Hunting for Barium using Azure Sentinel

techcommunity.microsoft.com/t5/azure-sentinel/hunting-for-barium-using-azure-sentinel/ba-p/1875913

November 11, 2020

Leveraging Indictors of Compromise (IOC) and searching historical data for attack patterns is one of the primary responsibilities of a security monitoring team. Relevant security data for threat hunting/investigation related to an enterprise is produced in multiple locations - cloud, on-premises, and being able to analyze all the data from a single point makes it easier to spot trends and attack. Azure Sentinel has made it super easy to collect data from multiple logs across different environments and run KQL queries of recently released threat indicators across this entire data set. For example, through its recently released Microsoft 365 Defender connector security teams can now easily ingest Microsoft 365 data into Azure Sentinel allowing correlation of M365 raw logs with Sentinel's additional data sources to provide additional insights for investigations, hunting and alerts. In this blog post we share some of the IOC's related to one such threat actor that Microsoft tracks as Barium and the sample Azure Sentinel queries related to it that leverage multiple logs including those coming from Microsoft 365 Defender connector .

About a month and a half ago, the US Department of Justice, Office of Public Affairs, released documents detailing work done by a range of public and private sector organizations, including Microsoft, to disrupt Barium's cyberattack infrastructure. Barium is intent on compromising research and development (R&D) heavy organizations in telecom, high tech, computer, and healthcare; their financially motivated operations have focused primarily on the video game industry. The techniques that the group have used in the past have varied from using malicious .lnk files, Word and PowerPoint macros, to the use of open-source tools like Cobalt strike to achieve their objective. The group has been pretty prolific in the use of "supply chain" attacks to compromise software providers and then modify the providers' code to facilitate further intrusions against its customers. They have also been seen using C2 "dead drops," which are seemingly legitimate web pages that have encoded instructions to their malware. In their campaign they have also been seen using typo-squatted domains to impersonate legitimate companies and products.

Microsoft Threat Intelligence Center (MSTIC) along with partner teams have been tracking and gathering information on Barium, monitoring the group's activities as they operate a number of websites, domains and internet-connected computers. You can read more about this in the September 2020 Microsoft Digital Defense report.

MSTIC has now shared many of these indicators (IP/domains) so that you can hunt for them in Azure Sentinel using relevant data like the newly integrated Microsoft 365 Defender data, DNS logs, Firewall data etc. Microsoft 365 E5 customers, now also have the advantage of getting Azure credits towards Microsoft 365 data ingestion into Azure Sentinel - see https://aka.ms/m365-sentinel-offer for details. With this new offer, you can take advantage of end-to-end integrated security and save significant costs when ingesting Microsoft 365 data into Azure Sentinel.

Below are sample Azure Sentinel queries that you can run to check for Barium activity in your environment.

Barium IP Indicators

id: 6ee72a9e-2e54-459c-bc9a-9c09a6502a63

name: Known Barium IP

description: |

'Identifies a match across various data feeds for IP IOCs related to the Barium activity group.

