ADAPTIVE SHIELD

A Guide to Shift Away from Legacy Authentication Protocols in Microsoft 365

Contents

Introduction

Microsoft 365 (M365), formerly called Office 365 (O365), is Microsoft's cloud strategy flagship product with major changes ahead, such as the deprecation of their legacy authentication protocols.

Often stored on or saved to the device, Basic Authentication protocols rely on sending usernames and passwords with every request, increasing the risk of attackers capturing users' credentials, particularly if not TLS protected. Basic Authentication, while necessary for companies using legacy software, is unable to enforce MFA and is superseded by Modern Authentication.

The legacy settings have been on Microsoft's radar to fix for years. In 2018, Microsoft announced it would introduce a series of changes — and ultimately deprecation — to its authentication controls as a means to help organizations mitigate the risk. These changes were set to take place over a number of years, and in September 2021, they announced that they will begin to permanently disable Basic Auth in all tenants, regardless of usage, with the exception of SMTP Auth by late 2022. In order to enable a smooth transition from these legacy protocols to a modern environment, we have created this step-by-step guide to help you reduce risk and reinforce your organization's M365 security.

Adaptive Shield has developed and released a PowerShell script that creates a unified report to map out the organization's posture to know which users have legacy protocols enabled — for you to copy-paste. This guide covers the discovery techniques and blocking access processes while showing you how to handle special exclusions.

List of Basic Authentication Protocols

To secure the organization's deployment, the first step is knowing what types of basic authentication protocols exist. Within Microsoft, the considered basic/legacy protocols include:

1/ Authenticated SMTP
Used by POP and IMAP clients to send email messages

2/ Autodiscover
Used by Outlook and EAS clients to find and connect to mailboxes in Exchange Online

3/ Exchange ActiveSync (EAS)
Used to connect to mailboxes in Exchange Online

4/ Exchange Online PowerShell
Used to connect to Exchange Online with remote PowerShell

5/ Exchange Web Services
A programming interface that's used by Outlook, Outlook for Mac, and third-party apps

6/ IMAP
Used by IMAP email clients, allowing users to access email from anywhere and any device

7/ OAB (Offline Address Book)
A copy of address list collections that are downloaded and used by Outlook

8/ Outlook Service
Used by the Mail and Calendar app for Windows 10

9/ POP3
Used by POP email clients to download new messages and delete them from the email serverReporting Web Services

10/ Other clients
Any other protocols identified as utilizing legacy authentication

These authentication protocols do not support modern authentication mechanisms like multi-factor authentication (MFA), which means that enabling MFA won't suffice.

To enhance security and mitigate risk, organizations must find all the users and services that use the legacy protocols, migrate to use modern protocols, and block the basic ones. This whitepaper will take you through the discovery and blocking process, in addition to sharing instructions for additional controls, like Mailbox services and Conditional Access policies, that can reinforce your Microsoft 365 security posture.

Discovery: Know Your Posture

Before shutting down all legacy protocols within the organization, it is important to identify users and services that are using basic authentication. Rather than reduce productivity and generate user frustration, it is important to let users know that the system is being upgraded, which will help avoid business interruptions and promote a painless transition to modern protocols.

The are a few ways to learn about your organization's posture using these methods:

PowerShell script
Shows which users have the exchange legacy protocols enabled

Conditional Access Report
Shows which users have the exchange legacy protocols enabled

Azure AD Sign-In Logs
Shows sign-ins performed with legacy authentication clients

PowerShell Script

Running the PowerShell script acts as a good starting point to map out the user and service landscape that needs to be mitigated.

After running a few PowerShell cmdlets, the Adaptive Shield team created this PowerShell script (See Appendix A or download here on GitHub) to merge them all into one unified report. The script generates a file: BasicProtocolsReport.csv. This file will show users and their legacy protocol statuses. Each protocols' status is tested against Authentication Policy, Mailbox services, and Transport config. Below is the list of the full payload:

1/ **user**

2/ **has_mailbox** Indicates if the user has a mailbox licensed

3/ **blocked** Account status (enabled/ disabled)

4/ **mfa** Multi Factor Authentication enrollment status

5/ **auth_policy** Name of effective authentication policy (if set)

6/ **is_ap_def** Indicates whether the effective authentication policy is an organization default or specifically assigned to the user

7/ **protocol columns** (activesync, imap, mapi, pop, smtp, outlookservice, PowerShell, ExchangeWebServices, autodiscover, OfflineAddressBook, rpc, ReportingWebServices) - Status (TRUE - enabled; FALSE - blocked)

8/ **protocl_method columns** (activesync, imap, mapi, pop, smtp, outlookservice) - Each of these protocols can be blocked using mailbox services settings, authentication policy, and transport config (global settings for SMTP) this column's details which methods are in place to block these protocols.

