AI vs. AI: Detecting an AI-obfuscated phishing campaign

By Microsoft Threat Intelligence ⋮ ⋮ 9/24/2025

Microsoft Threat Intelligence recently detected and blocked a credential phishing campaign that likely used AI-generated code to obfuscate its payload and evade traditional defenses. Appearing to be aided by a large language model (LLM), the activity obfuscated its behavior within an SVG file, leveraging business terminology and a synthetic structure to disguise its malicious intent. In analyzing the malicious file, Microsoft Security Copilot assessed that the code was “not something a human would typically write from scratch due to its complexity, verbosity, and lack of practical utility.”

Like many transformative technologies, AI is being adopted by both defenders and cybercriminals. While defenders use AI to detect, analyze, and respond to threats at scale, attackers are experimenting with AI to enhance their own operations, such as by crafting more convincing lures, automating obfuscation, and generating code that mimics legitimate content. Even though the campaign in this case was limited in nature and primarily aimed at US-based organizations, it exemplifies a broader trend of attackers leveraging AI to increase the effectiveness and stealth of their operations. This case also underscores the growing need for defenders to understand and anticipate AI-driven threats.

Despite the sophistication of the obfuscation, the campaign was successfully detected and blocked by Microsoft Defender for Office 365’s AI-powered protection systems, which analyze signals across infrastructure, behavior, and message context that remain largely unaffected by an attacker’s use of AI. By

sharing our analysis, we aim to help the security community recognize similar tactics being used by threat actors and reinforce that AI-enhanced threats, while evolving, are not undetectable. As we discuss in this post, an attacker's use of AI often introduces new artifacts that can be leveraged for detection. By applying these insights and our recommended best practices, organizations can strengthen their own defenses against similar emerging, AI-aided phishing campaigns.

Phishing campaign tactics and payload

On August 18, Microsoft Threat Intelligence detected a phishing campaign leveraging a compromised small business email account to distribute malicious phishing emails intended to steal credentials. The attackers employed a self-addressed email tactic, where the sender and recipient addresses matched, and actual targets were hidden in the BCC field, which is done to attempt to bypass basic detection heuristics. The content of the email was crafted to resemble a file-sharing notification, containing the message:

Figure 1. Phishing email example

Attached to the email was a file named *23mb – PDF- 6 pages.svg*, designed to look like a legitimate PDF document even though the file extension indicates it is an SVG file. SVG files (Scalable Vector Graphics) are attractive to attackers because they are text-based and scriptable, allowing them to embed JavaScript and other dynamic content directly within the file. This makes it possible to deliver interactive phishing payloads that appear benign to both users and many security tools. Additionally, SVGs support obfuscation-friendly features such as invisible elements, encoded attributes, and delayed script execution, all of which can be used to evade static analysis and sandboxing.

When opened, the SVG file redirected the user to a webpage that prompted them to complete a CAPTCHA for security verification, a common social engineering tactic used to build trust and delay suspicion. Although our visibility for this incident was limited to the initial landing page due to the activity being detected and blocked, the campaign would have very likely presented a fake sign in page after the CAPTCHA to harvest credentials.

