Finger**
    a. **Enumerate if user exists**
    b. **finger USERNAME@$ip**

3) Exploitation

Overview

- **The objective is to create a simple proof-of-concept to prove that the service running is indeed vulnerable and exploitable.**
- **After sourcing a publicly available exploit, I will then proceed to test it against the service. Depending on the depth of exploitation allowed, I will then use an appropriate payload, from popping calc.exe to gaining a foothold via a reverse shell into the application.**

Test cases

- **Tools**
    a. **Metasploit framework**
    b. **Impacket framework**
    c. **Crackmapexec**
    d. **Searchsploit**
    e. **Github**
- **Brute force of common services (SSH, RDP, FTP, MSSQL)**
    a. **Hydra**
    b. **Medusa**
- **DNS**
    a. **Perform Zone Transfer**
    b. **dig axfr**
- **Ldap injection**
- **Potential exploits**
    a. **Printnightmare if authenticated**
    b. **Zerologon**
    c. **Petitpotam**

4) Post-exploitation

Overview

- **After successful exploitation, we will be hopefully be greeted with a privileged shell if lucky, or a normal privileged shell.**
- **We will then perform basic host enumeration to assess our current privilege and gain situational awareness.**

Test cases

Run checks for:

- **OS details and patches**
    a. **wmic qfe**
    b. **systeminfo**
- **Vulnerable applications**
- **Potential privilege escalation vectors such as:**
    a. **Service Unquoted Paths/Weak Service executable permissions (Accesschk)**
    b. **AlwaysInstallElevated**
    c. **SUIDs and writeable cron (/etc/cron) for Linux**
    d. **GTFObins**

Integrity levels in Windows

1) **Untrusted**
2) **Low**
3) **Medium**
4) **High**
5) **System**
6) **Installer**

5) Privilege escalation

Overview

After identifying potential vectors for privilege escalation, we will then proceed to escalate our privileges.

Techniques

1) **Search order DLL hijacking**
2) **Replace weak service executables with a malicious executable**
3) **Run local privilege escalation exploits**

Interesting engagements

- **QNAP Local File Inclusion (Network Access Share software)**
    a. **I was able to gather 2 user hashes through disclosing the /etc/shadow file**
    b. **I then unshadow and cracked the hashes**
    c. **I was then able to access these shares with the cracked credentials to uncover sensitive personal information such as passports and spreadsheets containing biodata of employees.**
- **NBNS and LLMNR was enabled and SMB signing was disabled**
    a. **I attempted to relay the net-NTLMv2 hashes with Responder (Turn off SBM server), mitm6, and ntlmrelayx to the server, however it was unsuccessful as these users were not privileged.**
    b. **I then proceeded to gather net-NTLMv2 hashes instead to crack.**
    c. **After successfully cracking, I then proceeded to roast the service principals with impacket-GetUserSPNs to gather Service Tickets.**
    d. **I then proceeded to crack the service tickets.**
    e. **The cracking succeeded and I got lucky. I was able to confirm with crack-map-exec that one of the service principals was a member of the global domain admin group and was administrator on one of the victim servers.**
    f. **I then used crack-map-exec to dump the SAM hashes and LSA secrets.**
        i. **cme smb 192.168.1.0/24 -u UserNAme -p 'PASSWORDHERE' –sam**
        ii. **cme smb 192.168.1.0/24 -u UserNAme -p 'PASSWORDHERE' –lsa**

Interesting exploits

- **Recent word 0day CVE-2021-40444**
  - **Vulnerability exists in the legacy MSHTML rendering engine**
  - **The exploit requires an attacker to trick a user into opening an Office document containing a MHTML OLE object that points to an attacker-controlled endpoint.**
  - **The engine renders the malicious website's JavaScript code that creates an ActiveX control**
  - **The ActiveX control then downloads a cabinet archive (.cab) file which contains a malicious DLL and executes it as a control panel file (.cpl) via control.exe.**
    - **control.exe .cpl:../../../AppData/Local/Temp/championship.inf.**
  - **The control.exe process will then execute the DLL in the context of rundll32.exe.**
- **Razer synapse system shell**
  - **An easy system shell on any device which you have physical access to and ofcourse with the help of a cheap razer peripheral**
  - **Plugging in a razer mouse will prompt the USB driver to load.**
  - **It will then run the installer with system rights**
  - **A explorer.exe prompt for install location will popup, in which as the process is spawned with system rights, you can run spawn a PowerShell terminal by right clicking and opening in PowerShell**
- **HiveNightmare**
  - **Allows for copy the SAM from the volume shadow copy into the current directory**
- **PrintNightmare**
    - **The Microsoft Windows Print Spooler service fails to restrict access to the RpcAddPrinterDriverEx() function, which can allow a remote authenticated attacker to execute arbitrary code with SYSTEM privileges on a vulnerable system.**
    - **Allows a low privileged authenticated to add a remote printer driver to a victim server, in the form of a malicious DLL, which when loaded, will result in arbitrary command execution.**
- **PetitPotam**
  - **Requirements**
    - **Requires foothold**
    - **ADCS is configured to allow NTLM authentication;**
    - **NTLM authentication is not protected by EPA or SMB signing;**
    - **ADCS is running either of these services:**
    - **Certificate Authority Web Enrolment**
    - **Certificate Enrolment Web Service**
  - **Coerce target DC to authenticate to attacker-controlled box via MS-EFSRPC**
  - **Relay it to AD CS. The AD CS provides certificate for target DC$ computer account**
  - **Use target DC's computer account to request for its Kerberos TGT**
  - **Use target DC computer account TGT to perform DCSync to pull NTLM hash of krbtgt**
  - **Now can forge golden tickets to impersonate any domain user**

Others

Encryption vs Hashing vs Encoding

Encryption	Hashing	Encoding
Ensure confidentiality of data	Ensure Integrity of data	Used for neither confidentiality nor integrity, but rather is used to transform data into a standard format in which it can be properly consumed
Two-way function; encryption can be reversed by decrypting with a secret key	One way function; not possible to go from output to input. Any modification to the input will result in a drastic change to the hash	N/A
AES-CBC in block mode, RC4 in stream mode (CBC is block mode, CTR/CFB is stream mode)	MD5, SHA1, SHA256	Base64, ASCII encoding

Symmetric and Asymmetric cryptography

There are two forms of encryption, namely symmetric and asymmetric encryption.