Conditional Access — Report-Only

Create a report with Conditional Access (see figure 1) which simulates the users and services that would be affected if you were to block basic authentication protocols. This report gives you visibility into the users and services actually using the legacy protocols.

Suggested run time for this report is three months, over a business quarter, to catch any idle users, and sporadic or time-scheduled services.

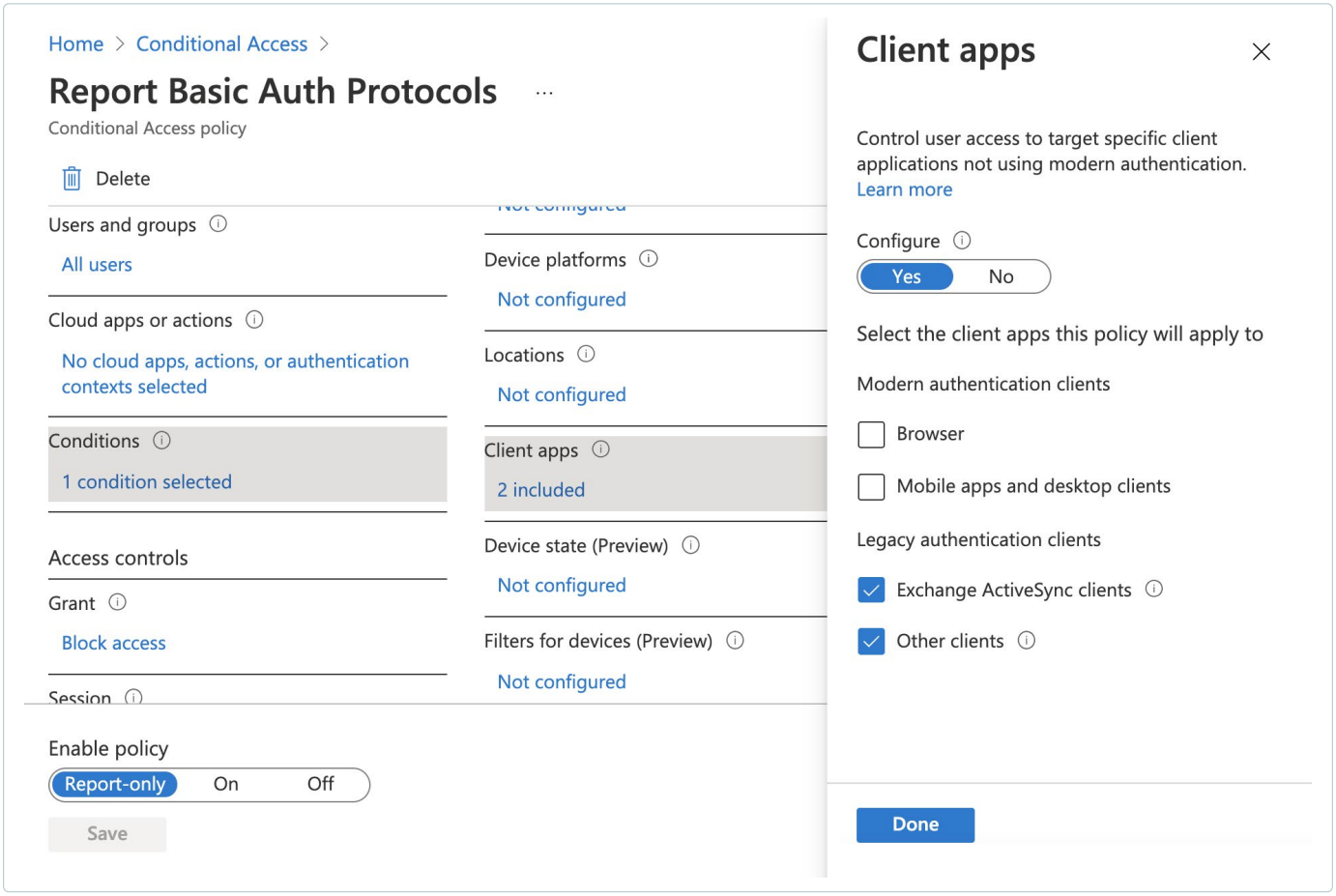

Figure 1: Generate a user and services report over a 3-month period

Reviewing the report and cross referencing it with the PowerShell script results will help you to have a better picture of legacy protocols in use, lowering the possibility of missing services or users that still have basic authentication protocols in play.

Azure AD Sign-In Logs

The Azure AD sign-in logs are another useful way to know your posture. Diving into the logs and filtering "Client app" can reveal sign-ins performed with legacy authentication clients.

