SIP Bruteforcing in the Wild - An Assessment of Adversaries, Techniques and Tools

Harm Griffioen
Hasso Plattner Institute for Digital Engineering
University of Potsdam, Germany
harm.griffioen@hpi.de

Huancheng Hu
Hasso Plattner Institute for Digital Engineering
University of Potsdam, Germany
huancheng.hu@hpi.de

Christian Doerr
Hasso Plattner Institute for Digital Engineering
University of Potsdam, Germany
christian.doerr@hpi.de

***Abstract*—Over the last two decades, Voice-over-IP (VoIP) and specifically SIP have become standard solutions to realize voice telephony in residential, commercial, and telecom environments. As by now, an abundance of SIP endpoints exist, it has become financially lucrative for cybercriminals to systematically search for VoIP installations, with for example the aim to abuse them for billing fraud or to hide their criminal activities behind a legitimate connection and phone number. By now, this has made SIP one of the most scanned UDP protocols on the Internet.**

In this paper, we take a look at the actors behind these attacks. Using a large network telescope, we collect over 822 million SIP brute-forcing attempts from 5,691 sources over 187 countries and analyze who is searching for and attacking VoIP endpoints. As each tool and campaign exhibits specific implementation differences, we can relate individual attempts into campaigns and can thereby provide a detailed view into different actors in the ecosystem, different techniques and tooling, and how these are developing over 5 years. We show that we can fingerprint different SIP scanning tools, show that actors hardly ever change their toolkit, and identify an increase in highly distributed and coordinated scanning.

I. Introduction

Voice-over-IP or VoIP provides the technical means for carrying telephony calls in the form of regular IP packets over data communication networks, thereby eliminating the need for dedicated voice lines and the associated circuit switching. This drastically reduces required resources, starting with lower bandwidth per call, but also simpler and less expensive telecommunication infrastructure. As a result, charges for telephony have plummeted, and the widespread adoption of voice-over-IP makes it possible for all-you-can-call plans to exist for landlines and mobile handsets, sometimes at prices a long-distance call had been charged per minute in the 90s.

There are two main components to Internet telephony: first, protocols such as RTP or SRTP transmit the packetized audio and video data, but before a call can be made, a signaling protocol has to locate the user, set up a call, negotiate the connection and ultimately end the session. One of the most popular protocols of this second category is the Session Initialization Protocol (SIP). Standardized in RFC 3261, it is responsible for VoIP signaling in Internet telephony but also adopted by the 3GPP for mobile phone calling over LTE. As VoIP has become the dominant solution to realize voice telephony in residential or commercial environments as well as been adopted by telecommunication networks themselves, a plethora of endpoints exists listening for incoming connections signaled by the SIP.

This ubiquity of SIP-speaking endpoints however also opens up the possibility for abuse, as misconfigured systems or those insufficiently secured may be used by cybercriminals, for example fraudulently placing long-distance calls via the phone lines of the victim [1]. Furthermore, by making malicious calls through a victim's systems, criminals could cloak their identities and origin, and even impersonate authorities or legitimate users as part of a social engineering attack [2]. To discover potentially vulnerable systems, criminals scan public networks such as the Internet for hosts responding to SIP requests, and given the wide opportunities for abuse and the plethora of potential victims, SIP has become a wide-scanned protocol on the Internet today.

While we know already much about the vulnerabilities of SIP in general and how it can be abused, we know relatively little about the adversaries who abuse it. Knowledge of the actors and the overall ecosystem, the techniques in use to scan and compromise hosts provides however important intelligence to detect malicious activities, modify systems to reduce incentives for an attack, and ultimately identify the criminals behind it. Furthermore, when we can effectively detect the tooling used by adversaries for a compromise, malicious activities can be stopped early before causing significant damage.

In this paper, we address this gap and asses the adversaries, their techniques, and tools for the exploitation of SIP endpoints. Using a large network telescope of two partially populated /16 networks located in enterprise systems, we redirect and analyze incoming SIP requests for analysis. Based on 822 million SIP connection attempts sent by 5,691 sources over 5 years, we are able to provide a detailed view into this ecosystem of cybercrime. With this work, we make the

ISBN 978-3-903176-39-3

following main contributions:

- We create a method for fingerprinting SIP scanning tools and apply this method in the wild to identify tools used to scan for SIP servers.
- We show that actors do not change their tools, but rather keep the same tool even when running their operations over multiple years.
- We show most actors use standard tools and run these without any customization.
- We show that custom-made tools are used more in highly distributed and coordinated scanning.
- We identify an increase in distributed scanning over the years, where actors aim to stay undetected.

