Commit 8d8fc0c

Daniel Neto committed last week · ✓ 11 / 11

```
fix: Enhance SSRF protection by implementing DNS pinning in proxy requests

GHSA-793q-xgj6-7frp
```

master

1 parent bf1c769 commit 8d8fc0c

2 files changed +114 -36 lines changed

Top

Filter files…

- objects
  - functions.php
- plugin/LiveLinks
  - proxy.php

2 files changed +114 -36 lines changed

Search within code

objects/functions.php

```
@@ -4261,7 +4261,7 @@ function isSafeRedirectURL($url)
4261 4261  * @param string $url The URL to validate
4262 4262  * @return bool True if safe from SSRF, false otherwise
4263 4263  */
4264      - function isSSRFSafeURL($url)
     4264 + function isSSRFSafeURL($url, &$resolvedIP = null)
4265 4265 {
4266 4266     global $global;
4267 4267     if (empty($url) || !is_string($url)) {
@@ -4317,17 +4317,19 @@ function isSSRFSafeURL($url)
```

```
4317 4317             return false;
4318 4318         }
4319 4319
4320      -      // Resolve hostname to IP to check for DNS rebinding attacks
     4320 +      // Resolve hostname to IP to check for DNS rebinding attacks.
     4321 +      // $resolvedIP (out-param) is set to the final validated IP so callers can
     4322 +      // pin the DNS resolution (e.g. via CURLOPT_RESOLVE) and eliminate TOCTOU races.
4321 4323         $ip = $host;
4322 4324         if (!filter_var($host, FILTER_VALIDATE_IP)) {
4323 4325             // It's a hostname, resolve it
4324      -          $resolvedIP = gethostbyname($host);
4325      -          if ($resolvedIP === $host) {
     4326 +          $dnsResolved = gethostbyname($host);
     4327 +          if ($dnsResolved === $host) {
4326 4328                 // DNS resolution failed
4327 4329                 _error_log("isSSRFSafeURL: DNS resolution failed for: {$host}");
4328 4330                 return false;
4329 4331             }
4330      -          $ip = $resolvedIP;
     4332 +          $ip = $dnsResolved;
4331 4333         }
4332 4334
4333 4335         // Remove IPv6 brackets if present
@@ -4369,6 +4371,8 @@ function isSSRFSafeURL($url)
4369 4371             }
4370 4372         }
4371 4373
     4374 +      // Expose the validated IP to the caller for DNS pinning.
     4375 +      $resolvedIP = $ip;
4372 4376         return true;
4373 4377     }
4374 4378
```