Symmetric cryptography is a process that uses the same key to both encrypt and decrypt plaintext and ciphertext respectively.

Some popular examples of symmetric cryptography are the Rijndael block cipher (which is currently adopted as the Advanced Encryption Standard or AES), and the RC4 steam cipher.

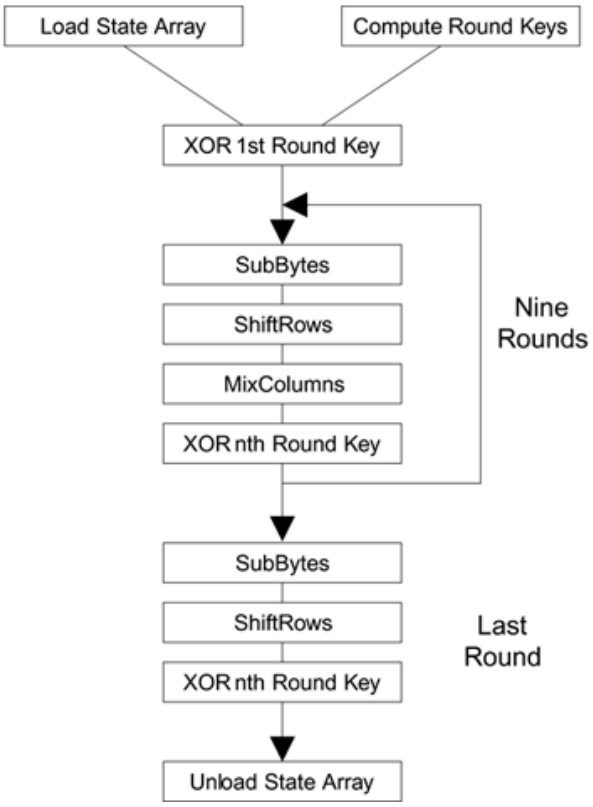
And for asymmetric cryptography, we have RSA encryption algorithm, Diffie-Hellman key exchange, and Elliptic Curve.

What is PKI?

- **Public Key Infrastructure is a system built on a hierarchical chain of trust. It uses trusted third parties to verify the identity of the entities and uses digital certificates for authentication.**
- **An entity will first enrol with the Registration Authority. The identity of the entities' is then verified and is then forwarded to the CA.**
- **The client then generates a public-private key pair and sends the public key along with additional information to the CA.**
- **The CA then creates a x509 digital certificate consisting of the user's public key, organization's name, and the date of validity. The certificate is signed by the CA with its private key and issued to the entity.**
- **Should the certificate expire or the private key has been compromised, the CA will revoke the certificate.**
- **OCSP provides information about SSL certificates issued from CA while CRL provides a list of revoked certificates.**

The client can then store the certificate in a certificate store or a Hardware security module (HSM)

How AES works



RC4 weakness

- **Weakness resides in implementation, in which an IV reuse can happen**
- **With a small IV (24 bits), IV reuse can happen**
- **By capturing 2 ciphertexts with the same IV, will allow an attacker to compare against the two and deriving the key and decrypting both plaintexts**

How DNS works

- **Type test.google.com in browser**
- **Browser checks for the browser cache of a DNS record to find the corresponding IP address of google.com. In order it checks for browser cache, then OS cache, then router cache, then local DNS server cache.**
- **If all of which do not exist, the local DNS server can either perform a recursive or iterative DNS query for us.**
- **In a recursive query, the DNS server performs DNS queries on behalf of us, in which it will initiate DNS queries to multiple DNS servers until it finds the answer for us.**
- **In an iterative query, instead of fetching the answer for us, we will be referred to another DNS server to which we will perform the DNS query against. The process continues with the Root server referring us to the .com server, and the .com server finally referring us to google.com.**
- **As google.com is the SOA, it will return test.google.com's IP address.**

How TLS works

- **TLS 1.2**
  - **Step 1: Client sends client hello, its supported protocols and cipher suites to server, along with a nonce.**
  - **Step 2: Server replies with server hello, its choice of the cipher suite, its certificate, a session ID and a nonce.**
  - **Step 3: Once client verifies certificate with CA, it sends a pre-master secret, and encrypts it with the server's public key.**
  - **Step 4: With the pre-master secret, both the client and server generate a master key along with the ephemeral session keys. These session keys are symmetric keys and will be used to encrypt data.**
  - **Step 5: Client sends "Change cipher spec" to tell server that it will switch the symmetric encryption and along with it, it sends "Client finished"**
  - **Step 6: Once server received "client change cipher spec", the server switches to symmetric encryption and sends "server finished".**

- **TLS 1.3**
  - **Zero Round Time Trip**
  - **Uses Diffie Hellman Ephemeral exclusively**
  - **Step 1: Client sends list of supported cipher suites, but guess which key exchange protocol the server is likely to select. Client sends its key share for that particular key exchange (i.e., Diffie-Hellman ephemeral)**
  - **Step 2: Server replies with "server hello", the key exchange protocol it has chosen, server's key share, certificate and "server finished" message. This saves 4 steps and one trip.**
  - **Step 3: Client checks the server certificate, generates keys from the server's key shar, and sends the "Client Finished" message. From here on, the symmetric encryption of the data begins.**