The remainder of this paper is structured as follows: Section II provides an overview of previous work and existing knowledge on SIP abuse. Section III briefly introduces the SIP protocol. Section IV outlines our method used for data collection and highlights the methodology used to create tool-specific fingerprints. Section V analyzes a set of 21 tools found in-the-wild. In Section VI we identify scanning campaigns conducted using these tools and show how actors are distributing their operations. Section VII summarizes and concludes our work.

II. Related work

In 1999, Handley et al. introduced the Session Initiation Protocol (SIP) [3], used for initiating and maintaining real-time session for messaging applications including voice and video messaging. Over the years, the protocol has seen many scrutinous changes adding new functionalities and security improvements [4], [5], [6], [7], [8].

As SIP is a widely adopted protocol, a large body of research work exists identifying and solving security vulnerabilities of the protocol. Attacks on SIP are discovered impairing the availability of servers, leaking sensitive information [9], injecting data in SQL databases, or even using SIP servers as an amplification service to perform DDoS attacks [10]. To aid in finding SIP vulnerabilities, many researchers have created automated tools which automatically scan and identify problems in the software [11], [12], [13], [14]. Along with automated vulnerability scanning tools, other tools were introduced to scan the Internet identifying hosts running SIP such as NMap [15] and SIPVicious [16]. This so-called port scanning allows attackers to identify and consequently exploit hosts running SIP. To examine the threat-landscape of SIP, Nassar et al. [17] created a honeypot system capable of capturing attack traffic in the wild. In similar work, Hoffstadt et al. [1] deploy a honeypot system in the wild for almost one full year, capturing over 47.5 million SIP messages. Aziz et al. [18] record in a similar study 857 different attackers in their honeypots over 1.5 months. While these works report on exploitation behavior in the wild, the authors do not go into differences in scanning tools and the evolution of SIP scanning over multiple years.

In other work capturing SIP scanning in-the-wild, Dainotti et al. [19] analyze a large horizontal network scan targeting the SIP protocol, identifying the SIP payloads used in the attack and noting that the structure of the payload is similar to that of the SIPVicious scanning tool, which the malware authors most likely modified. In a follow-up paper, Raftopoulos et al. [20] identify exploits attempts following the large SIP scan in 2011 and identify a surge in IP addresses performing exploitations of SIP. While previous works have identified in-the-wild attack traffic, only one focused on detecting the Tools, Techniques, and Procedures (TTPs) used by attackers to perform these attacks [21]. While the authors identify tools used to scan and exploit their SIP servers and note that adversaries change SIP fields to circumvent IDS detection, they did not identify large scanning and exploitation campaigns or fingerprints of specific tools. In this paper, we aim to address the gap by fingerprinting the tools used to perform SIP scanning and analyze 6 years of in-the-wild scanning traffic to identify actor evolution.

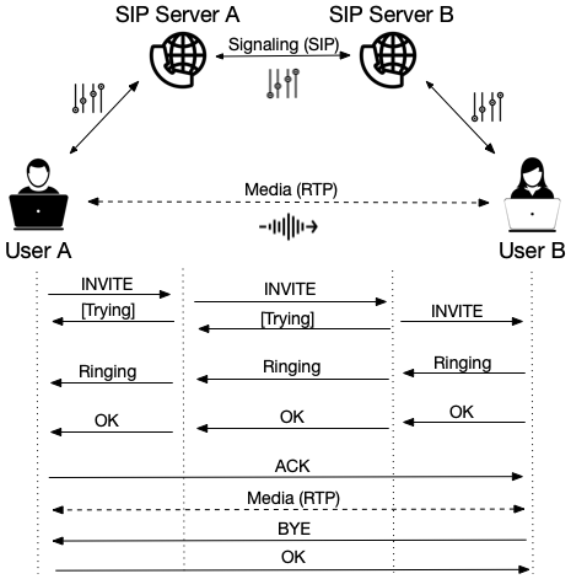

Fig. 1: SIP Signaling and RTP media flow during a VoIP call.

