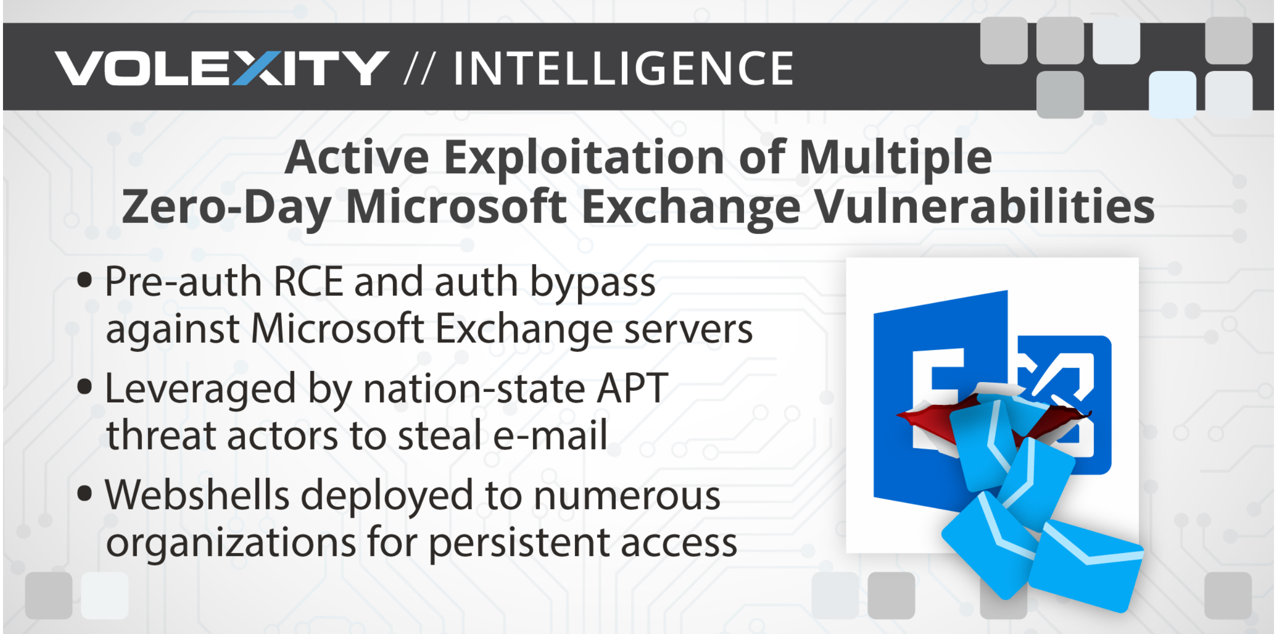
Operation Exchange Marauder: Active Exploitation of Multiple Zero-Day Microsoft Exchange Vulnerabilities

volexity.com/blog/2021/03/02/active-exploitation-of-microsoft-exchange-zero-day-vulnerabilities

March 2, 2021

by Josh Grunzweig, Matthew Meltzer, Sean Koessel, Steven Adair, Thomas Lancaster


VOLEXITY // INTELLIGENCE
Active Exploitation of Multiple Zero-Day Microsoft Exchange Vulnerabilities
• Pre-auth RCE and auth bypass against Microsoft Exchange servers
• Leveraged by nation-state APT threat actors to steal e-mail
• Webshells deployed to numerous organizations for persistent access

[UPDATE] March 8, 2021 – Since original publication of this blog, Volexity has now observed that cyber espionage operations using the SSRF vulnerability CVE-2021-26855 started occurring on January 3, 2021, three days earlier than initially posted.

Volexity is seeing active in-the-wild exploitation of multiple Microsoft Exchange vulnerabilities used to steal e-mail and compromise networks. These attacks appear to have started as early as January 6, 2021.

In January 2021, through its Network Security Monitoring service, Volexity detected anomalous activity from two of its customers' Microsoft Exchange servers. Volexity identified a large amount of data being sent to IP addresses it believed were not tied to legitimate users. A closer inspection of the IIS logs from the Exchange servers revealed rather alarming results. The logs showed inbound POST requests to valid files associated with images, JavaScript, cascading style sheets, and fonts used by Outlook Web Access (OWA). It was initially suspected the servers might be backdoored and that webshells were being executed through a malicious HTTP module or ISAPI filter. As a result, Volexity started its incident response efforts and acquired system memory (RAM) and other disk artifacts to initiate a forensics investigation. This investigation revealed that the servers were not backdoored and uncovered a zero-day exploit being used in the wild.