References: https://www.justice.gov/opa/pr/seven-international-cyber-defendants-including-apt41-actors-charged-c...'

```
severity: High
requiredDataConnectors:
  - connectorId: Office365
    dataTypes:
     - OfficeActivity
  - connectorId: DNS
    dataTypes:
      - DnsEvents
  - connectorId: AzureMonitor(VMInsights)
    dataTypes:
      - VMConnection
  - connectorId: CiscoASA
    dataTypes:
      - CommonSecurityLog
  - connectorId: PaloAltoNetworks
    dataTypes:
      - CommonSecurityLog
  - connectorId: SecurityEvents
    dataTypes:
      - SecurityEvent
  - connectorId: AzureActiveDirectory
    dataTypes:
      - SigninLogs
  - connectorId: AzureMonitor(WireData)
    dataTypes:
      - WireData
  - connectorId: AzureMonitor(IIS)
    dataTypes:
      - W3CIISLog
  - connectorId: AzureActivity
    dataTypes:
      - AzureActivity
  - connectorId: AWS
    dataTypes:
      - AWSCloudTrail
  - connectorId: Microsoft 365 Defender
    dataTypes:
      - DeviceNetworkEvents
queryFrequency: 1d
```

```
queryPeriod: 1d
triggerOperator: gt
triggerThreshold: 0
tactics:
  - CommandAndControl
query:  |
let timeframe = 1d;
  let IPList = dynamic(["216.24.185.74", "107.175.189.159", "192.210.132.102", "67.230.163.214",
"199.19.110.240", "107.148.130.176", "154.212.129.218", "172.86.75.54", "45.61.136.199",
"149.28.150.195", "108.61.214.194", "144.202.98.198", "149.28.84.98", "103.99.209.78",
"45.61.136.2", "176.122.162.149", "192.3.80.245", "149.28.23.32", "107.182.18.149", "107.174.45.134",
"149.248.18.104", "65.49.192.74", "156.255.2.154", "45.76.6.149", "8.9.11.130", "140.238.27.255",
"107.182.24.70", "176.122.188.254", "192.161.161.108", "64.64.234.24", "104.224.185.36",
"104.233.224.227", "104.36.69.105", "119.28.139.120", "161.117.39.130", "66.42.100.42", "45.76.31.159",
"149.248.8.134", "216.24.182.48", "66.42.103.222", "218.89.236.11", "180.150.227.249", "47.75.80.23",
 "124.156.164.19", "149.248.62.83", "150.109.76.174", "222.209.187.207", "218.38.191.38",
"119.28.226.59", "66.42.98.220", "74.82.201.8", "173.242.122.198", "45.32.130.72", "89.35.178.10",
"89.43.60.113"]);
  (union isfuzzy=true
  (CommonSecurityLog
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(SourceIP) or isnotempty(DestinationIP)
  | where SourceIP in (IPList) or DestinationIP in (IPList) or Message has_any (IPList)
  | extend IPMatch = case(SourceIP in (IPList), "SourceIP", DestinationIP in (IPList), "DestinationIP",
"Message")
  | summarize StartTimeUtc = min(TimeGenerated), EndTimeUtc = max(TimeGenerated) by
SourceIP, DestinationIP, DeviceProduct, DeviceAction, Message, Protocol, SourcePort, DestinationPort,
DeviceAddress, DeviceName, IPMatch
  | extend timestamp = StartTimeUtc, IPCustomEntity = case(IPMatch == "SourceIP", SourceIP,
IPMatch == "DestinationIP", DestinationIP, "IP in Message Field")
  ),
  (OfficeActivity
  | where TimeGenerated >= ago(timeframe)
  |extend SourceIPAddress = ClientIP, Account = UserId
  | where  SourceIPAddress in (IPList)
  | extend timestamp = TimeGenerated , IPCustomEntity = SourceIPAddress ,
AccountCustomEntity = Account
  ),
  (DnsEvents
```

```
  | where TimeGenerated >= ago(timeframe)
  | extend DestinationIPAddress = IPAddresses,  Host = Computer
  | where  DestinationIPAddress has_any (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity = DestinationIPAddress,
HostCustomEntity = Host
  ),
  (VMConnection
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(SourceIp) or isnotempty(DestinationIp)
  | where SourceIp in (IPList) or DestinationIp in (IPList)
  | extend IPMatch = case( SourceIp in (IPList), "SourceIP", DestinationIp in (IPList),
"DestinationIP", "None")
  | extend timestamp = TimeGenerated , IPCustomEntity = case(IPMatch == "SourceIP",
 SourceIp, IPMatch == "DestinationIP", DestinationIp, "None"), Host = Computer
  ),
  (Event
  | where TimeGenerated >= ago(timeframe)
  | where Source == "Microsoft-Windows-Sysmon"
  | where EventID == 3
  | extend EvData = parse_xml(EventData)
  | extend EventDetail = EvData.DataItem.EventData.Data
  | extend SourceIP = EventDetail.[9].["#text"], DestinationIP = EventDetail.[14].["#text"]
  | where SourceIP in (IPList) or DestinationIP in (IPList)
  | extend IPMatch = case( SourceIP in (IPList), "SourceIP", DestinationIP in (IPList), "DestinationIP", "None")
  | extend timestamp = TimeGenerated, AccountCustomEntity = UserName, HostCustomEntity = Computer ,
IPCustomEntity = case(IPMatch == "SourceIP", SourceIP, IPMatch == "DestinationIP", DestinationIP, "None")
  ),
  (WireData
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(RemoteIP)
  | where RemoteIP in (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity = RemoteIP, HostCustomEntity = Computer
  ),
  (SigninLogs
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(IPAddress)
  | where IPAddress in (IPList)
  | extend timestamp = TimeGenerated, AccountCustomEntity = UserPrincipalName,
IPCustomEntity = IPAddress
```

```
  ),
  (W3CIISLog
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(cIP)
  | where cIP in (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity = cIP, HostCustomEntity = Computer,
AccountCustomEntity = csUserName
  ),
  (AzureActivity
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(CallerIpAddress)
  | where CallerIpAddress in (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity = CallerIpAddress, AccountCustomEntity = Caller
  ),
  (
  AWSCloudTrail
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(SourceIpAddress)
  | where SourceIpAddress in (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity =
SourceIpAddress, AccountCustomEntity = UserIdentityUserName
  ),
  (
  DeviceNetworkEvents
  | where TimeGenerated >= ago(timeframe)
  | where isnotempty(RemoteIP)
  | where RemoteIP in (IPList)
  | extend timestamp = TimeGenerated, IPCustomEntity = RemoteIP, HostCustomEntity = DeviceName
  )
  )
```

