Bypassing MFA with the Pass-the-Cookie Attack

blog.netwrix.com/2022/11/29/bypassing-mfa-with-pass-the-cookie-attack

Multi-factor authentication (MFA) is a great way to increase both on-premises and cloud security. With MFA in place, when a user logs on, they are required to provide not only their user ID and password but another authentication factor, such as a code sent to their phone. This process reduces the attack surface by preventing adversaries with stolen user credentials from logging on.

However, MFA is not a cybersecurity panacea. In particular, adversaries can use the Pass-the-Cookie attack to use browser cookies to get around MFA and gain access to cloud services. In this post, we will explore how this attack works and the best security practices for defending against it.

Handpicked related content:

Active Directory Security Best Practices

How Pass-the-Cookie Attacks Work

Browser cookies enable web applications to store user authentication information, so a user can stay signed in instead of having to supply their username and password every time they navigate to a new page on a website. (This is similar to Kerberos or NTLM authentication, in which an artifact is stored locally on the user's system and used for future authentications.)

If MFA is enabled, the user has to provide additional proof of their identity, such as by accepting a push notification on their mobile device. Once the user has passed MFA, a browser cookie is created and stored for your web session.

While cookies simplify the user experience, they carry an obvious vulnerability: If somebody were able to extract the right browser cookies, they could authenticate as another user in a totally separate web browser session on another system. In short, they could use the cookie to bypass authentication via MFA. (This is analogous to the Pass the Hash attack in Active Directory.)

How an Adversary Can Extract Browser Cookies.

Let's see how an attacker could extract cookies using the example of the Google Chrome browser. Chrome stores cookies in the following location in a SQLite database:

```
%localappdata%GoogleChromeUser DataDefaultCookies
```

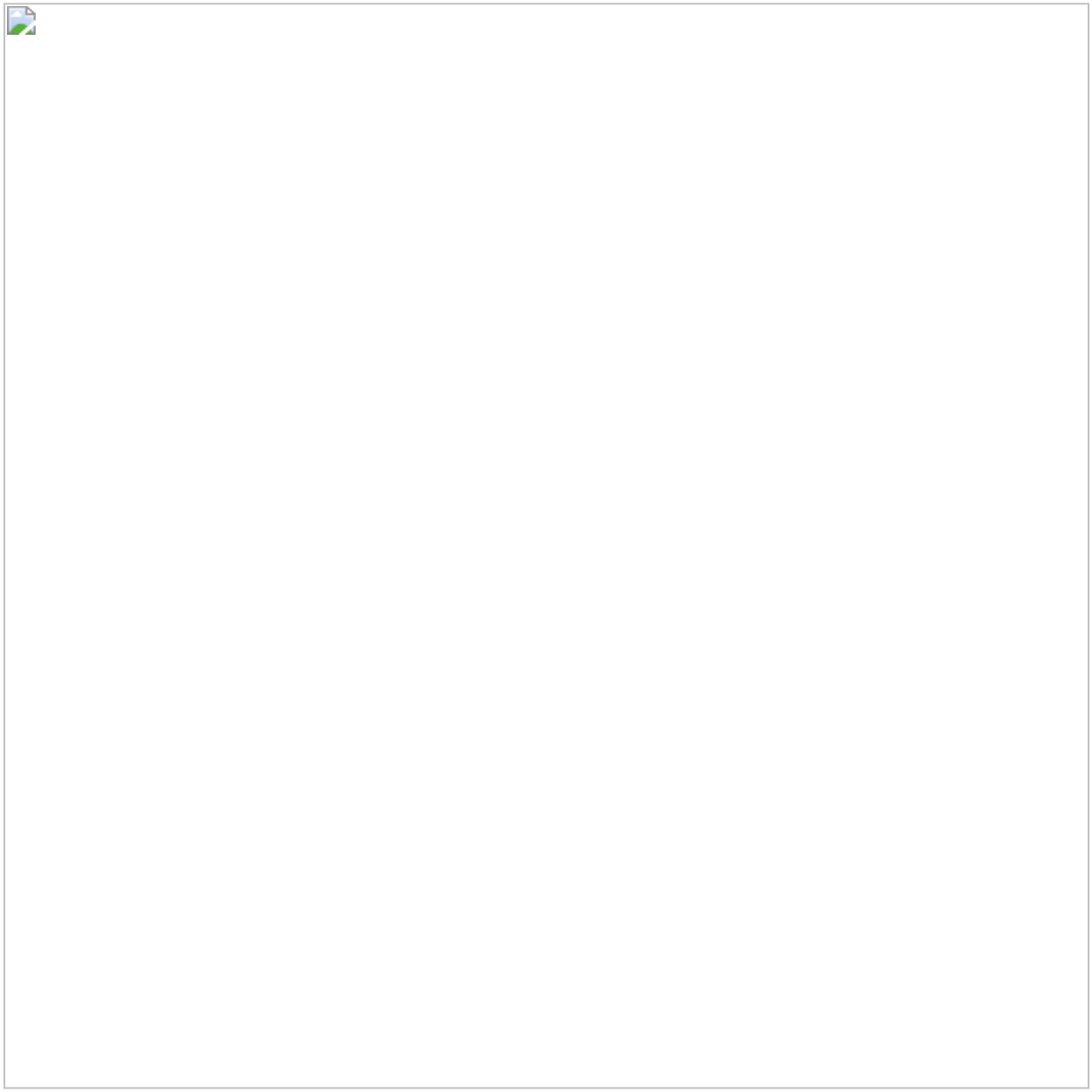
The cookies for a given user are encrypted using keys tied to that user via the Microsoft Data Protection API (DPAPI). To access the cookie database and decrypt the cookies, an adversary can use the following mimikatz command:

```
dpapi::chrome /in:"%localappdata%GoogleChromeUser DataDefaultCookies" /unprotect
```