- **CBC mode**
  - **Padding oracle attack**
    - **CBC padding pads with the number of required pads as the padding character i.e., if 1 pad is required, pad is 01.**
    - **If website responses with 200 OK if padding is correct and 500 internal error if not, this is a padding oracle**
    - **With this, an attacker will construct a block to retrieve one byte of the plaintext, as this is a chosen-ciphertext attack.**
    - **An attacker will intercept a cipher text and retrieve byte by byte the plaintext.**
- **ECB mode**
  - **Encrypts identical blocks into same output**
  - **If an image is encrypted with ECB, we can still make out the image**
  - **We can also run probability analysis if plaintext is English, as it has the index of coincidence of 0.065. (e is the most common character, followed by t so on)**
- **Vulnerabilities**
  - **DROWN**
    - **Attack on SSLv2 protocol (as long as server supports).**
  - **POODLE**
    - **An attack on SSLv3 protocol to decrypt communication**
  - **BEAST**
    - **Takes advantage of a vulnerability in the implementation of the Cipher Block Chaining (CBC) mode in TLS 1.0**
  - **CRIME**
    - **side-channel attack against HTTPS compression. DEFLATE is the most used compression algorithm. The attacker can then inject different characters and then monitor the size of the response.**
  - **BREACH**
    - **Similar to CRIME, but BREACH targets HTTP compression.**
  - **HEARTBLEED**
    - **Heartbleed extension of OpenSSL library is vulnerable to an arbitrary read vulnerability.**
  - **SWEET32**
    - **By capturing large amounts of encrypted traffic between the SSL/TLS server and the client, a remote attacker able to conduct a man-in-the-middle attack could exploit this vulnerability to recover the plaintext data and obtain sensitive information**
  - **SSL Stripping**
    - **Victims arrive on an SSL page via a link from a non-SSL site**
    - **Redirects and downgrades HTTPS to HTTP**

How SSH works

- **SSH identification string exchange**
- **Algorithm negotiation**
- **Key exchange to generate master key (Diffie-Hellman based)**
- **Now switch to symmetric encryption**

How PGP works

- **Encrypt document with secret key**
- **Encrypt secret key with recipient's public key**
- **Send encrypted document and key to recipient**
- **Recipient receives, decrypts encrypted secret key with its private key**
- **Uses decrypted encrypted key to decrypt document**

Buffer overflows

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold or when a program attempts to put data in a memory area past a buffer.

Causes

- **Off by one (Fence post error)**
    - **8 to 11 has a distance of 3 but has 4 numbers in total**
    - **Dangerous when doing iteration (for I=something; i<X; i++)**
- **Dangerous copy function like Memcpy/Strcpy (allocation from big to small buffer size)**

Types

- **Stack overflow**
    - **Overflow the EIP**
    - **JMP/CALL ESP**
- **SEH overflow**
    - **Override SEH with POP POP RET to return to NSEH, then short JMP over SEH**
    - **Continue to shellcode or do egg hunter (32 bytes of magic that searches for a unique pattern and executes shellcode after it)**

Compiler level protections against BOF

ASLR, DEP, CFG, stack canaries

What is a web shell?

- **A web shell is a malicious page deployed on web server within its accessible directories**
- **When interacted with, the payload containing calls to operating system modules will be interpreted server-side, allows for arbitrary interaction with the operating system by the web server on behalf of the attacker**

What is a reverse shell?

- **A reverse shell is where the victim initiates a connection to an attacker-controlled TCP or UDP socket and streams stdin/stdout of a command shell like powershell.exe into the pipe.**
- **This way, we as an attacker can remotely interact with the victim operating system via the pipe.**

Email spoof protection

An entity can add a SPF, DKIM and DMARC DNS TXT record.

A Sender Policy Framework (SPF TXT) record allows an entity to authorize IP addresses that are allowed to send email for the domain. Receiving servers can verify that messages appearing to come from a specific domain are sent from servers allowed by the domain owner.

A Domain Keys Identified Email (DKIM TXT) record adds a public key as a dns record. Messages sent to recipients are signed with the sender's public key. Receiving servers can then use the public key in the DKIM record to verify the sender.

_domainkey	TXT	o=~; r=postmaster@example.com
example.com._domainkey	TXT	k=rsa; t=y; p=GIMfMA0GCSqGSIb3DQEBAQUAA4GNADCBIQKBgQCfVzZoJ6YZph/1oTroL1NhkfHmMgZy uUyNBRVVPkXzQaeZMHMC+S+KxVP7TUPPQYZ6CKSELzqDwjv9Jz10u3zx1eB+Bmqc8cYA2oxZdda3EaJ/LEYtl A1auXxHzY2qaEllToSLrV97ll9F3m4p6V5M6Yho9zxfIfrlTHSECLsrQIDAQBA

A Domain-based Message Authentication, Reporting, and Conformance (DMARC) record tells receiving mail servers what to do if they receive emails from appearing to be from your entity but it does not pass authentication checks. This allows for identification of malicious activities for messages sent from your domain.

