Advanced IP Scanner: the preferred scanner in the A(P)T toolbox

huntandhackett.com/blog/advanced-ip-scanner-the-preferred-scanner-in-the-apt-toolbox

Krijn de Mik @
Oct 22, 2021 10:23:58 AM

1. Introduction

Hunt & Hackett has been working on a wide variety of targeted ransomware cases. During these targeted ransomware cases, 'Advanced IP Scanner' (AIS)[1] was regularly used as reconnaissance tool for Active Scanning (T1595) and Network Service Scanning (T1046). This has not only been observed by Hunt & Hackett, but also by other incident response parties.

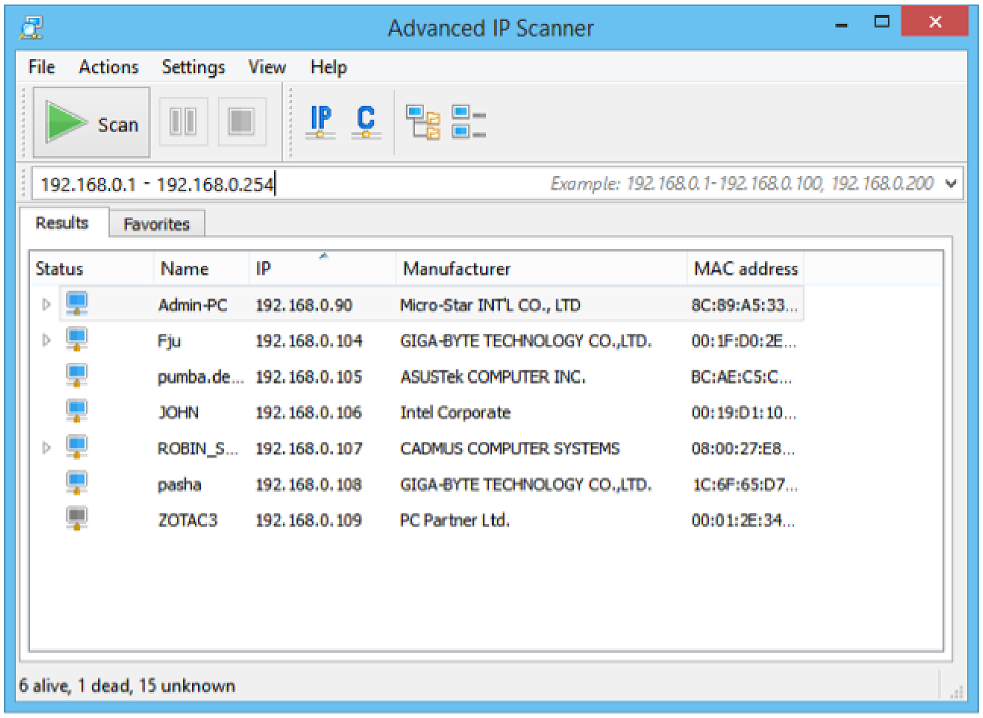
Groups that have (had) used Advanced IP Scanner include:

- Conti[2]
- Darkside/UNC2465[3]
- Egregor[4]
- Hades/ Evilcorp[5]
- REvil[6]
- Ryuk/ UNC1878[7]
- UNC2447[7]
- UNC Iranian actor[8]
- Dharma[9]

This small write-up focuses on some of the forensic traces left by AIS that Hunt & Hackett observed during Incident Response cases. The artefacts might be useful during an investigation, and can shine some (minor) light on threat actors' activities. Furthermore, this blogpost provides some pointers related to detecting Advanced IP Scanner.

2. Advanced IP Scanner

Advanced IP Scanner (AIS) is freely available online[1] and can be executed as an installer and as a portable version. Both have been used by threat actors. After the installation / execution of AIS, the end user is presented with an overview as shown in *Figure 1*.



192.168.0.1 - 192.168.0.254

Example: 192.168.0.1-192.168.0.100, 192.168.0.200

Status	Name	IP	Manufacturer	MAC address
	Admin-PC	192.168.0.90	Micro-Star INT'L CO., LTD	8C:89:A5:33...
	Fju	192.168.0.104	GIGA-BYTE TECHNOLOGY CO.,LTD.	00:1F:D0:2E...
	pumba.de...	192.168.0.105	ASUSTek COMPUTER INC.	BC:AE:C5:C...
	JOHN	192.168.0.106	Intel Corporate	00:19:D1:10...
	ROBIN_S...	192.168.0.107	CADMUS COMPUTER SYSTEMS	08:00:27:E8...
	pasha	192.168.0.108	GIGA-BYTE TECHNOLOGY CO.,LTD.	1C:6F:65:D7...
	ZOTAC3	192.168.0.109	PC Partner Ltd.	00:01:2E:34...