Through its analysis of system memory, Volexity determined the attacker was exploiting a zero-day server-side request forgery (SSRF) vulnerability in Microsoft Exchange (CVE-2021-26855). The attacker was using the vulnerability to steal the full contents of several user mailboxes. This vulnerability is remotely exploitable and does not require authentication of any kind, nor does it require any special knowledge or access to a target environment. The attacker only needs to know the server running Exchange and the account from which they want to extract e-mail.

Additionally, Volexity is providing alternative mitigations that may be used by defenders to assist in securing their Microsoft Exchange instances. This vulnerability has been confirmed to exist within the latest version of Exchange 2016 on a fully patched Windows Server 2016 server. Volexity also confirmed the vulnerability exists in Exchange 2019 but has not tested against a fully patched version, although it believes they are vulnerable. It should also be noted that is vulnerability does not appear to impact Office 365.

Following the discovery of CVE-2021-26855, Volexity continued to monitor the threat actor and work with additional impacted organizations. During the course of multiple incident response efforts, Volexity identified that the attacker had managed to chain the SSRF vulnerability with another that allows **remote code execution** (RCE) on the targeted Exchange servers (CVE-2021-27065). In all cases of RCE, Volexity has observed the attacker writing webshells (ASPX files) to disk and conducting further operations to dump credentials, add user accounts, steal copies of the Active Directory database (NTDS.DIT), and move laterally to other systems and environments.
A patch addressing both of these vulnerabilities is expected imminently.

Authentication Bypass Vulnerability

While Volexity cannot currently provide full technical details of the exploit and will not be sharing proof-of-concept exploit code, it is still possible to provide useful details surrounding the vulnerability's exploitation and possible mitigations. Volexity observed the attacker focused on getting a list of e-mails from a targeted mailbox and downloading them. Based on these observations, it was possible for Volexity to further improve and automate attacks in a lab environment.

There are two methods to download e-mail with this vulnerability, depending on the way that Microsoft Exchange has been configured. In corporate environments it is common that multiple Exchange servers will be set up. This is often done for load balancing, availability, and resource splitting purposes. While it is less common, it is also possible to run all Exchange functionality on a single server.

In the case where a single server is being used to provide the Exchange service, Volexity believes the attacker must know the targeted user's domain security identifier (SID) in order to access their mailbox. This is a static value and is not considered something secret. However, it is not something that is trivially obtained by someone without access to systems within a specific organization.

In a multiple server configuration, where the servers are configured in a Database Availability Group (DAG), Volexity has proven an attacker does not need to acquire a user's domain SID to access their mailbox. The only information required is the e-mail address of the user they wish to target.

In order to exploit this vulnerability, the attacker must also identify the fully qualified domain name (FQDN) of the internal Exchange server(s). Using a series of requests, Volexity determined that this information could be extracted by an attacker with only initial knowledge of the external IP address or domain name of a publicly accessible Exchange server. After this information is obtained, the attacker can generate and send a specially crafted HTTP POST request to the Exchange server with an XML SOAP payload to the Exchange Web Services (EWS) API endpoint. This SOAP request, using specially crafted cookies, bypasses authentication and ultimately executes the underlying request specified in the XML, allowing an attacker to perform any operation on the users' mailbox.

Volexity has observed this attack conducted via OWA. The exploit involved specially crafted POST requests being sent to a valid static resources that does not require authentication. Specifically, Volexity has observed POST requests targeting files found on the following web directory:

```
/owa/auth/Current/themes/resources/
```