Alternatively, they could execute the following from the command line:

```
mimikatz.exe privilege::debug log "dpapi::chrome
/in:%localappdata%googlechromeUSERDA~1defaultcookies /unprotect" exit
```

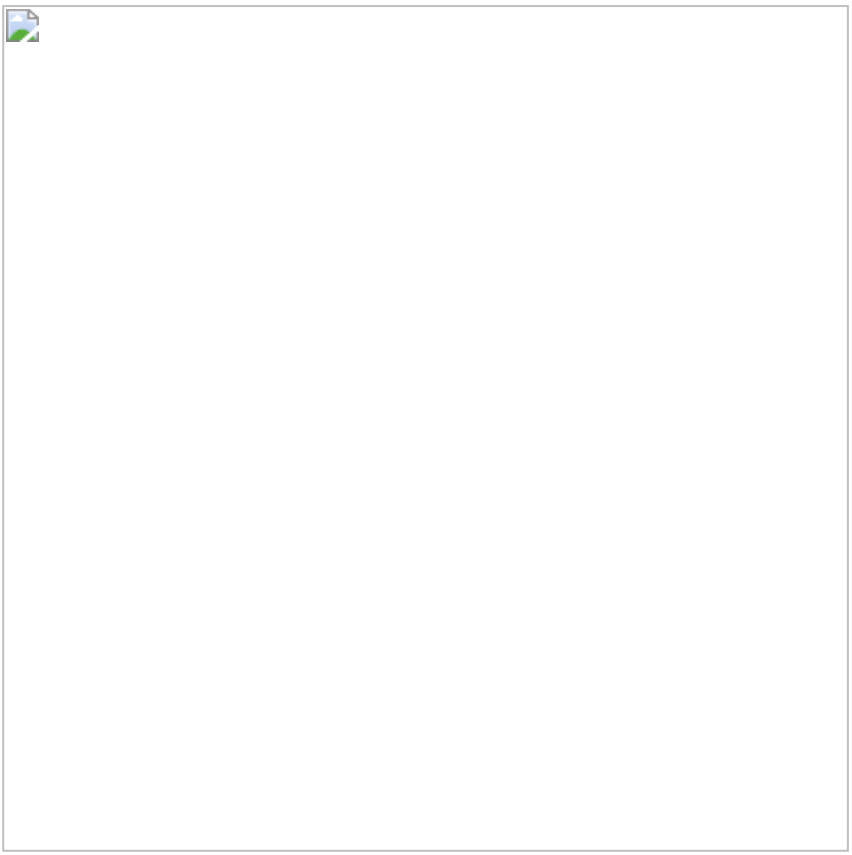
Using either of these options will provide the browser cookies:



Pass-the-Cookie Attack Scenario

Let’s walk through how a Pass-the-Cookie attack would work in a real-world scenario.