Barium Domain Indicators

```
id: 70b12a3b-4899-42cb-910c-5ffaf9d7997d
name: Known Barium domains
description: |
  'Identifies a match across various data feeds for domains IOCs related to the Barium activity group.'
   References: https://www.justice.gov/opa/pr/seven-international-cyber-defendants-including-apt41-actors-charged-connectio...
severity: High
requiredDataConnectors:
```

```
  - connectorId: DNS
    dataTypes:
      - DnsEvents
  - connectorId: AzureMonitor(VMInsights)
    dataTypes:
      - VMConnection
  - connectorId: CiscoASA
    dataTypes:
      - CommonSecurityLog
  - connectorId: PaloAltoNetworks
    dataTypes:
      - CommonSecurityLog
  - connectorId: Microsoft 365 Defender
    dataTypes:
      - DeviceNetworkEvents
queryFrequency: 1d
queryPeriod: 1d
triggerOperator: gt
triggerThreshold: 0
tactics:
  - CommandAndControl
query:  |
let timeframe = 1d;
let DomainNames = dynamic(["0.ns1.dns-info.gq", "1.ns1.dns-info.gq", "10.ns1.dns-info.gq", "102.ns1.dns-info.gq",
"104.ns1.dns-info.gq", "11.ns1.dns-info.gq", "110.ns1.dns-info.gq", "115.ns1.dns-info.gq", "116.ns1.dns-info.gq",
"117.ns1.dns-info.gq", "118.ns1.dns-info.gq", "12.ns1.dns-info.gq", "120.ns1.dns-info.gq", "122.ns1.dns-info.gq",
"123.ns1.dns-info.gq", "128.ns1.dns-info.gq", "13.ns1.dns-info.gq", "134.ns1.dns-info.gq", "135.ns1.dns-info.gq",
"138.ns1.dns-info.gq", "14.ns1.dns-info.gq", "144.ns1.dns-info.gq", "15.ns1.dns-info.gq", "153.ns1.dns-info.gq",
"157.ns1.dns-info.gq", "16.ns1.dns-info.gq", "17.ns1.dns-info.gq", "18.ns1.dns-info.gq", "19.ns1.dns-info.gq",
"1a9604fa.ns1.feedsdns.com", "1c7606b6.ns1.steamappstore.com", "2.ns1.dns-info.gq", "20.ns1.dns-info.gq",
"201.ns1.dns-info.gq", "202.ns1.dns-info.gq", "204.ns1.dns-info.gq", "207.ns1.dns-info.gq", "21.ns1.dns-info.gq",
"210.ns1.dns-info.gq", "211.ns1.dns-info.gq", "216.ns1.dns-info.gq", "22.ns1.dns-info.gq", "220.ns1.dns-info.gq",
"223.ns1.dns-info.gq", "23.ns1.dns-info.gq", "24.ns1.dns-info.gq", "25.ns1.dns-info.gq", "26.ns1.dns-info.gq",
"27.ns1.dns-info.gq", "28.ns1.dns-info.gq", "29.ns1.dns-info.gq", "3.ns1.dns-info.gq", "30.ns1.dns-info.gq",
"31.ns1.dns-info.gq", "32.ns1.dns-info.gq", "33.ns1.dns-info.gq", "34.ns1.dns-info.gq", "35.ns1.dns-info.gq",
"36.ns1.dns-info.gq", "37.ns1.dns-info.gq", "39.ns1.dns-info.gq", "3d6fe4b2.ns1.steamappstore.com",
"4.ns1.dns-info.gq", "40.ns1.dns-info.gq", "42.ns1.dns-info.gq", "43.ns1.dns-info.gq", "44.ns1.dns-info.gq",
"45.ns1.dns-info.gq", "46.ns1.dns-info.gq", "48.ns1.dns-info.gq", "5.ns1.dns-info.gq", "50.ns1.dns-info.gq",
"50417.service.gstatic.dnset.com", "51.ns1.dns-info.gq", "52.ns1.dns-info.gq", "53.ns1.dns-info.gq",
```