Version (v)	**The v tag is required and represents the protocol version. An example is v=DMARC1**
Policy (p)	**The required p tag demonstrates the policy for domain (or requested handling policy). It directs the receiver to report, quarantine, or reject emails that fail authentication checks. Policy options are: 1) None 2) Quarantine or 3) Reject.**
RUA Report Email Address(s) (rua):	**This optional tag is designed for reporting URI(s) for aggregate data. An rua example is rua=mailto:CUSTOMER@for.example.com.**
RUF Report Email Address(s) (ruf):	**Like the rua tag, the ruf designation is an optional tag. It directs addresses to which message-specific forensic information is to be reported (i.e., comma-separated plain-text list of URIs). An ruf example is ruf=mailto:CUSTOMER@for.example.com.**
Failure Reporting Options (fo):	**The FO tag pertains to how forensic reports are created and presented to DMARC users.**
ASPF Tag (aspf):	**The aspf tag represents alignment mode for SPF. An optional tag, aspf=r is a common example of its configuration.**
ADKIM Tag (adkim):	**Similar to aspf, the optional adkim tag is the alignment mode for the DKIM protocol. A sample tag is adkim=r.**
Report Format (rf):	**Forensic reporting format(s) is declared by the DMARC rf tag.**
Report Interval (ri):	**The ri tag corresponds to the aggregate reporting interval and provides DMARC feedback for the outlined criteria.**
Subdomain Policy (sp):	**This tag represents the requested handling policy for subdomains.**

How internal git works?

- **Git is a decentralized Version Control System and a File System.**
- **At its base, Git is a key-value data store. When we upload a file, Git internally store files as a blob object. The contents of the blob are compressed and the SHA1 hash of the content is calculated. This SHA1 hash value is then used a reference to that object. Should duplicate files have same hashes values, the blob is not duplicated in the internal git storage.**
- **Git is also a file system. It stores can folders along with its contents as tree objects which contains Unix like permissions, type of object along with the hash of what it is pointed to, and the filename.**
- **Lastly, as Git is Version Control System, it can store new commits in the .git folder as a Commit Object which contains the hash of a new tree, author info, commit message, and the hash of the previous commit. (merge-commit has two parents).**
- **Git commit is unique identifier generated by Git**
- **If you are pushing a piece of code, how does others know it is yours and not from someone else?**
    - **In Git, the we can set the author field to any value. The only way to prove that you have authored a commit is by attaching a digital signature to it. We can sign commits with a trusted GPG signature (GNU Privacy Guard).**
- **How does GIT ensure integrity of code?**
    - **each Git object *is* the hash ID of the object's content, so anything that uses the ID to look up and read the content can, verify that the hash of the extracted data matches the ID used as a key to extract that data**

Web Penetration Testing

How will you perform a web pentest?

- **Before starting the pentest, I will run through the scope of the application:**
  - **What's the URLs that are in scope**
  - **Is it a black box or grey box approach in which I will receive credentials with different access matrixes such as normal user and admin?**
  - **Are there any exclusions or WIP functions?**
- **After gathering the required information, I will start on the actual penetration test of the application:**
  - **I will first run through the application to get a better understanding of all existing functions of which, which users are able to access.**
  - **After gathering a brief understanding, I will proceed to use automated scanners such as Netsparker and Burpsuite with multiple burp extensions such as Active Scan++, J2EEScan, RetireJS to get an overview of the application and potential low hanging fruits or interesting entry points to further test.**
  - **Ofcourse using two scanners might seem overkill, but in my opinion running through different scanners act as safety net for each other, so when cross-referenced, we can sieve out false-positives that arise from one scanner.**
  - **Will also run directory fuzzing through the web application to see if there are sensitive hidden directories that can be further looked into.**
  - **Next, I will proceed to run through test cases from the WSTG checklist against the application. Such examples would be:**
    - **For information gathering, I would attempt to fingerprint the web server and enumerate for potentially insecure software in use. This could be web server banners or in the case of enumerating of wordpress plugins using WPScan.**
    - **In testing for Authorization, I would attempt to test for Insecure Direct Object References (or IDOR) by manipulating the user identifier to another user's identifier to see if the application is vulnerable**
    - **Or in testing for Input Validation Testing, where I would test for SQL injections.**
    - **Should I encounter potential vulnerabilities within the application, I would look up for it in exploit-db or Github to find a suitable exploit that could be used to create a simple proof-of-concept.**

OSI layers

1) Physical - Physical resources like rj45 cable

2) Data link - Data packaged into frames. At this layer, MAC address is used for addressing and Logical link control (LLC) provides flow and error control

3) Network - Packages frames into packets. This is where IP addressing comes in.

4) Transport - This layer manages sequencing and delivery of data packets. The two transport modes are TCP and UDP.

5) Session - Session layer controls conversation between computers, of how a session is set up, managed and terminated.

6) Presentation - This layer formats data for the application layer, and can also handle encryption and decryption required by application layer.

7) Application - This is the layer where end user interacts directly with the application.

HTTP HTTPS

Are they different protocols?

They are not. HTTPS is SSL/TLS over HTTP, where the unencrypted HTTP protocol uses SSL/TLS to encrypt HTTP requests and response.

XSS

What is it?

Cross-site scripting is a form of injection, where arbitrary script can be injected into application. XSS occurs due to a lack of user input sanitization. Cross-site scripting can be used to perform session hijacking or session riding (CSRF) attacks, and in some severe cases remote code execution (such as macOS teams @mentions functionality in 2020).

There are 3 types of XSS, Reflected, Stored, and DOM-based.

3 sinks of DOM:

- **Document Sink: <img+src+onerror=alert(1)>**
- **Location Sink: javascript:alert(1)**
- **Execution Sink (Eval): alert(1)**

Prevention

Perform proper user input sanitization

Encode data on output via HTML encoding

Use content-security-policy

CSRF

What is it?

Cross-site request forgery is an attack that forces leverage on a victim to perform actions on behalf of you while authenticated. Usual attacks would be to force the victim to change password, email addresses, or if the user is an administrative user, perform privileged actions such as installing an insecure extension to the application.