Suppose that the user Tobias is an IT administrator. One of the web applications that Tobias uses regularly is the Microsoft Azure management portal. Since MFA is enabled, when Tobias logs into Azure, he has to provide a code from the authenticator app on his mobile device, as shown below.



So, as long as nobody steals his iPhone, his Azure credentials should be safe, right? Not so fast. Suppose Tobias has clicked on a phishing email or his system has been compromised by some other means, and now an attacker is able to execute code within Tobias's user context. Tobias is NOT an administrator on his laptop so the damage should be contained, right?

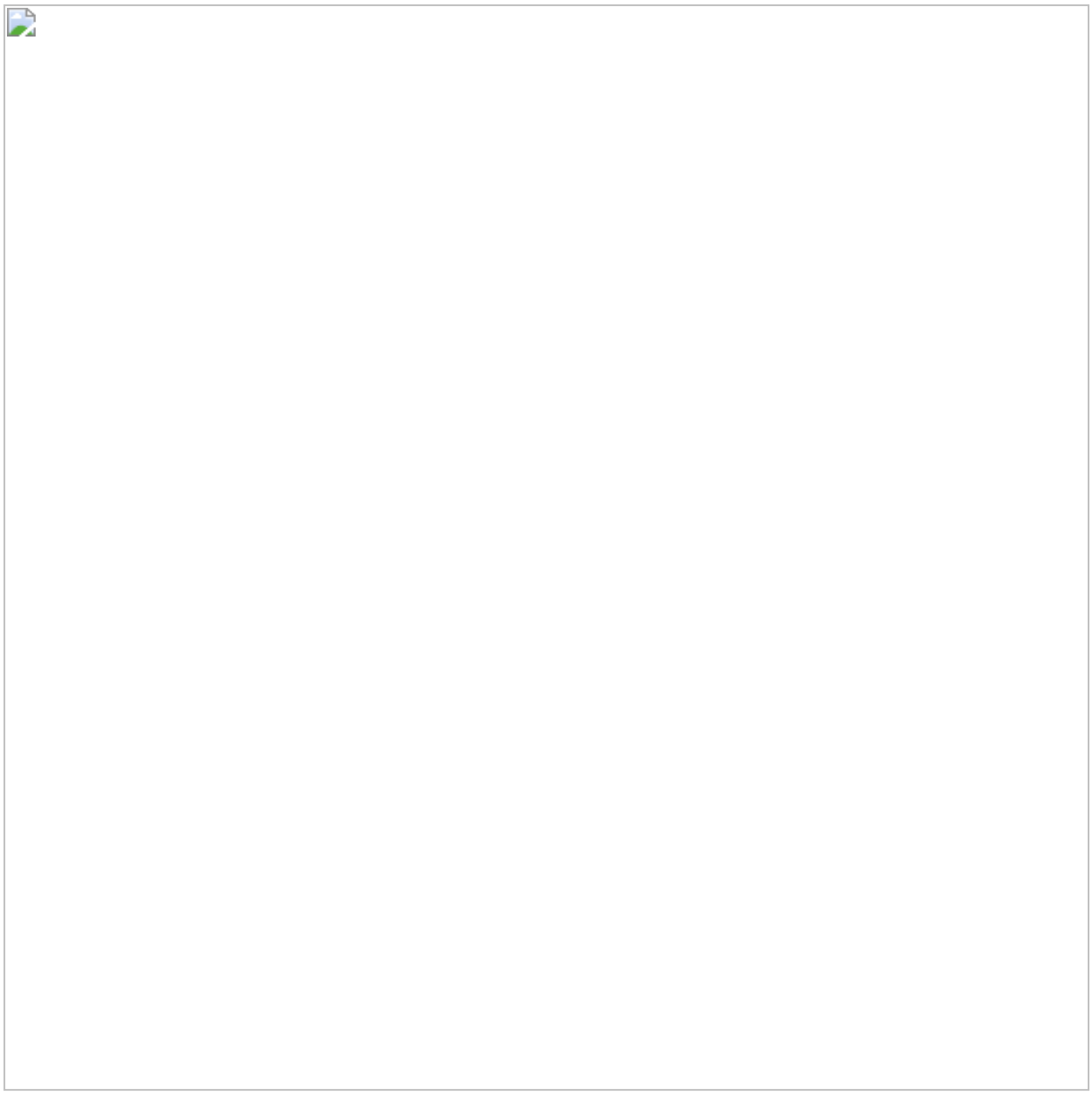
Let's see.