"54.ns1.dns-info.gq", "55.ns1.dns-info.gq", "56.ns1.dns-info.gq", "57.ns1.dns-info.gq", "58.ns1.dns-info.gq",
"6.ns1.dns-info.gq", "60.ns1.dns-info.gq", "62.ns1.dns-info.gq", "63.ns1.dns-info.gq", "64.ns1.dns-info.gq",
"65.ns1.dns-info.gq", "67.ns1.dns-info.gq", "7.ns1.dns-info.gq", "70.ns1.dns-info.gq", "71.ns1.dns-info.gq",
"73.ns1.dns-info.gq", "77.ns1.dns-info.gq", "77075.service.gstatic.dnset.com", "7c1947fa.ns1.steamappstore.com",
"8.ns1.dns-info.gq", "81.ns1.dns-info.gq", "86.ns1.dns-info.gq", "87.ns1.dns-info.gq", "9.ns1.dns-info.gq",
"94343.service.gstatic.dnset.com", "9939.service.gstatic.dnset.com", "aa.ns.mircosoftdoc.com",
"aaa.feeds.api.ns1.feedsdns.com", "aaa.googlepublic.feeds.ns1.dns-info.gq",
"aaa.resolution.174547._get.cache.up.sourcedns.tk", "acc.microsoftonetravel.com",
"accounts.longmusic.com", "admin.dnstemplog.com", "agent.updatenai.com",
"alibaba.zzux.com", "api.feedsdns.com", "app.portomnail.com", "asia.updatenai.com",
"battllestategames.com", "bguha.serveuser.com", "binann-ce.com", "bing.dsmtp.com",
"blog.cdsend.xyz", "brives.minivineyapp.com", "bsbana.dynamic-dns.net",
"californiaforce.000webhostapp.com", "californiafroce.000webhostapp.com",
"cdn.freetcp.com", "cdsend.xyz", "cipla.zzux.com", "cloudfeeddns.com", "comcleanner.info",
"cs.microsoftsonline.net", "dns-info.gq", "dns05.cf", "dns22.ml", "dns224.com",
"dnsdist.org", "dnstemplog.com", "doc.mircosoftdoc.com", "dropdns.com",
"eshop.cdn.freetcp.com", "exchange.dumb1.com", "exchange.misecure.com", "exchange.mrbasic.com",
"facebookdocs.com", "facebookint.com", "facebookvi.com", "feed.ns1.dns-info.gq", "feedsdns.com",
"firejun.freeddns.com", "ftp.dns-info.dyndns.pro", "goallbandungtravel.com", "goodhk.azurewebsites.net",
"googlepublic.feed.ns1.dns-info.gq", "gp.spotifylite.cloud", "gskytop.com", "gstatic.dnset.com",
"gxxservice.com", "helpdesk.cdn.freetcp.com", "id.serveuser.com", "infestexe.com", "item.itemdb.com",
"m.mircosoftdoc.com", "mail.transferdkim.xyz", "mcafee.updatenai.com", "mecgjm.mircosoftdoc.com",
"microdocs.ga", "microsock.website", "microsocks.net", "microsoft.sendsmtp.com",
"microsoftbook.dns05.com", "microsoftcontactcenter.com", "microsoftdocs.dns05.com", "microsoftdocs.ml",
"microsoftonetravel.com", "microsoftonlines.net", "microsoftprod.com", "microsofts.dns1.us", "microsoftsonline.net",
"minivineyapp.com", "mircosoftdoc.com", "mircosoftdocs.com", "mlcrosoft.ninth.biz", "mlcrosoft.site",
"mm.portomnail.com", "msdnupdate.com", "msecdn.cloud", "mtnl1.dynamic-dns.net", "ns.gstatic.dnset.com",
"ns.microsoftprod.com", "ns.steamappstore.com", "ns1.cdn.freetcp.com", "ns1.comcleanner.info", "ns1.dns-info.gq",
"ns1.dns05.cf", "ns1.dnstemplog.com", "ns1.dropdns.com", "ns1.microsoftonetravel.com",
"ns1.microsoftonlines.net", "ns1.microsoftprod.com", "ns1.microsoftsonline.net", "ns1.mlcrosoft.site",
"ns1.teams.wikaba.com", "ns1.windowsdefende.com", "ns2.comcleanner.info", "ns2.dnstemplog.com",
"ns2.microsoftonetravel.com", "ns2.microsoftprod.com", "ns2.microsoftsonline.net", "ns2.mlcrosoft.site",