III. The Session Initialization Protocol

Before we go into the discussion of our data collection and results, we will first briefly introduce the necessary flows in Voice-over-IP and specifically the Session Initialization Protocol as background for the messages we observe later and their interpretation in the protocol.

As discussed in the introduction, Internet telephony uses two components: first, the signaling protocol SIP communicates status messages, setups and terminates calls, or exchanges configuration information, and second, a media transfer protocol such as RTP carries the actual voice and video payload between the two communicating endpoints. While for efficiency and latency reasons media is transmitted directly between caller and callee, this of course requires knowledge of the location and network address of the remote party. When users are mobile or we are making a connection for the first time, we would clearly not know yet where to find the other party; hence

signaling messages are exchanged via proxy servers. VoIP user identities come in the form of *sip:username@domain*, which would tell the caller which server to contact to establish a connection with the desired endpoint. As shown in figure 1, a SIP endpoint requests its own SIP proxy to make a call, which would contact the server responsible for the destination. As the called user would keep alive a connection with its own SIP proxy, this server would know how to reach its user and pass along the request for a connection to the client. If the connection is accepted and passed back via the proxies, both parties would have all information necessary to start exchanging data directly.

The Session Initialization Protocol has a message structure comparable to HTTP. Requests are made from a client to a server, which returns a response message in the same connection. The protocol messages are text-based, the first line of the text contains the request or response (like an HTTP GET or POST), followed by headers and a message body. As SIP monitors for timeouts and has the functionality for retransmits built into the protocol, SIP messages are preferably sent over UDP instead of TCP to reduce connection latency and overhead of the transport layer. There are six general types of requests in the protocol: *INVITE* messages request the establishment of a SIP session - thus a VoIP call -, an *ACK* confirms the receipt, a *BYE* terminates an existing connection, and *CANCEL* stops a transaction that is currently ongoing. An *OPTIONS* message asks the other side for its capabilities with starting a session, and a *REGISTER* performs a sign-in of an endpoint with the proxy server of its SIP registrar. This means that to get an insight into SIP abuse and brute-forcing in the wild, we need to open and listen to UDP port 5060 on publicly exposed IP addresses. An abuse will either start by a registration attempt through *REGISTER* or the request for a call in an *INVITE*, which we can harvest when monitoring a large number of IP addresses for incoming SIP traffic.

IV. Dataset Collection and Processing

The data source used in this study is a network telescope of two partially populated /16 networks. All incoming data directed at the unused IP addresses is logged, which gives insights into all unsolicited traffic such as port scans probing for available systems as well as backscatter traffic originating from current attacks on the Internet. As a network telescope consists solely of unused IP space, this dataset is void of any user traffic and therefore provides a clear overview into Internet-wide scanning for SIP services. This section provides an overview of the dataset collected from the network telescope and the method used to process SIP packets.

A. Dataset

As the network telescope consists of two partially populated /16 networks, it receives a large amount of data that needs to be processed. On average in 2020, the telescope recorded 1.3 Tb of raw network traffic per month. To manage the size of the data we aim our analysis at the last 6 months of 2020, providing a recent analysis of SIP scanning attempts in the wild. In addition, to allow for a longitudinal analysis of the TTPs used by attackers and their evolution, we include the first two months of each year from 2016 until 2020.

We select all packets sent to the SIP service running on the IANA default UDP port 5060 and discard 9.6 million packets that do not contain a valid SIP message. During the measurement period, we observe 822 million packets containing a valid SIP header originating from 5,691 unique sources, sending on average 144,439 packets.

B. Data processing

While SIP runs over UDP and is therefore vulnerable to adversaries spoofing their source IP address, this would prevent the adversaries from receiving the response packet sent by a server. As scanners need the answer to the probe packet to establish whether a port is open, source IP addresses observed in the network telescope will be controlled by the scanning actor. We can thus group packets originating from an IP address into a scanning "flow". Flows are an aggregation of the SIP packets originating from a single source IP address with an inactivity time-out of 20 minutes. After these 20 minutes of not observing a packet in the telescope, the flow is closed and the next packet coming in from this IP address will create a new flow. These flows aim to identify artifacts inside the packets sent by a specific scanning tool.