Step 1. Extract the Cookies.

As we saw earlier, all we have to do to get Tobias's browser cookies is execute this command when running as Tobias:

```
mimikatz.exe privilege::debug log "dpapi::chrome
/in:%localappdata%googlechromeUSERDA~1defaultcookies /unprotect" exit
```

In this case, we care about the Azure authentication cookies, including ESTSAUTH, ESTSAUTHPERSISTENT and ESTSAUTHLIGHT. These cookies are there for the taking because Tobias has been active on Azure lately:

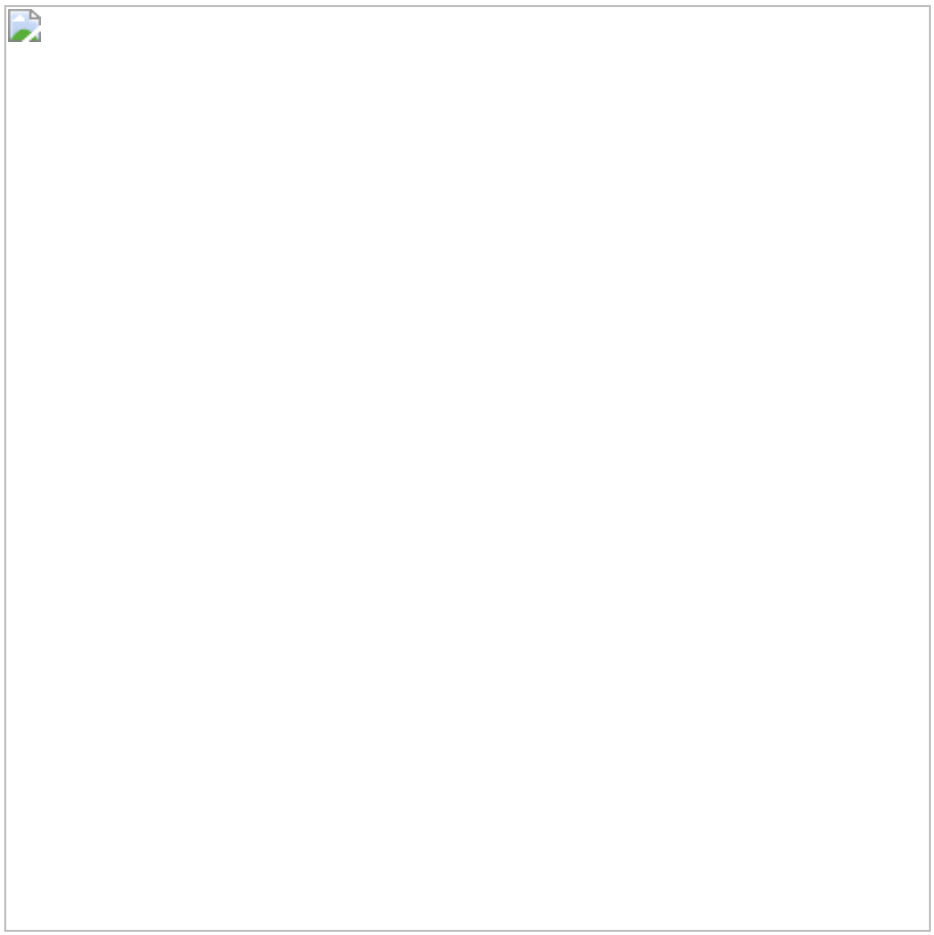


Step 2. Pass the Cookies.