6 alive, 1 dead, 15 unknown

Figure 1 - GUI of Advanced IP Scanner.

AIS is a simple and user-friendly IP scanner, which provides the end user with a concise overview of the systems found in the network. From a high-level perspective, AIS provides the user with the following functionality:

1. IP scanning: scan the given range for systems that are alive, or dead;
2. Tools: provide functionality to ping an IP, perform a Tracert, or connect with Telnet / SSH;
3. Remote connections: connect to the corresponding IP address via HTTP, HTTPS, FTP, RDP, or RADMIN (only works if RADMIN is installed).
4. (Re)boot/ Wake-On-Lan: the user can boot / reboot a system if the user is authorized to do so, or can send Wake-On-Lan (WOL) packages to a system. If enabled, this provides the user with functionality to remotely boot a system.

The fact that there is a portable version of Advanced IP Scanner, that it has a GUI and that the tool supports a variety of ways to interact with identified systems probably contributes to it popularity.

3. Forensic traces

AIS leaves traces on a host system when it is executed. This section describes the traces that are created during usage of the AIS tool and more specifically, the Windows registry keys that are being created. Do note that for both the portable version as well as the installer version, the same traces are created in the Windows registry.

3.1 AIS registry keys

After the installation of AIS, keys and subkeys are created in the `HKEY_USERS` hive of the user under which account AIS was installed. Data is more specifically stored at the following location:

```
Computer\HKEY_USERS\<SID>\SOFTWARE\famatech\advanced_ip_scanner
```

The mentioned registry keys are created, by both the installer version of AIS and the portable version, after the first usage of the application and subsequently some (sub)keys are updated after using AIS. While looking at it graphically, multiple keys and subkeys are created as shown in *Figure 2*. The most relevant traces from a forensic perspective are either stored under the ‘ `advanced_ip_scanner` ’, or the ‘ `State` ’ key.

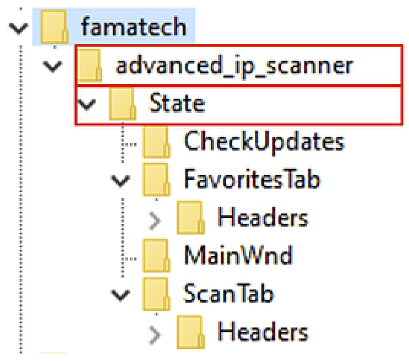


Figure 2 – Overview of keys created with regards to AIS.

3.2 The 'advanced_ip_scanner' keys and subkeys

If we look at the subkeys of the `advanced_ip_scanner` key, we are presented with the keys as shown in *Table 2*. This table shows the name and type of the subkeys, together with the value (data) of the subkey, a comment with what the subkey represents and what the function is within AIS. Especially the `locale` subkey might be of relevance, since this could potentially reveal something about the background of the threat actor, as well as the `locale_timestamp` key since this gives an indication of when the application has been used for the first time.

Name	Type	Data	Comment
`locale`	REG_SZ	en_us	Concerns the language settings of AIS as set by the user.
`locale_timestamp`	REG_SZ	1629953831432	The first time the application has been started in epoch time, which translates to UTC +0. After starting the application for the first time, Hunt & Hackett hasn't observed any updates to this value in subsequent restarts of the application.
`show_alive`	REG_SZ	true	Indicates whether found hosts, which are probably reachable, with the status 'alive' should be represented in the GUI. This is an option in the 'View' menu.
`show_dead`	REG_SZ	true	Indicates whether found hosts, which aren't reachable, with the status 'alive' should be represented in the GUI. This is an option in the 'View' menu.
`show_unknown`	REG_SZ	false	Indicates whether found hosts with the status 'unknown' should be represented in the GUI. This is an option in the 'View' menu.

Table 2: Overview of the AIS registry key 'advanced_ip_scanner' under HKEY_USERS hive.

3.3 The 'advanced_ip_scanner\State' keys and subkeys

A variety of subkeys are listed under the key `advanced_ip_scanner/State`, as shown in *Table 3*. This table shows the name of subkeys together with the type of subkeys, the value (data) of the subkey, a comment with what the subkey represents and what the function is within AIS. All the keys, except for `LastRangeUsed` are created upon the first time the application is started. The key `LastRangeUsed` is created upon the first time a scan is performed with AIS.