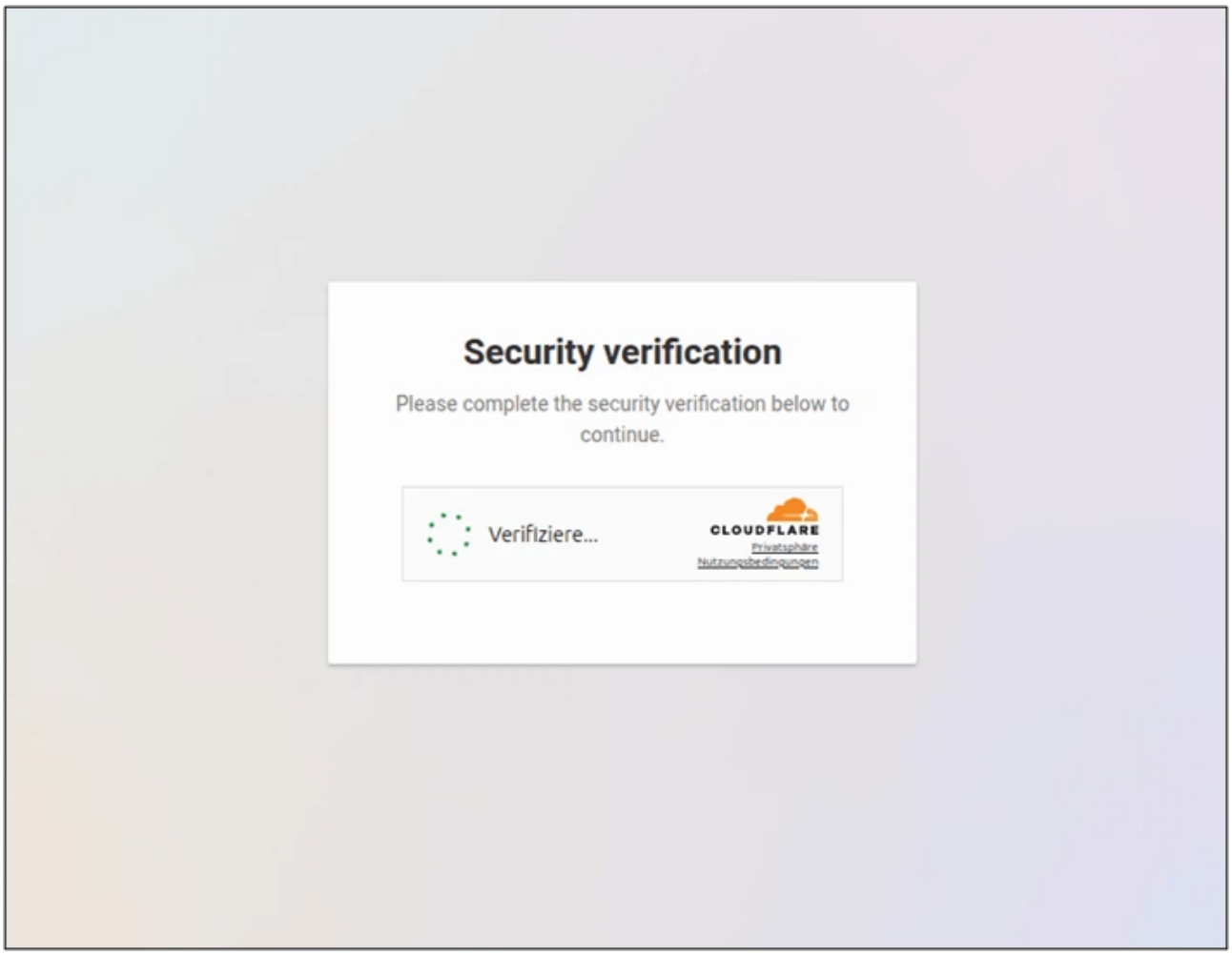

Figure 2. Security verification prompt

An analysis of the SVG code found that it used a unique method of obfuscating its content and behavior. Instead of using cryptographic obfuscation, which is commonly used to obfuscate phishing content, the SVG code in this campaign used business-related language to disguise its malicious activity. It did this in two ways:

First, the beginning of the SVG code was structured to look like a legitimate business analytics dashboard. It contained elements for a supposed Business Performance Dashboard, including chart bars and month labels. These elements, however, were rendered completely invisible to the user by setting their opacity to zero and their fill to transparent. This tactic is designed to mislead anyone casually inspecting the file, making it appear as if the SVG's sole purpose is to visualize business data. In reality, though, it's a decoy.

```
<!-- Background -->
<rect width="100%" height="100%" fill="transparent" opacity="0" />

<!-- Title -->
<text x="400" y="40" text-anchor="middle" font-family="Arial" font-size="24" font-weight="bold"
  fill="transparent" opacity="0">
    Business Performance Dashboard
</text>

<!-- Chart bars -->
<rect x="100" y="200" width="60" height="201" fill="transparent" rx="5" opacity="0" />
<rect x="200" y="180" width="60" height="235" fill="transparent" rx="5" opacity="0" />
<rect x="300" y="160" width="60" height="251" fill="transparent" rx="5" opacity="0" />
<rect x="400" y="140" width="60" height="251" fill="transparent" rx="5" opacity="0" />
<rect x="500" y="120" width="60" height="282" fill="transparent" rx="5" opacity="0" />
<rect x="600" y="100" width="60" height="289" fill="transparent" rx="5" opacity="0" />

<!-- Labels -->
<text x="130" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">Jan</text>
<text x="230" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">Feb</text>
<text x="330" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">Mar</text>
<text x="430" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">Apr</text>
<text x="530" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">May</text>
<text x="630" y="520" text-anchor="middle" font-size="12" fill="transparent" opacity="0">Jun</text>
```