How do you check for it

I will check if the application only relies on Cookie-based session handling or if CSRF tokens are present. If CSRF token are present, I will check for entropy, randomness, predictability (if it is sequential), and if it is checked on server-side (if I removed the CSRF token will I still be able to perform the action).

To validate the vulnerability, while authenticated I will use burp to capture a request to a relevant functionality i.e. such as change password and generate a CSRF poc. I will amend the request to set the password I will want to set to. If the poc is executed and the password is changed, the application is vulnerable to CSRF.

If you put your JWTs in a header, you don't need to worry about CSRF. You do need to worry about XSS, however. If someone can abuse XSS to steal your JWT via localStorage (it's an array, must use getItem), this person is able to impersonate you

```
<script>alert(localStorage.getItem('ServiceProvider.kdciaasdkfaeanfaegfpe23
.username@company.com.accessToken'))</script>
```

Prevention

A randomized, high entropy, CSRF token needs to be generated and checked server-side.

Access-Control-Allow-Origin: Same origin

JWT (TO BE EXPANDED ON)

Use JOSEPH extension in burpsuite to automate

Checking the validity of a token

Testing for known exploits:

- **(CVE-2015-2951) The alg=none signature-bypass vulnerability**
- **(CVE-2016-10555) The RS/HS256 public key mismatch vulnerability**
- **(CVE-2018-0114) Key injection vulnerability**
- **(CVE-2019-20933/CVE-2020-28637) Blank password vulnerability**
- **(CVE-2020-28042) Null signature vulnerability**

Scanning for misconfigurations or known weaknesses

There is no need to encrypt the token or to include the nonce. The key properties of a CSRF token are that it is not predictable by an attacker, and, unlike cookies, that it is not added to every request by the browser. A cryptographically secure JWT stored in a hidden field meets both of these properties.

Note that you need to use JWT's that have user-unique data in them. Having an empty or generic JWT that isn't specific to the user will not provide you with security.

A JWT, if used without Cookies, negates the need for a CSRF token - BUT! by storing JWT in session/localStorage, your expose your JWT and user's identity if your site has an XSS vulnerability (fairly common). It is better to add a csrfToken key to the JWT and store the JWT in a cookie with secure and http-only attributes set

HTTP Request Smuggling

Happens when the front-end and back-end server interpret the end of a request differently. This happens because HTTP allows for 2 different ways to specify where a request ends via the Content-length and Transfer-Encoding header. For example, the front-end server will use Content-Length header to determine at what length the request is completed whereas the back-end uses Transfer-Encoding and is terminated by placing a chunk size of 0 at the end. We could smuggle a request in by sending a request that has a content length of the body we want to smuggle in i.e., a GET request to a malicious URL, but we also set the Transfer-Encoding header to chunked and set a 0 before the smuggled request.

Should this request be smuggled in, the next victim who performs the request will be redirected to our malicious URL.

To prevent this, we can use HTTP/2 as it has a robust mechanism to determine the length of the requests. Alternatively, we can simply make the frontend server normalize any ambiguous request and force the back-end server to reject any request which are still ambiguous.

XXE

XML External Entity is when XML input containing a reference to an external entity is parsed the application via a declared system identifier. This happens when an application uses standard XML parsers which by default support these features even if they are not normally used by the application.

This can be used to disclose local files and if the PHP expect module is loaded, an RCE may be possible.

```
<?xml  version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM  "file:///dev/random" >]>
<foo>&xxe;</foo>
```

SQLi

Types of SQLi

There are 3 types of SQLi, In-band (Classic), Inferential (Blind) and Out-of-band (xp_dirtree to force database to resolve hostname via DNS).

How will you check for it in the login form?

For starters, I will append a single quote after the username or password field. This is in hopes that I can induce the database to throw error messages. From the errors, I can then infer what sort of database the application is using. I will then proceed to use Union-based queries specific to the database to extract more information out of the database, such as the database version, tables, and if it's MSSQL, will attempt to use xp_cmdshell to gain remote code execution.

If there are no errors thrown, I will then attempt to inject a time based payload like sleep(10). If the webpage takes 10 seconds long to load, the application might be vulnerable to time based SQLi. I will also look out if different inputs causes the webpage to throw different responses. If one page contains a 302 and the other is a 200 OK, the application might be vulnerable to boolean based SQLi.

Lastly if all else fails, I will run the input form through automated tools such as SQLMap to cover what a manual test might not cover.

Parameterized queries enforce a clear distinction between the SQL code and the data passed through parameters**. The way parameterized queries work, is that** the SQL query is precompiled, with placeholders for values to be set. **During execution time, whenever a call to the parameterized query is performed, the fields i.e., username and passwords are merely replacing the placeholders as values. This means the values cannot affect the query. Should there be an attempt of injection, the injection will not be able to affect the query, and will only be substituting the value of the placeholder.**

SSRF

Server-side request forgery is when we can force the server to perform an action or fetch contents on behalf of you. With SSRF, since the request is coming from the server itself, normal access controls are bypassed with full administrative functionality provided.

With SSRF, we can make local calls to the instance metadata and potentially steal credentials or privileged AWS keys like in the Capital One breach in 2019.

IMDSV1 GET request http://169.254.169.254/latest/meta-data

IDMSV2 PUT request to http://169.254.169.254/latest/api/token to retrieve token and a separate curl to the meta-data endpoint with the X-aws-ec2-metadata-token: $TOKEN as a custom header.

Can retrieve

User data - http://169.254.169.254/latest/user-data, a set of commands that can be provided to instance at launch, and executes with root privileges. Might contain exposed credentials.

iam – One of the most sensitive categories is the IAM category, through which the IAM roles associated with an instance, including the token that was assigned to the instance by STS, can be discovered.