The three registry keys `IpRangesMruList`, `LastRangeUsed` and `SearchMruList` under the key `advanced_ip_scanner\State` are especially interesting, as they could be of relevance during an IR assignment.

- `LastRangeUsed`: this key represents the last scan that is being performed and shows the range that has been scanned.

- `IpRangesMruList` : this key represents all ranges scanned by AIS, including the frequency. Every range scanned has a prefix of `[digit]-[digit]` , followed by `[start IP address]-[IP range]` . The first digit of the prefix represents the number of times a range has been scanned, which will be increased after every subsequent scan. The range `192.168.227.1-254` in *Table 3* has for example been scanned five times. It's unknown what the second digit represents, since this digit remained 'static' throughout the different tests performed. The scanned IP-range is stored in memory while the application is still running. Upon closure of the application, the data is written from a buffer in memory to the registry. The order in which the scanned IP ranges are stored is unknown.
- `SearchMruList` : This key contains an overview of all IP-addresses searched for by the user. These are added upon closure of the AIS application. Besides the last searched IP address, also other historical searches are saved here. Do note that a similar prefix is added to IP address, just like for the key `IpRangesMruList` . However, it's unknown what this prefix represents.

Name	Type	Data	Comment
`IpRangesMruList`	REG_SZ	5-1 192.168.227.1-254 2-1 192.168.250.1-254 1-1 192.168.116.1-254 1-1 192.168.228.1-254 1-1 192.168.240.1-254 1-1 192.168.230.1-254	All ranges scanned by the tool are stored in this subkey. Additionally, the first digit of the prefix indicates how frequent the range has been scanned.
`LastActiveTab`	REG_DWORD	0x00000000 (0)	The tab (either 'results' or 'favorites') that was active upon closure of AIS process.
`LastRangeUsed`	REG_SZ	192.168.226.1-254	This key is created after executing the first scan.
`lock_toolbars`	REG_SZ	true	A menu item that could be selected to lock the toolbars
`results_col_size_init`	REG_SZ	true	Unknown what this key represents.
`SearchMruList`	REG_SZ	(empty)	The IP-addresses searched for via the GUI search field. A prefix is added to the searched IP represented as `[digit]-[digit]` . It's unknown to Hunt & Hackett what the prefix represents.
`window_state`	REG_BINARY	40 00 42 00 79 00 74 00 64 00 41 00 …	Holds the state (dimensions) of the application windows upon closure of the application.

Table 3 - Overview of the AIS registry State key under HKEY_USERS hive.

4. Detection

Detection of the Advanced IP Scanner process is trivial. Here at Hunt & Hackett, we also try to come up with better, behavioral detection methods that do not rely on static signatures for individual applications.

4.1 Port scanning detection using Chronicle

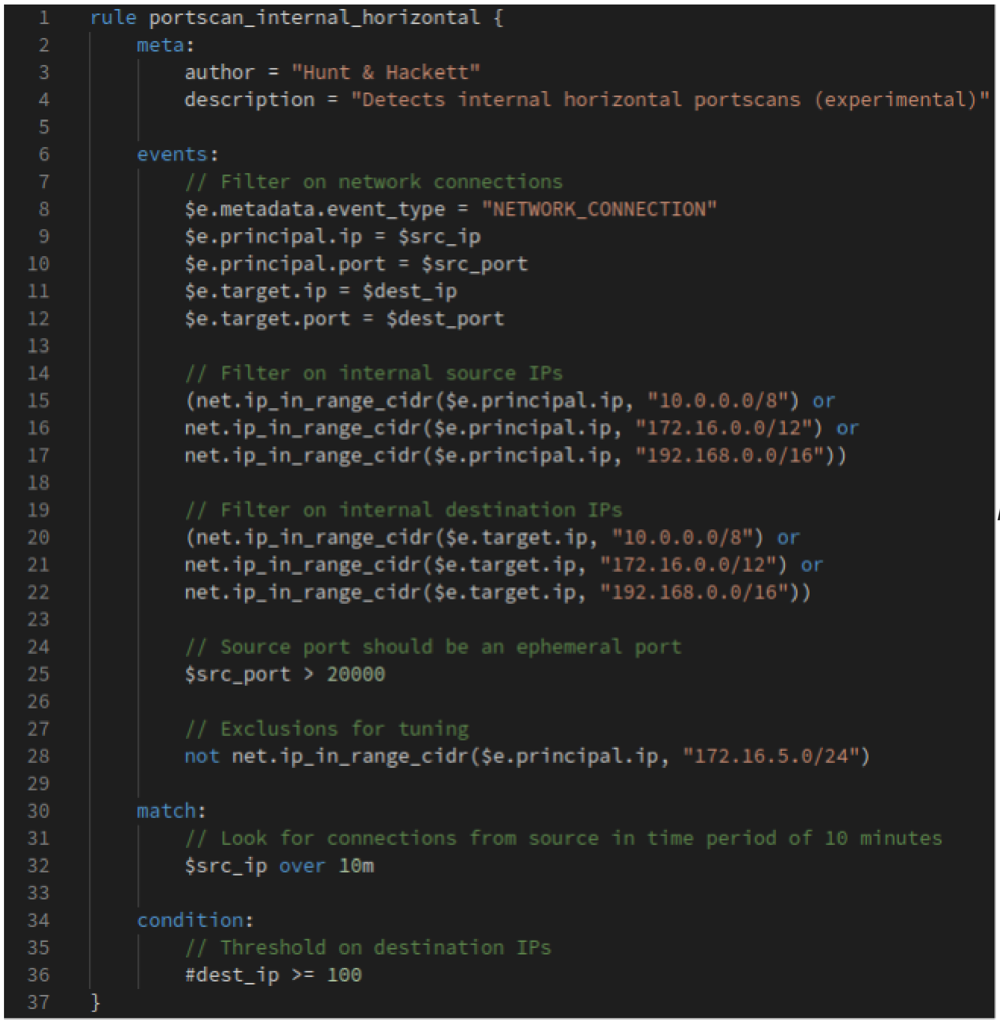
Detection should not only focus on the characteristics of individual pieces of software, but also on behavior of applications. This means we also try to identify internal port scanning itself, and not merely the execution of Advanced IP Scanner. As part of our detection engineering efforts, we experiment with custom YARA-L[10] rules for Chronicle[11] to detect internal port scans.