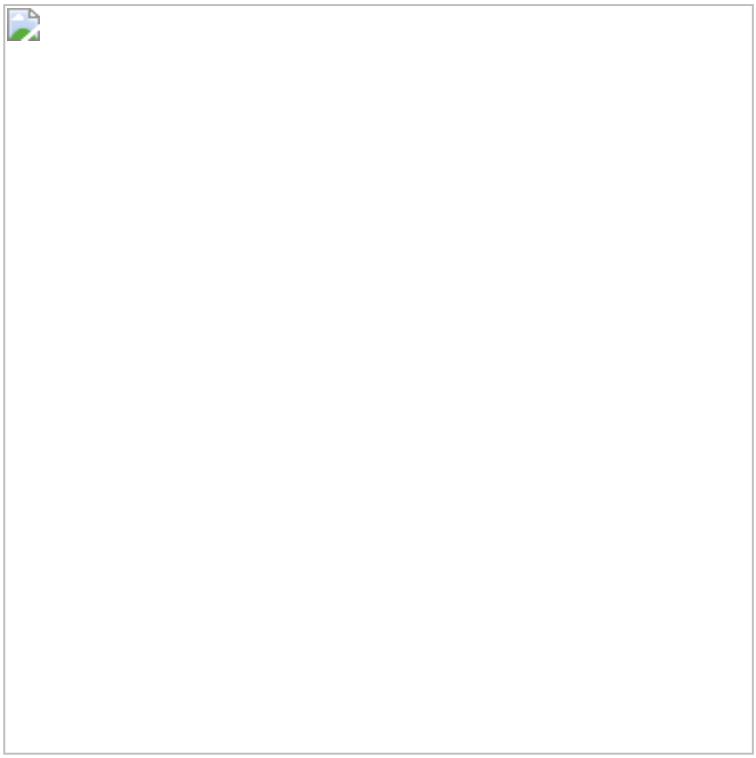
It might seem that since we don’t know Tobias’s user ID or password and we don’t have access to his mobile device, we cannot log into web apps like Azure as Tobias.

But since we have his cookies, we just need to pass them into another session to take over Tobias’s account. This is easy enough to do: We simply open Chrome on another server and use the “Inspect” interface to insert a cookie.