Figure 3. SVG code containing decoy business performance chart

Second, the payload's functionality was also hidden using a creative use of business terms. Within the file, the attackers encoded the malicious payload using a long sequence of business-related terms. Words like revenue, operations, risk, or shares were concatenated into a hidden *data-analytics* attribute of an invisible *<text>* element within the SVG.

```
<!-- Business analytics data -->
<text id="analyticsSourcec8c666" x="0" y="0" opacity="0" data-analytics="revenue-1468,client-
  earningsannualriskannualriskannualsharesannualyieldannualstatementquarterlycapitalquarterlycapitalquarter
  lyequityannualcashannualflowannualdebtannualflowquarterlyreturnannualsharesannualvalueannualreturnannuale
  xpenseannualflowannualriskannualexpenseannualdividendannualyieldannualflowquarterlycostannualexpenseannua
  l,profile-returnquarterlyyieldannualequityannualincomeannualsharesannual-->finance-quarterly-return-
  quarterly-analytics-annual-kpi-annual-customer-quarterly-report-quarterly-investment-annual-goal-
  quarterly-segment-quarterly-data-quarterly-report-quarterly-cost-annual-data-annual-overview-annual-
  summary-annual-kpi-annual-data-quarterly-analysis-quarterly-operations-quarterly-review-annual-objective-
  quarterly-overview-quarterly-strategy-quarterly-sales-quarterly-finance-annual-value-annual-segment-
  annual-cost-annual-return-quarterly-portfolio-quarterly-segment-quarterly-metrics-quarterly-budget-
  annual-metrics-quarterly-overview-quarterly-dashboard-annual-roi-quarterly-sales-quarterly-budget-
  quarterly-budget-quarterly-value-quarterly-metrics-quarterly-investment-quarterly-trends-quarterly-
  operations-quarterly-data-annual-sales-quarterly-operations-quarterly-data-annual-performance-annual-
  analysis-annual-summary-quarterly-sales-annual-kpi-quarterly-review-annual-business-annual-sales-annual-
  kpi-annual-goal-annual-goal-annual-balance-annual-kpi-quarterly-kpi-annual-strategy-annual-kpi-quarterly-
  budget-annual-objective-quarterly-strategy-quarterly-data-quarterly-expense-annual-performance-quarterly-
  insight-quarterly-target-quarterly-roi-annual-"></text>
```

Figure 4. Sequence of business-related terms

The terms in this attribute were later used by embedded JavaScript, which systematically processed the business-related words through several transformation steps. Instead of directly including malicious code, the attackers encoded the payload by mapping pairs or sequences of these business terms to specific characters or instructions. As the script runs, it decodes the sequence, reconstructing the hidden functionality from what appears to be harmless business metadata. This obfuscated functionality included redirecting a user's browser to the initial phishing landing page, triggering browser fingerprinting, and initiating session tracking.

```
// Convert business terminology to processable data format
function convertMetricsDataf98e36(businessMetrics) {
    const standardTerms = [
        "quarterly", "annual", "monthly", "revenue", "profit", "growth", "market", "sales",
        "customer", "analytics", "metrics", "forecast", "performance", "strategy", "operations", "budget",
        "finance", "report", "dashboard", "insight", "data", "trends", "analysis", "business",
        "overview", "summary", "review", "target", "goal", "objective", "kpi", "roi",
        "segment", "portfolio", "investment", "return", "cost", "expense", "value", "margin",
        "earnings", "income", "assets", "equity", "debt", "cash", "flow", "capital",
        "shares", "stock", "dividend", "yield", "risk", "beta", "alpha", "ratio",
        "balance", "sheet", "statement", "audit", "tax", "fiscal", "quarter", "year"
    ];
```

Figure 5. Conversion of business terminology to processable malicious code

Using AI to analyze the campaign

Given the unique methods used to obfuscate the SVG payload's functionality, we hypothesized that the attacker may have used AI to assist them. We asked Security Copilot to analyze the contents of the SVG file to assess whether it was generated by AI or an LLM. Security Copilot's analysis indicated that it was highly likely that the code was synthetic and likely generated by an LLM or a tool using one. Security Copilot determined that the code exhibited a level of complexity and verbosity rarely seen in manually written scripts, suggesting it was produced by an AI model rather than crafted by a human.