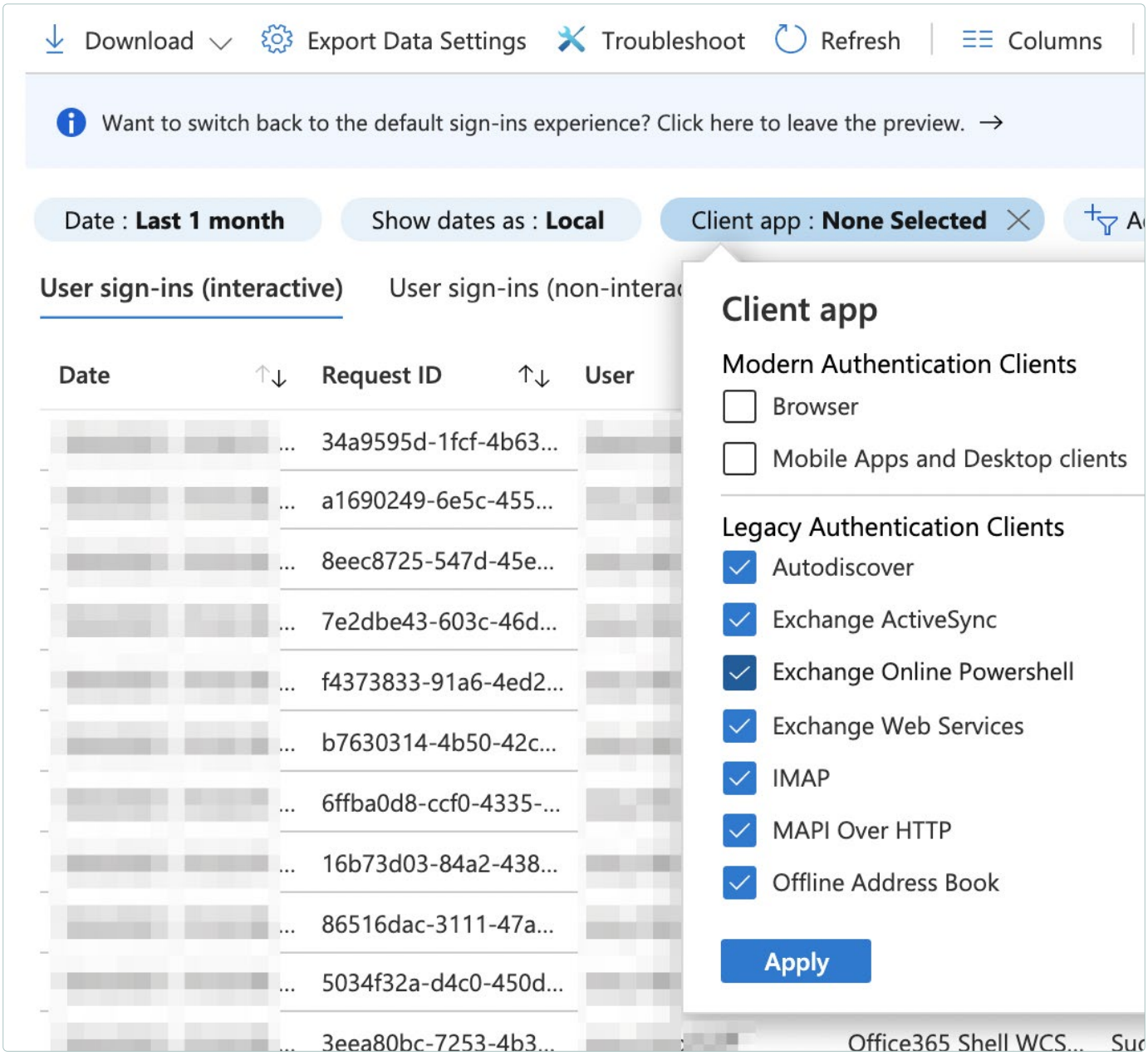

Figure 2: Reveal sign-ins performed with legacy authentication clients

Reviewing the report and cross referencing it with the PowerShell script results will help you to have a better picture of legacy protocols in use, lowering the possibility of missing services or users that still have basic authentication protocols in play.

Blocking Access

After carefully investigating and discovering all of the usage of basic authentication protocols, it is time to block them.

There are a few well-known approaches to blocking authentication protocols, a popular one being using the Conditional Access policies.

However, there are drawbacks to using Conditional Access as the first line of defense. Conditional Access policies are processed post first-factor authentication. This means that the credentials can be compromised (as feedback will still be provided to the client, an advantage in a brute force attack for instance), so the mailbox might not have been breached but the attacker can try the validated password on other systems.

Conditional Access — Report-Only

Start at the source. Microsoft has a dedicated feature for blocking basic authentication protocols, making it easy to control using the Admin console.

Go to the Office Admin center -> Settings -> Org Settings -> Modern authentication and uncheck all of the basic authentication protocols (make sure that modern authentication is checked). See Figure 3.