plugin/LiveLinks/proxy.php

```
@@ -22,8 +22,11 @@
```

```
22  22          exit;
23  23      }
24  24
25      - // SSRF Protection: Block requests to internal/private networks
26      - if (!isSSRFSafeURL($_GET['livelink'])) {
    25  + // SSRF Protection: Block requests to internal/private networks.
    26  + // $resolvedIP receives the validated IP so we can pin it in the cURL call below,
    27  + // eliminating the DNS TOCTOU race between validation and TCP connect.
    28  + $resolvedIP = null;
    29  + if (!isSSRFSafeURL($_GET['livelink'], $resolvedIP)) {
27  30          _error_log("LiveLinks proxy: SSRF protection blocked URL: " . $_GET['livelink']);
28  31          echo "Access denied: URL targets restricted network";
29  32          exit;
        @@ -32,43 +35,114 @@
32  35      header("Content-Type: video/vnd.mpegurl");
33  36      header("Content-Disposition: attachment;filename=playlist.m3u");
34  37
35      - $options = array(
36      -     'http' => array(
37      -         'user_agent' => 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36',
38      -         "method" => "GET",
39      -         "header" => array("Referer: localhost\r\nAccept-languange: en\r\nCookie: foo=bar\r\n"),
40      -         'follow_location' => 0,
41      -         'max_redirects' => 0,
42      -     )
43      - );
44      - $context = stream_context_create($options);
45      -
46  38      $_GET['livelink'] = addGlobalTokenIfSameDomain($_GET['livelink']);
47  39
48      - $headers = get_headers($_GET['livelink'], 1, $context);
49      - if (!empty($headers["Location"])) {
50      -     $redirectUrl = $headers["Location"];
51      -
52      -     // Validate the redirect target URL format and scheme before SSRF check
```

```
53        -     if (!filter_var($redirectUrl, FILTER_VALIDATE_URL) ||
              !preg_match("/^https?:\/\//i", $redirectUrl)) {
54        -         _error_log("LiveLinks proxy: invalid redirect URL: " . $redirectUrl);
55        -         echo "Access denied: Invalid redirect URL";
56        -         exit;
      40  + /**
      41  +  * Fetch a URL through a DNS-pinned cURL request, following redirects manually
      42  +  * so every hop is re-validated with isSSRFSafeURL() and re-pinned via
              CURLOPT_RESOLVE.
      43  +  *
      44  +  * This eliminates the DNS TOCTOU race: gethostbyname() is called once per hop
      45  +  * (inside isSSRFSafeURL), and that same IP is forced into the TCP connection
              via
      46  +  * CURLOPT_RESOLVE — no second DNS lookup can occur between validation and
              connect.
      47  +  *
      48  +  * @param string $url           The initial URL to fetch.
      49  +  * @param string|null $pinnedIP The pre-validated IP returned by
              isSSRFSafeURL().
      50  +  * @param int $maxRedirects     Maximum number of redirects to follow.
      51  +  * @return array{content:string,finalUrl:string}|false
      52  +  */
      53  + function proxyDNSPinnedFetch($url, $pinnedIP, $maxRedirects = 5)
      54  + {
      55  +     if (!function_exists('curl_init')) {
      56  +         // cURL unavailable: fall back to the non-pinned path and log the
              degradation.
      57  +         // TOCTOU risk remains in this path; operators should ensure cURL is
              installed.
      58  +         _error_log("LiveLinks proxy: cURL unavailable, DNS pinning disabled for
              {$url}");
      59  +         $content = fakeBrowser($url);
      60  +         return $content !== false ? ['content' => $content, 'finalUrl' => $url]
              : false;
57    61          }
58    62
59        -     // SSRF Protection: Re-validate redirect target against internal/private
              networks
60        -     if (!isSSRFSafeURL($redirectUrl)) {
```

```
61  -           _error_log("LiveLinks proxy: SSRF protection blocked redirect URL: " . $redirectUrl);
62  -           echo "Access denied: Redirect URL targets restricted network";
63  -           exit;
63  +       $currentUrl  = $url;
64  +       $currentIP   = $pinnedIP;
65  +
66  +       for ($hop = 0; $hop <= $maxRedirects; $hop++) {
67  +           $host   = parse_url($currentUrl, PHP_URL_HOST);
68  +           $scheme = strtolower((string) parse_url($currentUrl, PHP_URL_SCHEME));
69  +           $port   = parse_url($currentUrl, PHP_URL_PORT) ?: ($scheme === 'https' ? 443 : 80);
70  +
71  +           $curlOpts = [
72  +               CURLOPT_URL            => $currentUrl,
73  +               CURLOPT_RETURNTRANSFER => true,
74  +               CURLOPT_FOLLOWLOCATION => false,   // we handle each redirect ourselves
75  +               CURLOPT_SSL_VERIFYPEER => false,
76  +               CURLOPT_SSL_VERIFYHOST => 0,
77  +               CURLOPT_USERAGENT      => 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36',
78  +               CURLOPT_HTTPHEADER     => ['Referer: localhost', 'Accept-Language: en', 'Cookie: foo=bar'],
79  +               CURLOPT_HEADER         => true,    // include response headers so we can read Location:
80  +           ];
81  +
82  +           // Pin the validated IP — CURLOPT_RESOLVE format: "hostname:port:ip"
83  +           // cURL uses this map instead of re-resolving the hostname, closing the TOCTOU window.
84  +           if (!empty($currentIP)) {
85  +               $curlOpts[CURLOPT_RESOLVE] = ["{$host}:{$port}:{$currentIP}"];
86  +           }
87  +
88  +           $ch = curl_init();
89  +           curl_setopt_array($ch, $curlOpts);
90  +           $response   = curl_exec($ch);
91  +           $httpCode   = curl_getinfo($ch, CURLINFO_HTTP_CODE);
```

```
92  +         $headerSize = curl_getinfo($ch, CURLINFO_HEADER_SIZE);
93  +         curl_close($ch);
94  +
95  +         if ($response === false || $httpCode === 0) {
96  +             _error_log("LiveLinks proxy: cURL fetch failed for {$currentUrl}");
97  +             return false;
98  +         }
99  +
100 +         $responseHeaders = substr($response, 0, $headerSize);
101 +         $body            = substr($response, $headerSize);
102 +
103 +         if ($httpCode >= 300 && $httpCode < 400) {
104 +             // Parse the single Location header from the raw response headers.
105 +             if (!preg_match('/^Location:\s*(.+)$/im', $responseHeaders, $m)) {
106 +                 _error_log("LiveLinks proxy: 3xx with no Location header at
      {$currentUrl}");
107 +                 return false;
108 +             }
109 +             $redirectUrl = trim($m[1]);
110 +
111 +             // Validate redirect target and get its pinned IP for the next hop.
112 +             $nextIP = null;
113 +             if (!isSSRFSafeURL($redirectUrl, $nextIP)) {
114 +                 _error_log("LiveLinks proxy: SSRF protection blocked redirect
      to: {$redirectUrl}");
115 +                 return false;
116 +             }
117 +
118 +             $currentUrl = $redirectUrl;
119 +             $currentIP  = $nextIP;
120 +             continue;
121 +         }
122 +
123 +         return ['content' => $body, 'finalUrl' => $currentUrl];
64  124         }
65  125
66      -     $_GET['livelink'] = $redirectUrl;
67      -     $urlinfo = parse_url($_GET['livelink']);
68      -     $content = fakeBrowser($_GET['livelink']);
    126 +     _error_log("LiveLinks proxy: too many redirects for {$url}");
```

```
127 +     return false;
128 + }
129 +
130 + $fetchResult = proxyDNSPinnedFetch($_GET['livelink'], $resolvedIP);
131 + if ($fetchResult === false) {
132 +     echo "Access denied or fetch failed";
133 +     exit;
134 + }
135 +
136 + $content  = $fetchResult['content'];
137 + $finalUrl = $fetchResult['finalUrl'];
138 +
139 + // Preserve the original base-URL logic for relative path resolution in m3u8 playlists.
140 + if ($finalUrl !== $_GET['livelink']) {
141 +     // URL was redirected — use scheme://host:port as the base (no trailing path).
142 +     $urlinfo = parse_url($finalUrl);
69 143       $_GET['livelink'] = "{$urlinfo["scheme"]}://{$urlinfo["host"]}:{$urlinfo["port"]}";
144 +     unset($pathinfo);
70 145   } else {
71     -     $content = fakeBrowser($_GET['livelink']);
72 146       $pathinfo = pathinfo($_GET['livelink']);
73 147   }
74 148   if($content === "Empty Token"){
```

Comments 0

Please sign in to comment.