Identity-credentials – according to the documentation, these credentials that AWS uses to identify an instance to the rest of the Amazon EC2 infrastructure and are for internal use only.

Deserialization attack

Serialization is a mechanism of converting the state of an object into a byte stream. Deserialization is the reverse process where the byte stream is used to recreate the actual Java object in memory. This is generally a safe practice only if user input is not expected.

However, unsafe deserialization happens when untrusted user input is allowed to be deserialized. We can use tools like ysoserial to generate a serialized object with a payload to be passed into the input.

Identification:

https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html

https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Deserialization_Cheat_Sheet.md

- **White box**
    a. **XMLdecoder with external user defined parameters**
    b. **XStream with fromXML method**
       (xstream version <= v1.46 is vulnerable to the serialization issue)
    c. **ObjectInputStream with readObject**
    d. **Uses of readObject, readObjectNodData, readResolve or readExternal**
    e. **ObjectInputStream.readUnshared**
    f. **Serializable**
- **Black box**
    a. **Java serialized data always begins with the hex pattern ACED or Ro0 in Base64.**
    b. **Content-type header of an HTTP response set to application/x-java-serialized-objec**

Prevention

The only safe way is not to accept any serialized objects from untrusted sources, or use serialization mediums that only permit primitive data types. If it is not possible, we can consider implementation of integrity checks on any serialized objects to prevent serialized objects to prevent dangerous object creations, we should also log deserialization exceptions and failures, if the incoming type is not the expect type.

https://github.com/JoyChou93/java-sec-code/

XXE:
"SAXReader",
"DocumentBuilder",
"XMLStreamReader",
"SAXBuilder",
"SAXParser",
"XMLReader",
"SAXSource",
"TransformerFactory",
"SAXTransformerFactory",
"SchemaFactory",
"Unmarshaller",
"XPathExpression"

JavaObjectDeserialization:

"readObject",
"readUnshared",
"Yaml.load",
"fromXML",
"ObjectMapper.readValue",
"JSON.parseObject"
SSRF:
"HttpClient",
"Socket",
"URL",
"ImageIO",
"HttpURLConnection",
"OkHttpClient"
"SimpleDriverDataSource.getConnection"
"DriverManager.getConnection"
FILE:
"MultipartFile",
"createNewFile",
"FileInputStream"
SPelInjection:
"SpelExpressionParser",
"getValue"
Autobinding:
"@SessionAttributes",
"@ModelAttribute"
URL-Redirect:
"sendRedirect",
"forward",
"setHeader"
EXEC:
"getRuntime.exec",
"ProcessBuilder.start",
"GroovyShell.evaluate"

Click jacking (iFrame)

x-frame-options: deny

x-frame-options: sameorigin

XSSi

Cross-site inclusion is when a victim's sensitive information is included in a malicious page.

If private information is located in a global accessible JS file, we can include the script with private information in our malicious content.

In XSSi, the goal is to leak data cross-origin, such as private keys.

There are four types, Static JavaScript (regular XSSI), Static JavaScript (which is only accessible when authenticated), Dynamic JavaScript Non-JavaScript

SAML AUTHENTICATION

SAML is an open standard SSO for logging users into applications. It allows identity providers (IdP) to pass authorization credentials to service providers (SP), this allows for Single Sign On. SAML uses XML for communications between the IdP and Service Providers.

Identity provider: Provides authentication and passes the user's identity and authorization level to the service provider

Service Provider: Trusts the identity provider and authorizes the user access to the requested resource.

Steps

- **First, the user tries to access an application (which is the service provider).**
- **Then, the application detects the IdP (which may be Okta to authenticate the user. It then generates a SAML AuthnRequest with an AuthnRequest ID, and redirects the client back to the IdP**
- **The IdP authenticates the user and creates the SAMLResponse signed with its digital certificate and posts it to the SP via the user's browser. (InResponseTo the AuthnRequest ID) (Here is where the Golden SAML attack can take place)**
- **The SP then verifies the SAMLresponse signature as a trusted IdP and grants the user access.**

Assertion Attributes

- **ID: Newly generated number for identification**
- **IssueInstant: Timestamp to indicate the time it was generated**
- **AssertionConsumerServiceURL: The SAML URL interface of the service provider, where the Identity provider sends the authentication token.**
- **Issuer: The EntityID (unique identifier) of the service provider**
- **InResponseTo: The ID of the SAML request that this response belongs to**
- **Recipient: The EntityID (unique identifier) of the service provider**

SAML Attacks

We can attempt SAML attacks such as replaying expired messages for another application or tampering the userid to another valid user with elevated privileges.

Golden SAML

If we gain administrative access to the IdP (through lateral movement and Privilege escalation), we can then perform a Golden SAML attack.

Similar to a golden ticket attack in Kerberos, if we obtain the key that signs the object which holds the user's identity and permissions, we can forge an authentication object to impersonate any user to gain privileged access to the SP.

Requirements:

- **The Identity Provider's private key**
- **The Identity Provider's public certificate**
- **The Identity Provider's name**
- **The name of the role to be spoofed (e.g. administrator).**

Impact:

The attacker can control every aspect of the SAMLResponse object, like username, permissions set and validity period. More importantly, this attack is effective even when 2FA is enabled.

- **They can be generated from practically anywhere. You don't need to be a part of a domain, federation of any other environment you're dealing with**
- **They are effective even when 2FA is enabled**
- **The token-signing private key is not renewed automatically**
- **Changing a user's password won't affect the generated SAML**

Detection:

Renew IdP certificate and its private key at regular intervals, or when there is doubt relating its confidentiality. Sufficient logging will allow for detection of trust modifications.