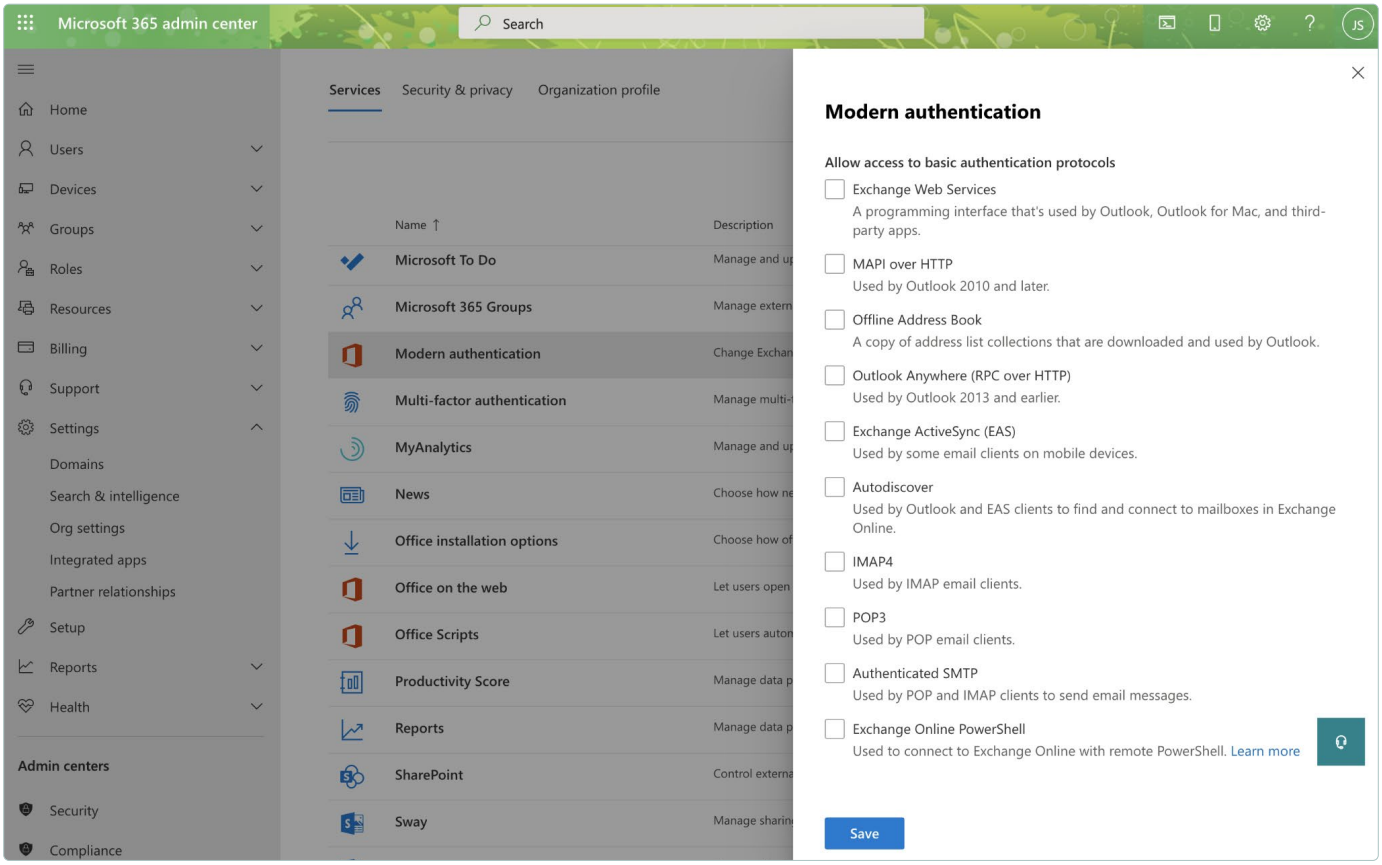

Figure 3: All basic authentication protocols are unchecked

Changing settings in the admin center creates a new authentication policy and sets it as the organization's default policy.

Use PowerShell to validate:

```
$default_policy = Get-OrganizationConfig | Select DefaultAuthenticationPolicy;
Get-AuthenticationPolicy $default_policy.DefaultAuthenticationPolicy;
```

```
RunspaceId                          : c363c75b-b425-4c4b-b7c8-e4ffb79d719c
AllowBasicAuthActiveSync            : False
AllowBasicAuthAutodiscover          : False
AllowBasicAuthImap                  : False
AllowBasicAuthMapi                  : False
AllowBasicAuthOfflineAddressBook    : False
AllowBasicAuthOutlookService        : False
AllowBasicAuthPop                   : False
AllowBasicAuthReportingWebServices  : False
AllowBasicAuthRest                  : False
AllowBasicAuthRpc                   : False
AllowBasicAuthSmtp                  : False
AllowBasicAuthWebServices           : False
AllowBasicAuthPowershell            : False
AdminDisplayName                    :
ExchangeVersion                     : 0.20 (15.0.0.0)
Name                                : Block Basic Auth
```

This example creates a new authentication policy named Engineering Group that allows basic authentication with IMAP and assigns it to a user.

Authentication policies are a must but not enough to stop the threat risk of these legacy protocols alone. The authentication policy covers legacy clients, mailbox protocols such as IMAP and SMTP, and other clients such as PowerShell. However, like Conditional Access, even though the service is blocked, some clients will still provide feedback (allowing certain cyber attacks to succeed in gleaning a password for application in other SaaS apps). To avoid this incriminating feedback, completely turn off the service.

Shutting down a service can only be done for mailboxes, which covers six protocols out of the 13. Blocking the authentication policy covers the rest.

Mailbox Services and Transport Config

Disabling a mailbox service (or enabling in case of exclusion) can be done using the UI per user.

Go to the Office Admin center -> Users -> Active users -> select a user (with mailbox) -> Mail tab -> Manage email apps and uncheck the basic authentication protocols: POP, IMAP, SMTP. See figure 4.

Note that SMTP, MAPI over HTTP, and Mobile (Exchange ActiveSync) support both basic and modern authentication.