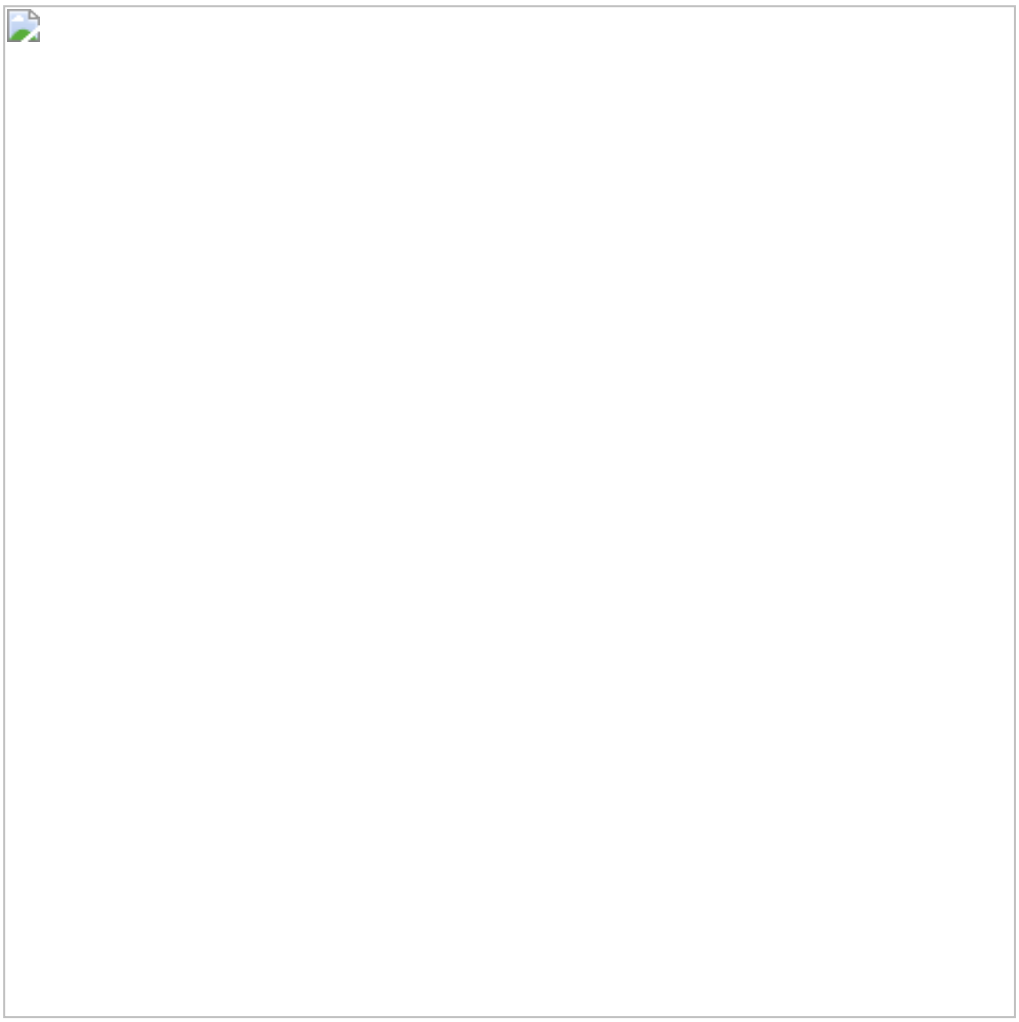
2.1. First, we inspect the stolen session:



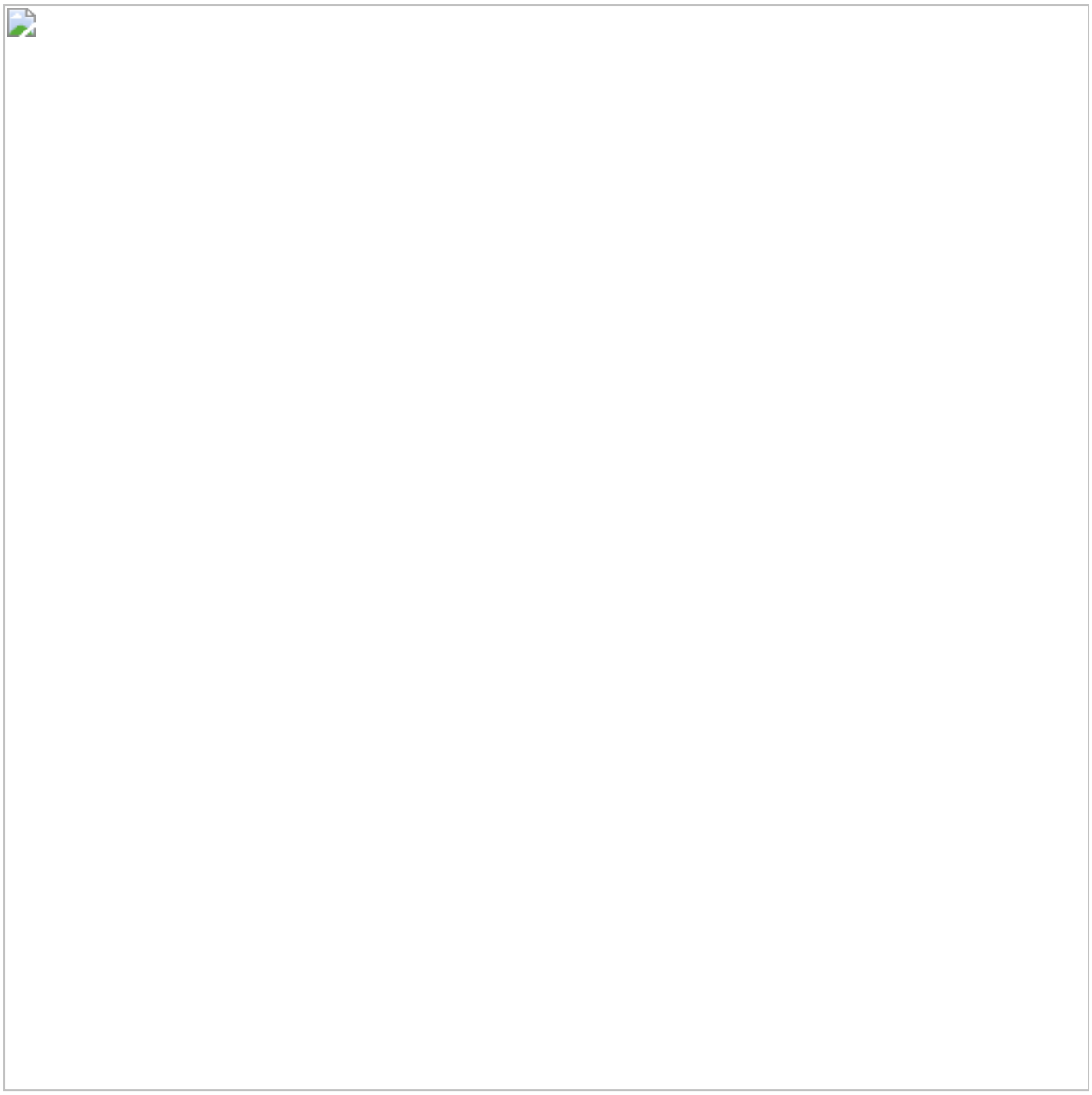
2.2. Next, we navigate to **Application > Cookies**. As you can see, the current cookies do not include the ESTSAUTH or ESTSAUTHPERSISTENT”:



2.3. We add the ESTSAUTH or If ESTSAUTHPERSISTENT cookie. (If ESTSAUTHPERSISTENT is available, it is preferred because it is generated by the “Stay Signed In” option.)



2.4. We refresh the page and now we have logged into Azure as Tobias — no MFA required!



Mitigating Your Risk

Pass-the-Cookie attacks are a serious threat for a few reasons. First, a Pass-the-Cookie attack does not require administrative rights; all users have access to read and decrypt their own browser cookies, regardless of whether they have privileged rights on their workstations. Second, the attacker doesn't have to know the compromised account's user ID or password, so this attack is possible with minimal information. Third, we were even able to complete Pass-the Cookie attacks after the browser had been closed.

One way to minimize the risk of cookie theft is to clear users' cookies more often. However, this will force users to re-authenticate each time they navigate to a site, which will frustrate them and cause them to avoid ever closing their browsers to avoid losing their cookies.

A better strategy is to implement authentication monitoring and threat detection products of these. Netwrix StealthDEFEND can promptly detect accounts that are being used in unexpected ways so you can quickly shut down malicious activity.

FAQ

What is a Pass-the-Cookie attack?

In a Pass-the-Cookie attack, cyber criminals use stolen web session cookies to impersonate legitimate user in order to access data and systems in the victim's on-premises or cloud environment.

How are cookies used?

Web applications use browser cookies to store user settings and authentication information.

Why do hackers steal cookies?

Using stolen cookies, hackers can impersonate a legitimate user session. By gaining access to the user's network infrastructure, the adversary can steal sensitive data, plant malware and exploit vulnerabilities.

Why is MFA important?

Multifactor authentication (MFA) is a security measure that enhances system and data security by requiring the user to provide an additional verification factors in addition to their credentials, such as a code sent to their mobile devices or dedicated access tokens.

Jeff Warren