Security Copilot provided five key indicators to support its conclusion:

1. **Overly descriptive and redundant naming**
    - The function and variable names (e.g., *processBusinessMetricsf43e08*, *parseDataFormatf19e04*, *convertMetricsDataf98e36*, *initializeAnalytics4e2250*, *userIdentifierb8db*, *securityHash9608*) follow a consistent pattern of descriptive English terms concatenated with random hexadecimal strings. This naming convention is typical of AI/LLM-generated code, which often appends random suffixes to avoid collisions and increase obfuscation.

```
function processBusinessMetricsf43e08(businessData92f9) { ... }
function parseDataFormatf19e04(formattedInputd7c5) { ... }
const reportMetadata259b = { ... }
```

Figure 6. Example of overly descriptive variable and function names

2. **Modular and over-engineered code structure**
    - The code structure is highly modular, with clear separation of concerns and repeated use of similar logic blocks (e.g., mapping business terms to character codes, block reversal, offset correction, token-based validation). This systematic approach is characteristic of AI/LLM output, which tends to over-engineer and generalize solutions.

```
// Parse business terminology pairs for data reconstruction
while (remainingTerms.length > 0) {
    let termFound = false;
    for (let termIndex = 0; termIndex < businessTermsLibfe82.length && !termFound;
termIndex++) {
        for (let subIndex = 0; subIndex < businessTermsLibfe82.length && !termFound;
subIndex++) {
            const businessPattern = businessTermsLibfe82[termIndex] +
businessTermsLibfe82[subIndex];
            if (remainingTerms.indexOf(businessPattern) === 0) {
                const charCode = (termIndex % 64) + (subIndex * 64);
                processedOutputbc64 += String.fromCharCode(charCode);
                remainingTerms = remainingTerms.substring(businessPattern.length);
                termFound = true;
            }
        }
    }
    if (!termFound) break;
}
```

Figure 7. Example of over-engineered logic parsing the business terminology

3. **Generic comments**
    - Comments are verbose, generic, and use formal business language (“Advanced business intelligence data processor”, “Business terminology parser for standardized format conversion”, “Generate secure processing token for data validation”), which is a hallmark of AI-generated documentation.

```
// Convert to processing format
let processedBytes3d5e = new Uint8Array(rawData.length);
for (let byteIndex = 0; byteIndex < rawData.length; byteIndex++) {
    processedBytes3d5e[byteIndex] = rawData.charCodeAt(byteIndex);
}

// Apply reverse data transformation (block reversal)
const segmentSize = 8;
let processedSegments = new Uint8Array(processedBytes3d5e.length);
for (let segIndex = 0; segIndex < processedBytes3d5e.length; segIndex += segmentSize) {
    let segmentEnd = Math.min(segIndex + segmentSize, processedBytes3d5e.length);
    for (let segPos = 0; segPos < segmentEnd - segIndex; segPos++) {
        processedSegments[segIndex + segPos] = processedBytes3d5e[segIndex + (segmentEnd -
          segIndex - 1) - segPos];
    }
}

// Apply character offset correction
let offsetCorrected = new Uint8Array(processedSegments.length);
for (let corrIndex = 0; corrIndex < processedSegments.length; corrIndex++) {
    offsetCorrected[corrIndex] = (processedSegments[corrIndex] + 256 - 7) % 256;
}

// Apply token-based validation processing
let executionPayload85b8 = new Uint8Array(offsetCorrected.length);
for (let tokenIndex = 0; tokenIndex < offsetCorrected.length; tokenIndex++) {
    const tokenChar = processingToken2362.charCodeAt(tokenIndex % processingToken2362.length);
    executionPayload85b8[tokenIndex] = offsetCorrected[tokenIndex] ^ tokenChar;
}

// Generate executable business logic
let businessLogic = "";
for (let logicIndex = 0; logicIndex < executionPayload85b8.length; logicIndex++) {
    businessLogic += String.fromCharCode(executionPayload85b8[logicIndex]);
}
```

Figure 8. Examples of verbose, generic comments.