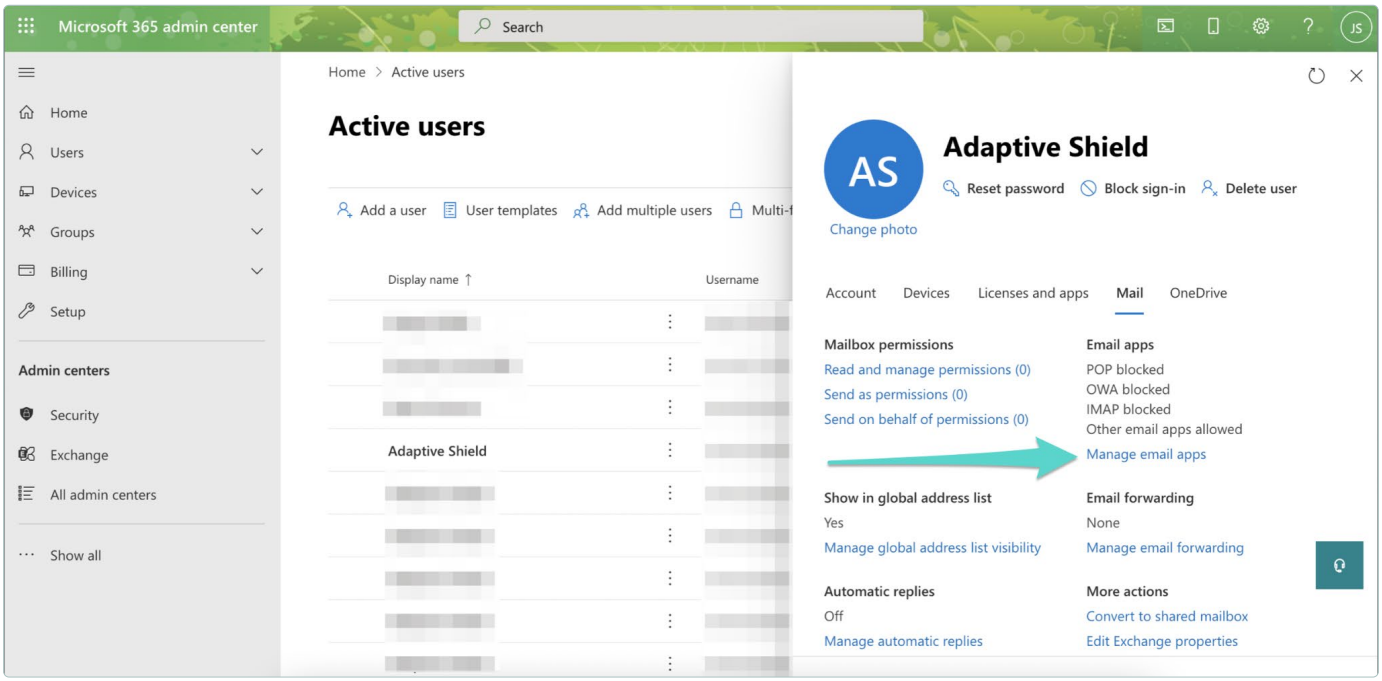

Figure 4. Basic authentication protocols are unchecked

There is no SMTP bulk edit multiple mailboxes (POP and IMAP bulk edit can be found in the classic Exchange Admin Center). Transport config controls the entire Exchange organization, and one of its capabilities is to turn off the SMTP service (both basic and modern).

Use PowerShell command to disable SMTP globally.

```
Set-TransportConfig -SmtpClientAuthenticationDisabled $true
```

In order to block basic authentication protocols for all mailboxes or subset use PowerShell cmdlets:

```
$Users = Get-CASMailbox -ResultSize unlimited
$Users | foreach {Set-CASMailbox -Identity $_ -SmtpClientAuthenticationD
isabled $true -ActiveSyncEnabled $false -ImapEnabled $false -MapiEnabled
$false -PopEnabled $false -OWAEnabled $false}
```

Exclusions

There are cases which you might consider to exclude and allow legacy protocols. For example, a manager who is using an older device or a script that was developed using the legacy protocols and now needs to be redeveloped might require an exclusion.

In these cases, it is strongly recommended to:

1/ Document

Have a procedure in place for requests and their reasoning

2/ Limit

Put in place a time period that will allow the requester time to resolve the issue that they need the legacy protocols, whether replacing the device or time to rewrite the code, etc.

3/ Conditional Access

Use compensating controls by allowing only specific devices, or put in place IP restrictions, geofencing, and more with the Conditional Access policies.

Conclusion

Managing SaaS configurations in an enterprise is complicated and this guide is meant to help ease the pain and smooth the transition from the M365 legacy protocols to a modern environment. The process has multiple steps and requires continuous oversight. From discovery of the legacy authentication protocols opened and used by users and business processes to blocking access and implementing and managing the exclusions, security teams need to dive in, remediate and manage every problematic use of the authentication protocol.