Experimenting with horizontal port scanning detection

Internal port scanning of entire ranges results in a large number of network connections from the scanning machine to other machines in a corporate network. We can use build-in functionality of Chronicle to experiment with YARA-L rules capable of detecting internal scanning behavior.

In large enough networks, we can attempt to detect hosts that have performed a large number of connection attempts to other internal machines, which is indicative of internal port scans.

The experimental YARA-L rule depicted in *Figure 3* for example triggers when a machine has made more than 100 internal connections or connection attempts within 10 minutes. Real-world testing shows that this approach can detect (some) internal port scans, though heavy tuning is required to filter out benign machines / processes that perform internal network discovery for legitimate reasons.



```
rule portscan_internal_horizontal {
    meta:
        author = "Hunt & Hackett"
        description = "Detects internal horizontal portscans (experimental)"

    events:
        // Filter on network connections
        $e.metadata.event_type = "NETWORK_CONNECTION"
        $e.principal.ip = $src_ip
        $e.principal.port = $src_port
        $e.target.ip = $dest_ip
        $e.target.port = $dest_port

        // Filter on internal source IPs
        (net.ip_in_range_cidr($e.principal.ip, "10.0.0.0/8") or
        net.ip_in_range_cidr($e.principal.ip, "172.16.0.0/12") or
        net.ip_in_range_cidr($e.principal.ip, "192.168.0.0/16"))

        // Filter on internal destination IPs
        (net.ip_in_range_cidr($e.target.ip, "10.0.0.0/8") or
        net.ip_in_range_cidr($e.target.ip, "172.16.0.0/12") or
        net.ip_in_range_cidr($e.target.ip, "192.168.0.0/16"))

        // Source port should be an ephemeral port
        $src_port > 20000

        // Exclusions for tuning
        not net.ip_in_range_cidr($e.principal.ip, "172.16.5.0/24")

    match:
        // Look for connections from source in time period of 10 minutes
        $src_ip over 10m

    condition:
        // Threshold on destination IPs
        #dest_ip >= 100
}
```

Figure 3 - Experimental YARA-L rule for detecting internal port scans

Limitations of detection attempts

With some filtering effort, we can use Chronicle to detect internal port scans at customers, though this approach is impractical and requires heavy tuning. Additionally, our detection attempt is not foolproof and can be circumvented by a determined attacker.

Some methods of bypassing this detection attempt include:

- Slow scanning to circumvent any time-based detection thresholds.
- Scanning of small IP blocks at a time to bypass any count-based thresholds.

4.2 Port scanning detection using Canary tokens

In the previous section, we concluded that it is not feasible to detect all internal port scans using only network connections data and threshold-based detection rules. To enhance detection, we can place canary tokens at strategic places in an organization's network. Whenever a canary token is scanned, the cyber defense center will receive an alert and an analyst can investigate why a canary token was triggered.