"ns2.windowsdefende.com", "ns3.microsoftprod.com", "ns3.mlcrosoft.site", "nutrition.mrbasic.com",
"nutrition.youdontcare.com", "online.mlcrosoft.site", "online.msdnupdate.com", "outlookservce.site",
"owa.jetos.com", "owa.otzo.com", "pornotime.co", "portomnail.com",
"post.1a0.066e063ac.7c1947fa.ns1.steamappstore.com", "pricingdmdk.com", "prod.microsoftprod.com",
"product.microsoftprod.com", "ptcl.yourtrap.com", "query.api.sourcedns.tk", "rb.itemdb.com", "redditcdn.com",
"rss.otzo.com", "secure.msdnupdate.com", "service.dns22.ml", "service.gstatic.dnset.com", "service04.dns04.com",
"settings.teams.wikaba.com", "sip.outlookservce.site", "sixindent.epizy.com", "soft.msdnupdate.com", "sourcedns.ml",

```
"sourcedns.tk", "sport.msdnupdate.com", "spotifylite.cloud", "static.misecure.com", "steamappstore.com",
"store.otzo.com", "survey.outlookservce.site", "team.itemdb.com", "temp221.com", "test.microsoftprod.com",
"thisisaaa.000webhostapp.com", "token.dns04.com", "token.dns05.com", "transferdkim.xyz",
"travelsanignacio.com", "update08.com", "updated08.com", "updatenai.com", "wantforspeed.com",
 "web.mircosoftdoc.com", "webmail.pornotime.co", "webwhois.team.itemdb.com", "windowsdefende.com", "wnswindows.com",
 "ashcrack.freetcp.com", "battllestategames.com", "binannce.com", "cdsend.xyz", "comcleanner.info", "microsock.website",
"microsocks.net", "microsoftsonline.net", "mlcrosoft.site", "notify.serveuser.com", "ns1.microsoftprod.com",
"ns2.microsoftprod.com", "pricingdmdk.com", "steamappstore.com", "update08.com", "wnswindows.com",
"youtube.dns05.com", "z1.zalofilescdn.com", "z2.zalofilescdn.com", "zalofilescdn.com"]);
 (union isfuzzy=true
 (CommonSecurityLog
 | where TimeGenerated >= ago(timeframe)
 | parse Message with * '(' DNSName ')' *
 | where DNSName in~ (DomainNames)
 | extend Account = SourceUserID, Computer = DeviceName, IPAddress =  DestinationIP
 ),
 (DnsEvents
 | where TimeGenerated >= ago(timeframe)
 | extend DNSName = Name
 | where isnotempty(DNSName)
 | where DNSName  in~ (DomainNames)
 | extend IPAddress =  ClientIP
 ),
 (VMConnection
 | where TimeGenerated >= ago(timeframe)
 | parse RemoteDnsCanonicalNames with * '["' DNSName '"]' *
 | where isnotempty(DNSName)
 | where DNSName  in~ (DomainNames)
 | extend IPAddress = RemoteIp
 ),
 (
  DeviceNetworkEvents
 | where isnotempty(RemoteUrl)
 | where RemoteUrl  in~ (DomainNames)
 | extend IPAddress = RemoteIP
 | extend Computer = DeviceName
 )
 )
 | extend timestamp = TimeGenerated, AccountCustomEntity = Account, HostCustomEntity = Computer, IPCustomEntity = IPAddress
```

References:

https://www.justice.gov/opa/pr/seven-international-cyber-defendants-including-apt41-actors-charged-c...

https://blogs.microsoft.com/on-the-issues/2020/09/29/microsoft-digital-defense-report-cyber-threats/

https://docs.microsoft.com/en-us/azure/sentinel/connect-microsoft-365-defender

https://aka.ms/m365-sentinel-offer

https://techcommunity.microsoft.com/t5/azure-sentinel/what-s-new-microsoft-365-defender-connector-no...