In large-scale environments, where changes always happen and configurations are in the thousands, it is recommended to manage SaaS misconfigurations with an automated SaaS Security Posture Management solution (SSPM).

Appendix A: The Adaptive Shield PowerShell Script

```
function Get-DeepClone {
  [CmdletBinding()]
  param( $InputObject )

  if($InputObject -is [hashtable]) {
    $clone = @{}
    foreach($key in $InputObject.keys) {
      $clone[$key] = Get-DeepClone $InputObject[$key]
    }
    return $clone
  } else {
      return $InputObject
  }
}

$my _ protocols _ hash = @{
  activesync = @{
    attribute = 'AllowBasicAuthActiveSync'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  imap = @{
    attribute = 'AllowBasicAuthImap'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  mapi = @{
    attribute = 'AllowBasicAuthMapi'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  pop = @{
    attribute = 'AllowBasicAuthPop'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  smtp = @{
    attribute = 'AllowBasicAuthSmtp'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  ews = @{
    attribute = 'AllowBasicAuthWebServices'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  autodiscover = @{
    attribute = 'AllowBasicAuthAutodiscover'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  oab = @{
    attribute = 'AllowBasicAuthOfflineAddressBook'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  outlookservice = @{
    attribute = 'AllowBasicAuthOutlookService'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  rpc = @{
    attribute = 'AllowBasicAuthRpc'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  rws = @{
    attribute = 'AllowBasicAuthReportingWebServices'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  PowerShell = @{
    attribute = 'AllowBasicAuthPowerShell'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
}
$cas _ protocols _ hash = @{
  activesync = @{
    attribute = 'ActiveSyncEnabled'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  imap = @{
    attribute = 'ImapEnabled'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  mapi = @{
    attribute = 'MapiEnabled'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  pop = @{
    attribute = 'PopEnabled'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
  outlookservice = @{
    attribute = 'OWAEnabled'
    enabled = $true
    blocked _ method = "NOT BLOCKED"
  }
}
$msol = Get-MsolUser
$exchange _ users = Get-User
$policies = Get-AuthenticationPolicy
$cas = Get-CASMailbox
$org _ config = Get-OrganizationConfig
$transport _ config = Get-TransportConfig | Select-Object SmtpClientAuthenticationDisabled
$def _ auth _ policy = $policies | Where-Object {$ _ .Name -eq $org _ config.
DefaultAuthenticationPolicy}
$users = @()

foreach ($user in $msol){
  $policy = $null
  $is _ def = $null
  $exchange _ user = $null
  $user _ policy = $null
  if ($def _ auth _ policy){
    $policy = $def _ auth _ policy
    $is _ def = $true
  }
  $exchange _ user = $exchange _ users | Where-Object {$ _ .UserPrincipalName -eq $user.
UserPrincipalName}
  $user _ policy = $policies | Where-Object {$ _ .Name -eq $exchange _ user.
AuthenticationPolicy}
  if ($user _ policy){
    $policy = $user _ policy
    $is _ def = $false
  }
  $user _ protocols = Get-DeepClone $my _ protocols _ hash
  if ($policy){
    foreach ($protocol in $user _ protocols.Keys){
      $attr = $user _ protocols[$protocol].attribute
      $user _ protocols[$protocol].enabled = $policy.$attr
      if ($policy.$attr -eq $false){
        $user _ protocols[$protocol].blocked _ method = "Authentication Policy"
      }
    }
  }
  #SMTP
  $cas _ mailbox = $cas | Where-Object {$ _ .PrimarySmtpAddress -eq $user.UserPrincipalName}
  if ($cas _ mailbox){
    if ($cas _ mailbox.SmtpClientAuthenticationDisabled -eq $null){
      # Transport Config SMTP AUTH
      if ($transport _ config.SmtpClientAuthenticationDisabled -eq $true){
        if ($user _ protocols.smtp.enabled -eq $false){
          $user _ protocols.smtp.blocked _ method = $user _ protocols.smtp.blocked _ method +