IMDS (Instance metadata)

https://blog.checkpoint.com/2020/12/07/aws-instance-metadata-service-imds-best-practices/

The IMDS endpoint holds metadata regarding the instance itself and is only accessible from the machine itself. (via http://169.254.169.254/). As an attacker, are interested in credentials, like AWS, IAM and identity credentials.

We can also look at the IDMS user data, which exposes the set of commands provided to an instance at launch time (when instance is launch, the data script is executed with root privileges). We would then be able to look out for any plaintext credentials.

iam – One of the most sensitive categories is the IAM category, through which the IAM roles associated with an instance, including the token that was assigned to the instance by STS, can be discovered.

Identity-credentials – according to the documentation, these credentials that AWS uses to identify an instance to the rest of the Amazon EC2 infrastructure and are for internal use only.

Remediation

Use AWS Instance metadata service (IMDS) Version 2 instead of Version 1(only 30% using version 2).

1. **Uses a token generated by the PUT request to http://169.254.169.254/latest/api/token**
2. **A PUT request to http://169.254.169.254/latest/api/token with the custom header x-aws-ec2-metadata-token-ttl-seconds with the value of the number of seconds for which the token needs to be active. This PUT request generates the token to be used in Step 1.**
3. **This update fixes all vanilla SSRF where the attacker can only control the URL**

For SSRF to succeed, an attacker would need to control the HTTP method (force server to make a PUT instead of standard GET) and pass custom headers to the server, which the server then will use to make the request.

Thick Client

Yes, I have. With thick client applications (2 tier or 3 tier),

- **I usually look through config files for plaintext credentials,**
- **traffic inspection with wireshark and echo mirage,**
- **inspection of the portable executable and its modules if it was compiled with DEP/ASLR/Control Flow Guard (CFG),**
- **static and dynamic analysis,**
- **and general windows specific test cases such as weak service permissions, unquoted service paths, and dll hijacking**

Mobile

- **As a framework, I will use MobSF to run through the application. The tool will disassemble the APK to Smali and decompile it into a Java Source file**
- **For static analysis, I'll be performing code review to grep out for credentials, internal IP addresses, and SQL queries.**
- **I'll also check if the application performs integrity checks. I will modify the source code and attempt to compile it back into an APK. If the application still runs, it would mean that no integrity checks are performed to check if modifications to the application are performed.**
- **For traffic analysis, if the application has SSL pinning, I'll attempt to bypass it using Frida-objection. Should this method fail, I'll request for a non-SSL pinned application for testing. Then, I'll hook up the android application to run through burpsuite as a proxy to inspect the traffic.**

Red team

Tell me about your red teaming experiences

1) **In my previous red teaming engagements, it has mainly been against conventional windows AD enabled environments. We used cobalt strike configured with both HTTP and DNS channels along with domain fronting with AzureCDN**
2) **For weaponization, we used both Office macros and executables.**
3) **For evasion, we came up with means to evade detection by encrypting our VBA payload with environment variables such as USERDNSDOMAIN and opted for the safer Task Service object instead of dangerous execution methods such as WMI. Removed XOR and rewritten the function with its NAND equivalence.**
4) **As a result, we were able to successfully bypass both the email sandboxing and the Antivirus solution on the victim's endpoint**
5) **For binaries, we injected raw shellcode into it with Go4arun or Shellter pro**
6) **Unfortunately, for the said engagement we purchased our domains a little too late, so our callback domains were placed on newly observed domains and were blocked by the web proxy. Hence, we were unable to receive a callback**

Mitre Attack is a curated knowledge base that tracks cyber adversary tactics and techniques:

1. **Initial Access**
2. **Execution**
3. **Persistence**
4. **Privilege Escalation**
5. **Defense Evasion**
6. **Credential Access**
7. **Discovery**
8. **Lateral Movement**
9. **Collection**
10. **Exfiltration**
11. **Command and Control**

APTs

1) **Dark Pink**
    a. **Powershell**
    b. **Link shortener > download iso**
2) **Stone panda**
3) **APT41 Double dragon**
    a. **Chinese state sponsored**
    b. **Conducts financially motivated activities**
        i. **Chinachop webshell**
        ii. **Stolen credentials**
        iii. **Spear phishing with compiled html attachments for initial access**
        iv. **WMIexec for lateral movement**
        v. **Deploys rootkits on Linux systems and Master boot records bootkits**
        vi. **Use of legitimate websites such as github, pastebin to avoid detection**
        vii. **Before deploying Ransomware as a service (Raas), blocked victim systems from retrieving AV updates by accessing DNS management console and implemented a forward lookup on the domain used for anti-virus updates.**

Threat intelligence/modelling

1) **Threat intel feeds**
    a. **Open source**
        i. **Mandiant advantage**
        ii. **Singcert**
        iii. **Cisco Talos Intelligence**
    b. **Close source**
2) **Ryuk ransomware**
    a. **Send phishing email with malicious link**
    b. **Bait users to execute with dual extensions**
    c. **Use zerologon to reset domain controller password**
    d. **Dump hashes**
    e. **Pivot with WMI exec**
    f. **Disable endpoint protections**
    g. **Exfiltrate and encrypt**

Technical questions

Recon

1) **Code injection**
    a. **Obtain handle to process/CreateProcess**
    b. **Allocate memory in remote process**
        i. **VirtualAllocEx -> WriteProcessMemory**
        ii. **CreateFileMapping -> MapViewOfFile -> NtMapViewOfSection**
    c. **Code Execution**
        i. **CreateThread**
2) **Process Hollowing**
    a. **Create a suspended process (svchost)**
    b. **Unmap/Hollowing out memory**
        i. **ZwUnmapViewOfSection**
    c. **Replace memory section with Arbitrary code**
        i. **VirtualAllocEx**
        ii. **WriteProcessMemory**
    d. **Unsuspend process**
        i. **Resumethread**
3) **Threat Intelligence**
    a. **Fireeye threat reports**
    b. **Mandiant Threat Intelligence**
    c. **Cisco talos**
4) **Passive**
    a. **Rengine (OSINT framework)**
    b. **Shodan port scan (Shoscan)**
    c. **DNSDB for subdomains**
    d. **Certificates**
    e. **theHarvester to harvest email address and verify with a Microsoft login portal**
    f. **Look out for any UAT or orphaned application with login pages that might be a good entry point**
5) **Active**
    a. **Sweep through the network for unusual ports**

b. **Run a quick scan with masscan to assess if it's worthwhile to attack the application**