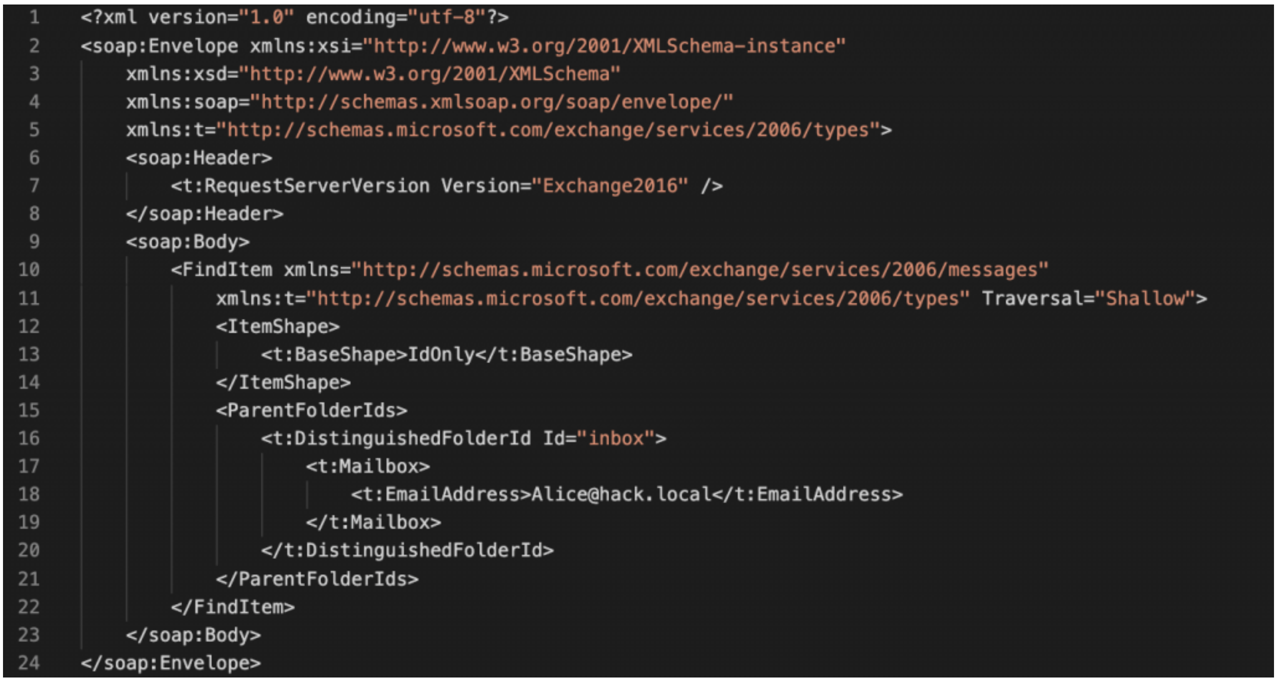
This folder contains image, font, and cascading style sheet files. Using any of these files for the POST request appears to allow the exploit to proceed. If a file such as /owa/auth/logon.aspx or simply a folder such as /owa/auth/ were to be used, the exploit will not work.

Authentication Bypass Exploit Demonstration

The video below demonstrates the vulnerability being exploited in a lab environment:

Figure 1. Video demonstrating the authentication bypass vulnerability at work in a lab environment.