4. **Formulaic obfuscation techniques**
    - The obfuscation techniques (e.g., encoding business terms, multi-stage data transformation, dynamic function creation) are implemented in a way that is both thorough and formulaic, matching the style of AI/LLM code generation.
5. **Unusual use of CDATA and XML declaration**
    - The SVG code includes both an XML declaration and a CDATA-wrapped script, which is more typical of LLM-generated code that aims to be “technically correct” or to mimic documentation examples, even when such elements are unnecessary for the attack to function.

```
<?xml version="1.0" encoding="UTF-8"?>
<script type="text/javascript"><![CDATA[ ... ]]></script>
```

Figure 9. Example of the SVG’s XML declaration and CDATA-wrapped script

Using AI to detect the campaign

While the use of AI to obfuscate phishing payloads may seem like a significant leap in attacker sophistication, it's important to understand that AI does not fundamentally change the core artifacts that security systems rely on to detect phishing threats. AI-generated code may be more complex or syntactically polished, but it still operates within the same behavioral and infrastructural boundaries as human-crafted attacks.

Microsoft Defender for Office 365 uses AI and machine learning models trained to detect phishing and are designed to identify patterns across multiple dimensions—not just the payload itself. These include:

- Attack infrastructure (such as suspicious domain characteristics, hosting behavior)
- Tactics, techniques, and procedures (TTPs) (such as the use of redirects, CAPTCHA gates, session tracking)
- Impersonation strategies (such as pretending to share documents, mimicking file-sharing notifications)
- Message context and delivery patterns (such as self-addressed emails, BCC usage, mismatched sender/recipient behavior)

These signals are largely unaffected by whether the payload was written by a human or an LLM. In fact, AI-generated obfuscation often introduces synthetic artifacts, like verbose naming, redundant logic, or unnatural encoding schemes, that can become new detection signals themselves.

Despite the use of AI to obfuscate the SVG payload, this campaign was blocked by Microsoft Defender for Office 365's detection system through a combination of infrastructure analysis, behavioral indicators, and message context, none of which were impacted by the use of AI. Signals used to detect this campaign included the following:

- **Use of self-addressed email with BCCed recipients** – This tactic is commonly used to attempt to bypass basic email heuristics and hide the true recipient list.
- **Suspicious file type/name** – SVG files, generally, have been an emerging payload used in phishing attacks and the attachments in this campaign were named to resemble a PDF, which is atypical for legitimate document sharing.
- **Redirect to malicious infrastructure** – The SVG payload redirected to a domain that had previously been identified as being linked to phishing content.
- **General use of code obfuscation** – While the SVG file contained novel obfuscation tactics that hadn't been seen before, the presence of obfuscation alone was an indicator of potentially malicious intent.
- **Suspicious network behavior** – Automated analysis of the phishing site indicated that it employed session tracking and browser fingerprinting, which can be used to selectively serve content based on geography or environment, a behavior used by some phishing actors.

Recommendations

While this campaign was limited in scope and effectively blocked, similar techniques are increasingly being leveraged by a range of threat actors. Sharing our findings equips organizations to identify and mitigate these emerging threats, regardless of the specific threat actor behind them. Microsoft Threat Intelligence

recommends the following mitigations, which are effective against a range of phishing threats, including those that may use AI-generated code.

- Review our recommended settings for Exchange Online Protection and Microsoft Defender for Office 365.
- Configure Microsoft Defender for Office 365 to recheck links on click. Safe Links provides URL scanning and rewriting of inbound email messages in mail flow, and time-of-click verification of URLs and links in email messages, other Microsoft 365 applications such as Teams, and other locations such as SharePoint Online. Safe Links scanning occurs in addition to the regular anti-spam and anti-malware protection in inbound email messages in Microsoft Exchange Online Protection (EOP). Safe Links scanning can help protect your organization from malicious links used in phishing and other attacks.
- Turn on Zero-hour auto purge (ZAP) in Defender for Office 365 to quarantine sent mail in response to newly-acquired threat intelligence and retroactively neutralize malicious phishing, spam, or malware messages that have already been delivered to mailboxes.
- Encourage users to use Microsoft Edge and other web browsers that support Microsoft Defender SmartScreen, which identifies and blocks malicious websites, including phishing sites, scam sites, and sites that host malware.
- Turn on cloud-delivered protection in Microsoft Defender Antivirus or the equivalent for your antivirus product to cover rapidly evolving attack tools and techniques. Cloud-based machine learning protections block a majority of new and unknown variants
- Configure Microsoft Entra with increased security.
- Pilot and deploy phishing-resistant authentication methods for users.
- Implement Entra ID Conditional Access authentication strength to require phishing-resistant authentication for employees and external users for critical apps.