Resource Development

6) **Infrastructure automation**
    a. **Stand up our redirectors, long haul short haul c2 servers, smtp server w Terraform/Ansible**
    b. **Decide on supported channels**
        i. **DNS**
        ii. **HTTPS**
        iii. **Hybrid**
    c. **Domain fronting (findfrontabledomains)**
        i. **SNI (outside) vs Host header (Encapsulated via TLS)**
        ii. **If the CDN checks for discrepancies, we can use domainless frontless which we use a blank SNI field instead to bypass it**
        iii. **AzureCDN is no longer useful, we can use Firebase instead**
7) **Weaponization**
    a. **Craft and test payloads in a payload lab against hardened machines and various endpoint and network protection products (i.e., Antivirus, mail sandboxes, and web proxy)**
    b. **The delivery method can be in a few flavors such as office macros (with WinHttpRequest), or Dynamic Data Exchange (DDE), or executables disguised as installers (stomped, ditto)**
    c. **We will break ou payload into 2 stages**
    d. **Our first stage would be to set up a scheduled task via a task service VBA object on the victim which will lay dormant for a set duration. This will achieve both execution and persistence on the target.**
    e. **In the second stage, the scheduled task will then retrieve our payloads from our dropper and execute. (We can download a benign binary and sideload with our malicious DLL)**
    f. **For evasion, we can encrypt the macro with an environment variable like USERDNSDOMAIN and remove the MZ header from our second stage binaries**
    g. **This will ensure that the macro survives heuristic checks, as a sandbox would likely not be configured with the same environment variables as an internal network**

Evasion

1) **Patching userland hooks**
    a. **Kernel32, User32 vs NTDLL.dll**
    b. **Corrupt the first few bytes of the hook**
    c. **Or patch the entrypoint**
2) **Avoid WinAPI, use direct system calls insteads**
    a. **NtMapViewofSection**
    b. **VirtualAlloc, RtlMoveMemory, CreateThread (caught)**
3) **Stomping the headers**
4) **Removing magic bytes from the file and adding it later**
5) **Encryption stub**
6) **Obfuscate the code with ConfuseEx**
7) **Replacing XOR with safer logical equivalences such a NAND boolean function (A NAND (A NAND B)) NAND (B NAND (A NAND B))**
8) **Opt for a fileless approach, like in memory execution (in windows it would be dot net execute-assembly)**

Initial access

1) **Pre-phishing**
    a. **Send a test email to a few emails to check for out-of-office replies and SMTP headers indicative of underlying defense products**
2) **Phishing**
    a. **Craft a phish revolving around a set theme like COVID or an Urgent application update to coerce users into clicking whilst internally connected via a VPN**
    b. **We will either embed download links to our binary or attach a Microsoft office macro embedded documents like xls or doc**
3) **Low hanging fruits on public surfaces like UAT or orphaned applications in attempts to gain access**
4) **Once initial access has been established, we will lace persistence on the target to our long haul c2 servers.**
5) **Migrate to other processes with process injection (CreateRemoteThread might trigger event 8)**
    a. **CreateThread**
    b. **SetThreadContext**
    c. **NtQueueApcThread-s**
    d. **NtQueueApcThread**
    e. **CreateRemoteThread**
    f. **RtlCreateUserThread**
6) **screenshots**

Discovery

1) Smash and grab approach on our first beacon
2) Gain situation awareness
    a. Seatbelt to find interesting files and potential avenues for exploit
    b. Dump the Outlook offline address book
    c. Dump the AD with Adexplorer
    d. Bloodhound
        i. GenericAll - full rights to the object (add users to a group or reset user's password)
        ii. GenericWrite - update object's attributes (i.e logon script)
        iii. WriteOwner - change object owner to attacker controlled user take over the object
        iv. WriteDACL - modify object's ACEs and give attacker full control right over the object
        v. AllExtendedRights - ability to add user to a group or reset password
        vi. ForceChangePassword - ability to change user's password
        vii. Self (Self-Membership) - ability to add yourself to a group
        viii. Unconstrained Delegation
    e. Rubeus to roast for service tickets, which we can later crack offline

Execution

1) We can use fork and run techniques
2) C# Execute assembly
3) Beacon Object Files (BOF)
4) Sideloading DLLs
5) Windows Management Instrumentation (WMI)
6) Avoid userland hooks with direct syscalls

Privilege Escalation

1) Windows Exploits
2) Cracking kerberoasted hashes
3) Weak service permissions
4) Unquoted service paths
5) Vulnerable drivers
6) AlwaysInstallElevated

Lateral movement

1) Winrm
2) Psexec
3) Wmi

Interesting exploits

1) Zerologon (reset DC password)
2) HiveNightmare (Dump sam from registry)
3) PrintNightmare (Authenticated user, LPE RCE)
4) MS Exchange RCEs like ProxyShell by OrangeTsai
    a. SSRF auth bypass
    b. Post auth file write into webshell
    c. Deserialization and escalate into SYSTEM

Initial Access

- **Evilginx for mitm phishing login credentials along with session cookies to allow 2fa bypass**