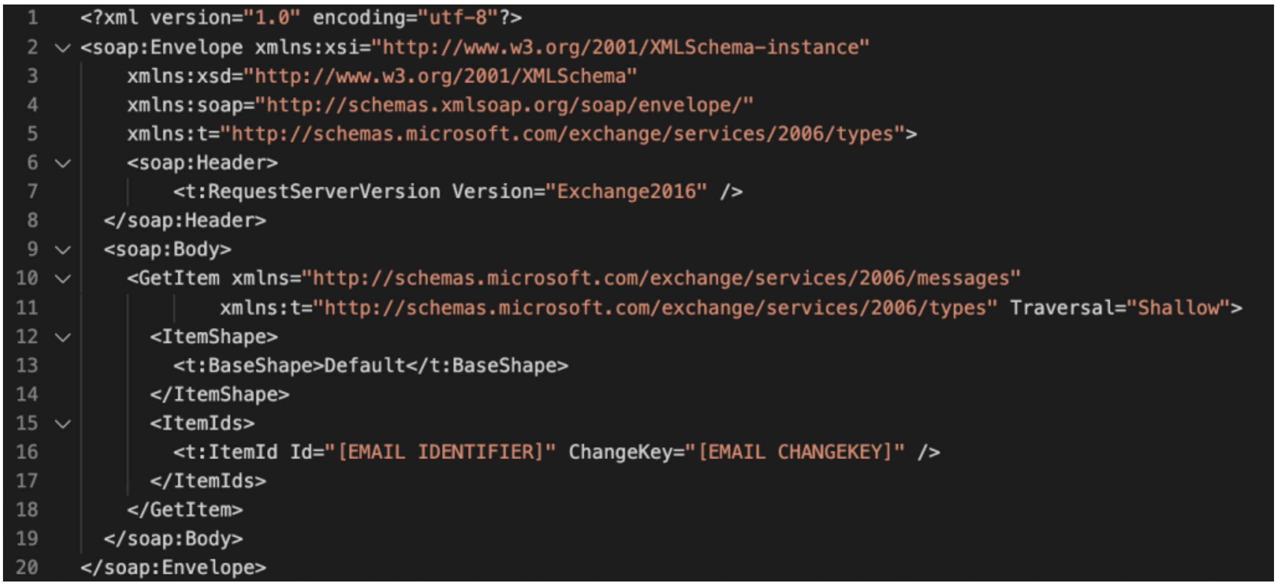
In the video demonstration, the following SOAP XML payload is used to retrieve the identifiers of each email in Alice's inbox:



```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
    <soap:Header>
        <t:RequestServerVersion Version="Exchange2016" />
    </soap:Header>
    <soap:Body>
        <FindItem xmlns="http://schemas.microsoft.com/exchange/services/2006/messages"
            xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types" Traversal="Shallow">
            <ItemShape>
                <t:BaseShape>IdOnly</t:BaseShape>
            </ItemShape>
            <ParentFolderIds>
                <t:DistinguishedFolderId Id="inbox">
                    <t:Mailbox>
                        <t:EmailAddress>Alice@hack.local</t:EmailAddress>
                    </t:Mailbox>
                </t:DistinguishedFolderId>
            </ParentFolderIds>
        </FindItem>
    </soap:Body>
</soap:Envelope>
```

Figure 2. XML payload used to pull email identifiers from Alice's inbox without authentication.

Then, the following payload is used to pull down each individual email:



```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types">
    <soap:Header>
        <t:RequestServerVersion Version="Exchange2016" />
  </soap:Header>
  <soap:Body>
    <GetItem xmlns="http://schemas.microsoft.com/exchange/services/2006/messages"
            xmlns:t="http://schemas.microsoft.com/exchange/services/2006/types" Traversal="Shallow">
      <ItemShape>
        <t:BaseShape>Default</t:BaseShape>
      </ItemShape>
      <ItemIds>
        <t:ItemId Id="[EMAIL IDENTIFIER]" ChangeKey="[EMAIL CHANGEKEY]" />
      </ItemIds>
    </GetItem>
  </soap:Body>
</soap:Envelope>
```

Figure 3. Payload used to retrieve individual emails without authentication.

Remote Code Execution Vulnerability

As mentioned in the introduction to this post, a remote code execution (RCE) exploit was also observed in use against multiple organizations. This RCE appears to reside within the use of the Set-OabVirtualDirectory ExchangePowerShell cmdlet. Evidence of this activity can be seen in Exchange's ECP Server logs. A snippet with the exploit removed is shown below.

> ;'S:CMD=Set-OabVirtualDirectory.ExternalUrl="<removed>"

IIS logs for the server would show an entry similar to what is shown below; however, this URL path may be used for items not associated with this exploit or activity.

> /ecp/DDI/DDIService.svc/SetObject

In this case, this simple backdoor, which Volexity has named SIMPLESEESHARP, was then used to drop a larger webshell, named SPORTSBALL, on affected systems. Further, Volexity has observed numerous other webshells in use, such as China Chopper variants and ASPXSPY.

POST Exploitation Activity

While the attackers appear to have initially flown largely under the radar by simply stealing e-mails, they recently pivoted to launching exploits to gain a foothold. From Volexity's perspective, this exploitation appears to involve multiple operators using a wide variety of tools and methods for dumping credentials, moving laterally, and further backdooring systems. Below is a summary of the different methods and tools Volexity has observed thus far:

Method/Tool	Purpose
rundll32 C:\windows\system32\comsvcs.dll MiniDump lsass.dmp	Dump process memory of lsass.exe to obtain credentials
PsExec	Windows Sysinternals tool used to execute commands on remote systems
ProcDump	Windows Sysinternals tool to dump process memory
WinRar Command Line Utility	Used archive data exfiltration
Webshells (ASPX and PHP)	Used to allow command execution or network proxying via external websites
Domain Account User Addition	Leveraged by attackers to add their own user account and grant it privileges to provide access in the future