Microsoft Defender XDR detections

Microsoft Defender XDR customers can refer to the list of applicable detections below. Microsoft Defender XDR coordinates detection, prevention, investigation, and response across endpoints, identities, email, apps to provide integrated protection against attacks like the threat discussed in this blog.

Customers with provisioned access can also use Microsoft Security Copilot in Microsoft Defender to investigate and respond to incidents, hunt for threats, and protect their organization with relevant threat intelligence.

Tactic	Observed activity	Microsoft Defender coverage
Initial access	-Phishing emails sent from a compromised small business email account. -Phishing emails contained an attached SVG file.	–**Microsoft Defender for Office 365** tenant admins can use Threat Explorer to query associated SVG file attachments using file type, file extension, or attachment file name fields. The rule description from Threat Explorer is: *This SVG has traits consistent with credential phishing campaigns*.

		–**Microsoft Defender XDR** Malicious email-sending activity from a risky user
Execution	-Embedded JavaScript within the attached SVG file executed upon opening in a browser.	
Defense evasion	-Obfuscation using invisible SVG elements and encoded business terminology. -Fake CAPTCHA, browser fingerprinting, and session tracking used to evade detection.	
Impact	-Potential credential theft if targeted user completes the phishing flow.	–**Microsoft Defender XDR** Risky sign in attempt following a possible phishing campaign

Microsoft Security Copilot

Security Copilot customers can use the standalone experience to create their own prompts or run the following prebuilt promptbooks to automate incident response or investigation tasks related to this threat:

- Incident investigation
- Microsoft User analysis
- Threat actor profile
- Threat Intelligence 360 report based on MDTI article
- Vulnerability impact assessment

Note that some promptbooks require access to plugins for Microsoft products such as Microsoft Defender XDR or Microsoft Sentinel.

Hunting queries

Microsoft Sentinel

Microsoft Sentinel customers can use the TI Mapping analytics (a series of analytics all prefixed with 'TI map') to automatically match the malicious domain indicators mentioned in this blog post with data in their workspace. If the TI Map analytics are not currently deployed, customers can install the Threat Intelligence solution from the Microsoft Sentinel Content Hub to have the analytics rule deployed in their Sentinel workspace.

Below are the queries using Sentinel Advanced Security Information Model (ASIM) functions to hunt threats across both Microsoft first party and third-party data sources. ASIM also supports deploying parsers to specific workspaces from GitHub using an ARM template or manually.

Detect network domain indicators of compromise using ASIM

The following query checks IP addresses and domain IOCs across data sources supported by ASIM network session parser:

```
//Domain list- _Im_NetworkSession
let lookback = 30d;
let ioc_ip_addr = dynamic([]);
let ioc_domains = dynamic(["kmnl.cpfcenters.de"]);
_Im_NetworkSession(starttime=todatetime(ago(lookback)), endtime=now())
| where DstDomain has_any (ioc_domains)
| summarize imNWS_mintime=min(TimeGenerated), imNWS_maxtime=max(TimeGenerated),
EventCount=count() by SrcIpAddr, DstIpAddr, DstDomain, Dvc, EventProduct, EventVendor
```

Detect domain and URL indicators of compromise using ASIM

The following query checks domain and URL IOCs across data sources supported by ASIM web session parser:

```
// Domain list - _Im_WebSession
let ioc_domains = dynamic(["kmnl.cpfcenters.de”]);
_Im_WebSession (url_has_any = ioc_domains)
```

Indicators of compromise

Indicator	Type	Description	First seen	Last seen
kmnl[.]cpfcenters[.]de	Domain	Domain hosting phishing content	08/18/2025	08/18/2025
23mb – PDF- 6 Pages[.]svg	File name	File name of SVG attachment	08/18/2025	08/18/2025

Learn more

For the latest security research from the Microsoft Threat Intelligence community, check out the Microsoft Threat Intelligence Blog.

To get notified about new publications and to join discussions on social media, follow us on LinkedIn, X (formerly Twitter), and Bluesky.

To hear stories and insights from the Microsoft Threat Intelligence community about the ever-evolving threat landscape, listen to the Microsoft Threat Intelligence podcast.