"; Transport Config"
        }else{
          $user _ protocols.smtp.enabled = $false
          $user _ protocols.smtp.blocked _ method = "Transport Config"
        }
      }
    }elseif ($cas _ mailbox.SmtpClientAuthenticationDisabled -eq $true){
      #CAS SMTP
      if ($user _ protocols.smtp.enabled -eq $false){
        $user _ protocols.smtp.blocked _ method = $user _ protocols.smtp.blocked _ method + ";
CAS Mailbox"
      }else{
        $user _ protocols.smtp.enabled = $false
        $user _ protocols.smtp.blocked _ method = "CAS Mailbox"
      }
    }
  }else{
    $user _ protocols.smtp.blocked _ method = "No Mailbox"
    $user _ protocols.smtp.enabled = $false
  }

  # CAS Clients
  if ($cas _ mailbox){
    $cas _ protocols = Get-DeepClone $cas _ protocols _ hash
    foreach ($protocol in $cas _ protocols.Keys){
      $attr = $cas _ protocols[$protocol].attribute
      if ($cas _ mailbox.$attr -eq $false){
        if ($user _ protocols[$protocol].enabled -eq $false){
          $user _ protocols[$protocol].blocked _ method = $user _ protocols[$protocol].
blocked _ method + "; CAS Mailbox"
        }else{
          $user _ protocols[$protocol].blocked _ method = "CAS Mailbox"
        }
        $user _ protocols[$protocol].enabled = $false
      }
    }
  }else{
    $user _ protocols[$protocol].blocked _ method = "No Mailbox"
    $user _ protocols[$protocol].enabled = $false
  }


  $users += [pscustomobject]@{
    user = $user.UserPrincipalName
    has _ mailbox = !!($cas _ mailbox)
    blocked = $user.BlockCredential
    mfa = ($user.StrongAuthenticationMethods.Count -gt 0)
    auth _ policy = $policy.Name
    is _ ap _ def = $is _ def
    activesync = $user _ protocols.activesync.enabled
    activesync _ method = $user _ protocols.activesync.blocked _ method
    imap = $user _ protocols.imap.enabled
    imap _ method = $user _ protocols.imap.blocked _ method
    mapi = $user _ protocols.mapi.enabled
    mapi _ method = $user _ protocols.mapi.blocked _ method
    pop = $user _ protocols.pop.enabled
    pop _ method = $user _ protocols.pop.blocked _ method
    smtp = $user _ protocols.smtp.enabled
    smtp _ method = $user _ protocols.smtp.blocked _ method
    outlookservice = $user _ protocols.outlookservice.enabled
    outlookservice _ method = $user _ protocols.outlookservice.blocked _ method
    PowerShell = $user _ protocols.PowerShell.enabled
    ExchangeWebServices = $user _ protocols.ews.enabled
    autodiscover = $user _ protocols.autodiscover.enabled
    OfflineAddressBook = $user _ protocols.oab.enabled
    rpc = $user _ protocols.rpc.enabled
    ReportingWebServices = $user _ protocols.rws.enabled
  }
}
$users | Export-Csv -Path .\BasicProtocolsReport.csv
```

DISCLAIMER: The script isn't optimized for large M365 accounts. Organizations need to take into account the compute/memory capacity that the host is running.

Appendix B: Execution

1/ Download the script **posturer.ps1** and save it on the machine you use to connect PowerShell.

2/ Connect using PowerShell to your organization (with a Global reader permissions)

```
$User = "user@org.com"
$PWord = ConvertTo-SecureString -String 'PASSWORD123' -AsPlainText
-Force
$LiveCred = New-Object -TypeName System.Management.Automation.
PSCredential -ArgumentList $User, $PWord
$Session = New-PSSession -ConfigurationName Microsoft.Exchange
-ConnectionUri https://ps.outlook.com/PowerShell/ -Credential
$LiveCred -Authentication Basic -AllowRedirection
Import-PSSession $Session
Connect-MsolService -Credential $LiveCred;
```

3/ Execute the script **./posturer.ps1**

About Us

Adaptive Shield, the leading SaaS Security Posture Management (SSPM) company, enables security teams to see and fix configuration weaknesses quickly in their SaaS environment, ensuring compliance with company and industry standards. Adaptive Shield works with numerous Fortune 500 enterprises to help them gain control over their SaaS threat landscape.

Visit us at www.adaptive-shield.com

Follow us

Request a Demo