Whenever an alert is deemed malicious, this indicates an attacker is performing active reconnaissance in an organization's network. Though, whenever a canary token is triggered, this does not automatically mean a network has been breached. Canary tokens can also be triggered by benign activities, including:

- Misspelled IP addresses or server names
- Applications that (for various reasons) contain network discovery functionality
- System administrators or other users performing network scans for legitimate reasons

5. Detection rules

This section contains an overview of different rules in both Carbon Black as well as Yara-L format.

Additionally, you can find an already existing Sigma rule in the Sigma rule repository `https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/win_advanced_ip_scanner.yml`. For completeness the Sigma rules is listed in this section as well.

5.1 Sigma

```
action: global

title: Advanced IP Scanner

status: experimental

description: Detects the use of Advanced IP Scanner. Seems to be a popular tool for ransomware groups.

references:

    - https://news.sophos.com/en-us/2019/12/09/snatch-ransomware-reboots-pcs-into-safe-mode-to-bypass-protection/

    - https://www.fireeye.com/blog/threat-research/2020/05/tactics-techniques-procedures-associated-with-maze-ransomware-incidents.html

    - https://labs.f-secure.com/blog/prelude-to-ransomware-systembc

    - https://assets.documentcloud.org/documents/20444693/fbi-pin-egregor-ransomware-bc-01062021.pdf

    - https://thedfirreport.com/2021/01/18/all-that-for-a-coinminer

author: '@ROxPinTeddy'

date: 2020/05/12

modified: 2021/05/11

tags:
    - attack.discovery
    - attack.t1046
falsepositives:
    - Legitimate administrative use
level: medium
---
id: bef37fa2-f205-4a7b-b484-0759bfd5f86f
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Image|contains: '\advanced_ip_scanner'
    condition: selection
---
id: fed85bf9-e075-4280-9159-fbe8a023d6fa
logsource:
    category: file_event
    product: windows
detection:
    selection:
        TargetFilename|contains: '\AppData\Local\Temp\Advanced IP Scanner 2'
    condition: selection
```

5.2 CarbonBlack

Action	CarbonBlack query
Advanced IP Scanner execution	`(process_name:advanced_ip_scanner.exe OR process_publisher:"Famatech Corp.") OR (process_file_description:"Advanced IP Scanner" OR process_original_filename:advanced_ip_scanner.exe OR process_internal_name:"Advanced IP Scanner" OR process_product_name:"Advanced IP Scanner")`
Advanced IP Scanner registry access	`regmod_name:*\\famatech\\advanced_ip_scanner\\*`

5.3 Chronicle

```
rule advanced_ip_scanner_execution {

   meta:

       author = "Hunt & Hackett"

       description = "Detects execution of Advanced IP Scanner"



   events:

       $event.metadata.event_type = "PROCESS_LAUNCH"



       ($event.principal.process.file.full_path = /Advanced_IP_Scanner/ nocase or

       $event.principal.process.command_line = /Advanced_IP_Scanner/ nocase or

       $event.target.process.file.full_path = /Advanced_IP_Scanner/ nocase or

       $event.target.process.command_line = /Advanced_IP_Scanner/ nocase)



     condition:

       $event

}
```

```
rule portscan_internal_horizontal {
  meta:
      author = "Hunt & Hackett"
      description = "Detects internal horizontal portscans (experimental)"

  events:
      $e.metadata.event_type = "NETWORK_CONNECTION"
      $e.principal.ip = $src_ip
      $e.principal.port = $src_port
      $e.target.ip = $dest_ip
      $e.target.port = $dest_port

      // Filter on internal source IPs
      (net.ip_in_range_cidr($e.principal.ip, "10.0.0.0/8") or
     net.ip_in_range_cidr($e.principal.ip, "172.16.0.0/12") or
      net.ip_in_range_cidr($e.principal.ip, "192.168.0.0/16"))

      // Filter on internal destination IPs
      (net.ip_in_range_cidr($e.target.ip, "10.0.0.0/8") or
      net.ip_in_range_cidr($e.target.ip, "172.16.0.0/12") or
      net.ip_in_range_cidr($e.target.ip, "192.168.0.0/16"))

      // Source port should be an ephemeral port
      $src_port > 20000

      // Exclusions for tuning
      // not net.ip_in_range_cidr($e.principal.ip, "__IP_RANGE__")

   match:
      // Look for connections from source
      // in time period of 10 minutes
      $src_ip over 10m

 condition:
      // Threshold on destination IPs
      #dest_ip >= 100
}
```

Sources