Indicators of Compromise

Authentication Bypass Indicators

In Volexity's observations of authentication bypass attacks being performed in the wild, files such as the following were the targets of HTTP POST requests:

> /owa/auth/Current/themes/resources/logon.css
> /owa/auth/Current/themes/resources/owafont_ja.css
> /owa/auth/Current/themes/resources/lgnbotl.gif
> /owa/auth/Current/themes/resources/owafont_ko.css
> /owa/auth/Current/themes/resources/SegoeUI-SemiBold.eot
> /owa/auth/Current/themes/resources/SegoeUI-SemiLight.ttf
> /owa/auth/Current/themes/resources/lgnbotl.gif

Remote Code Execution Indicators

To identify possible historical activity related to the remote code execution exploit, organizations can search their ECP Server logs for the following string (or similar).

```
S:CMD=Set-OabVirtualDirectory.ExternalUrl='
```

ECP Server logs are typically located at <exchange install path>\Logging\ECP\Server\

Webshell Indicators

Further, Volexity has observed indicators that are consistent with web server breaches that can be used to look on disk and in web logs for access to or the presence of ASPX files at the following paths:

> \inetpub\wwwroot\aspnet_client\ (any .aspx file under this folder or sub folders)
>
> \<exchange install path>\FrontEnd\HttpProxy\ecp\auth\ (any file besides TimeoutLogoff.aspx)
> \<exchange install path>\FrontEnd\HttpProxy\owa\auth\ (any file or modified file that is not part of a standard install)
> \<exchange install path>\FrontEnd\HttpProxy\owa\auth\Current\<any aspx file in this folder or subfolders>
> \<exchange install path>\FrontEnd\HttpProxy\owa\auth\<folder with version number>\<any aspx file in this folder or subfolders>

It should be noted that Volexity has observed the attacker adding webshell code to otherwise legitimate ASPX files in an attempt to blend in and hide from defenders.

Web Log User-Agents

There are also a handful of User-Agent that may be useful for responders to look for when examining their web logs. These are not necessarily indicative of compromise, but should be used to determine if further investigation.

Volexity observed the following non-standard User-Agents associated with POST requests to the files found under folders within /owa/auth/Current.

> DuckDuckBot/1.0;+(+http://duckduckgo.com/duckduckbot.html)
> facebookexternalhit/1.1+(+http://www.facebook.com/externalhit_uatext.php)
> Mozilla/5.0+(compatible;+Baiduspider/2.0;++http://www.baidu.com/search/spider.html)
> Mozilla/5.0+(compatible;+Bingbot/2.0;++http://www.bing.com/bingbot.htm)
> Mozilla/5.0+(compatible;+Googlebot/2.1;++http://www.google.com/bot.html
> Mozilla/5.0+(compatible;+Konqueror/3.5;+Linux)+KHTML/3.5.5+(like+Gecko)+(Exabot-Thumbnails)
> Mozilla/5.0+(compatible;+Yahoo!+Slurp;+http://help.yahoo.com/help/us/ysearch/slurp)
> Mozilla/5.0+(compatible;+YandexBot/3.0;++http://yandex.com/bots)
> Mozilla/5.0+(X11;+Linux+x86_64)+AppleWebKit/537.36+(KHTML,+like+Gecko)+Chrome/51.0.2704.103+Safari/537.36

Volexity observed the following User-Agents in conjunction with exploitation to /ecp/ URLs.

> ExchangeServicesClient/0.0.0.0
> python-requests/2.19.1
> python-requests/2.25.1

Further other notable User-Agent entries tied to tools used for post-exploitation access to webshells.

> antSword/v2.1
>
> Googlebot/2.1+(+http://www.googlebot.com/bot.html)
> Mozilla/5.0+(compatible;+Baiduspider/2.0;++http://www.baidu.com/search/spider.html)

Additional Auth Bypass and RCE Indicators

To identify possible historical activity relating to the authentication bypass and RCE activity, IIS logs from Exchange servers can be examined for the following:

> POST /owa/auth/Current/
> POST /ecp/default.flt
>
> POST /ecp/main.css
>
> POST /ecp/<single char>.js

Note that the presence of log entries with POST requests under these directories does not guarantee an Exchange server has been exploited. However, its presence should warrant further investigation.

Yara signatures for non trivial webshells deployed by attackers following successful exploitation may be found in the Appendix of this post.

Network Indicators – Attacker IPs

Volexity has observed numerous IP addresses leveraged by the attackers to exploit the vulnerabilities described in this blog. These IP addresses are tied to VPS servers and VPN services. Volexity has also observed the attackers using Tor, but has made attempts to remove those entries from the list below.

> 103.77.192.219
> 104.140.114.110
> 104.250.191.110
> 108.61.246.56
> 149.28.14.163
> 157.230.221.198
> 167.99.168.251
> 185.250.151.72
> 192.81.208.169
> 203.160.69.66
> 211.56.98.146
> 5.254.43.18
> 80.92.205.81

Conclusion

Highly skilled attackers continue to innovate in order to bypass defenses and gain access to their targets, all in support of their mission and goals. These particular vulnerabilities in Microsoft Exchange are no exception. These attackers are conducting novel attacks to bypass authentication, including two-factor authentication, allowing them to access e-mail accounts of interest within targeted organizations and remotely execute code on vulnerable Microsoft Exchange servers.

Due to the ongoing observed exploitation of the discussed vulnerabilities, Volexity urges organizations to immediately apply the available patches or temporarily disabling external access to Microsoft Exchange until a patch can be applied.

Need Assistance?

If you have concerns that your servers or networks may have been compromised from this vulnerability, please reach out to the Volexity team and we can help you make a determination if further investigation is warranted.

Appendix

```
rule webshell_aspx_simpleseesharp : Webshell Unclassified
{

meta:

author= “threatintel@volexity.com”
date= “2021-03-01”
description= “A simple ASPX Webshell that allows an attacker to write further files to disk.”
hash= “893cd3583b49cb706b3e55ecb2ed0757b977a21f5c72e041392d1256f31166e2”

strings:

$header= “<%@ Page Language=\”C#\” %>”
$body= “<% HttpPostedFile thisFile = Request.Files[0];thisFile.SaveAs(Path.Combine”

condition:

$header at 0 and
$body and
filesize < 1KB

}
```

```
rule webshell_aspx_reGeorgTunnel : Webshell Commodity
{
meta:
author= “threatintel@volexity.com”
date= “2021-03-01”
description= “variation on reGeorgtunnel”
hash= “406b680edc9a1bb0e2c7c451c56904857848b5f15570401450b73b232ff38928”
reference= “https://github.com/sensepost/reGeorg/blob/master/tunnel.aspx”
strings:
$s1= “System.Net.Sockets”
$s2= “System.Text.Encoding.Default.GetString(Convert.FromBase64String(StrTr(Request.Headers.Get”
$t1 = “.Split(‘|’)”
$t2= “Request.Headers.Get”
$t3= “.Substring(“
$t4= “new Socket(“
$t5= “IPAddress ip;”
condition:
all of ($s*) or
all of ($t*)
}
```

```
rule webshell_aspx_sportsball : Webshell
{

meta:

author= “threatintel@volexity.com”
date= “2021-03-01”
description= “The SPORTSBALL webshell allows attackers to upload files or execute
commands on the system.”
hash= “2fa06333188795110bba14a482020699a96f76fb1ceb80cbfa2df9d3008b5b0a”

strings:

$uniq1= “HttpCookie newcook = new HttpCookie(\”fqrspt\”,
HttpContext.Current.Request.Form”
$uniq2= “ZN2aDAB4rXsszEvCLrzgcvQ4oi5J1TuiRULlQbYwldE=”

$var1= “Result.InnerText = string.Empty;”
$var2= “newcook.Expires = DateTime.Now.AddDays(”
$var3= “System.Diagnostics.Process process = new System.Diagnostics.Process()”
$var4= “process.StandardInput.WriteLine(HttpContext.Current.Request.Form[\””
$var5= “else if (!string.IsNullOrEmpty(HttpContext.Current.Request.Form[\””
$var6= “<input type=\”submit\” value=\”Upload\” />”

condition:

any of ($uniq*) or
all of ($var*)

}
```