To ensure we do not lose essential data while aggregating the packets, we leverage the structure of a SIP packet to devise a single-pass algorithm that can create a fingerprint for a scanning tool. The SIP and SDP header contain text-based variables in the form: *key:value*. To combine these text-based values, we compare the values by key, and only keep the overlapping characters in the values corresponding to a key in two packets. For example if we compare the following two "from" fields: *<sip:nm@nm>;tag=1526* and *<sip:nm@nm>;tag=8711*, we keep the part that is an exact match: *<sip:nm@nm>;tag=* and replace the non-matching part with a non-Ascii character to ensure the location of all parts of the string remains the same. In this paper we use a dash to denote a non-matching character, the compared strings will thus return: *<sip:nm@nm>;tag=* - - - -. When all packets are compared and aggregated, only the part that is constant over all packets will be saved. By aggregating all keys of both the SIP and SDP header, we obtain a fingerprint containing parts of fields that are randomized and parts of fields that are constant in every packet. The characteristics captured in these fingerprints allow us to distill information about specific tools, as every implementation of a SIP scanner will generate these packets differently and therefore randomize header fields differently. For example, the standard NMap [15] SIP fingerprint is:

```
accept:application/sdp
call-id:50000
contact:<sip:nm@nm>
content-length:0
cseq:42 OPTIONS
from:<sip:nm@nm>;tag=root
max-forwards:70
```

```
to:<sip:nm2@nm2>
via:SIP/2.0/UDP nm;branch=foo;rport
```

As seen from the fingerprint NMap always sends the same headers in a SIP packets, as the aggragation of all packets does not show any randomizations in the header fields. A tool that randomizes more fields is SIPVicious [16], with the fingerprint:

```
accept:application/sdp
call-id:-------------------------
contact:sip:100@SourceIP:SourcePort
content-length:0
cseq:1 OPTIONS
from:sipvicious<sip:100@1.1.1.1>;
          tag=38336234--------313363340...
max-forwards:70
to:sipvicious<sip:100@1.1.1.1>
user-agent:friendly-scanner
via:SIP/2.0/UDP SourceIP:SourcePort;
           branch=z9hG4bK-----------------
```

In this fingerprint, the *call-id* field is always randomized together with the branch number and tag in the *from* field, but the *user-agent* is set to *friendly-scanner* in every packet. Additionally, we replace IP addresses and port numbers used inside the header with the strings "SourceIP", "DestIP", "SourcePort" and "DestPort" as they are commonly used in SIP headers and we will otherwise discard this characteristic if the source port is for example not constant between packets. The resulting fingerprints allow for the comparison of tools and are robust to small changes to a tool, as most other fields will still match. We have tested this method against four base and modified versions of NMap and SIPVicious, and can uniquely identify all the different implementations.

V. SIP SCANNING TTPS

After applying our algorithm to the data, we obtain 187,327 separate flows, with an average of 4,393 packets per flow. Using these flows, this section will describe and analyze the behavior of SIP scanners. Using the implementation characteristics mined from the packets as described in the previous section, this section will list and analyze broadly used, as well as custom-made tools used in SIP scanning.

A. SIP scanning tools

As shown in section IV, tools generate packets with certain combinations of SIP header fields. A tool such as SIPVicious for example includes an optional header field *user-agent* and sets this to *friendly-scanner*. To detect scanning packets from this tool, a match can be made with this field and all of the probes will be filtered out. With an abundance of tutorials showing sysadmins how to filter probes based on this field, an actor might change this to prevent the probes from being dropped. To circumvent this detection, some actors might make minor changes to existing tools to alter the fingerprint to circumvent various scanning detection mechanisms rather than create a new tool. In this section, we will differentiate between tools derived from known open-source tools and custom-made tools from scratch by comparing fingerprints to those of the open-source tools NMap and SIPvicious.

Table I shows the 21 most used tools we have been able to identify in our dataset, the year they first appeared, the number of packets reaching the telescope, a selection of SIP headers, their scanning strategy, and whether this tool is known when searching for it on the Internet. While SIPVicious is easily detected and the authors even provide a program that can be used to crash unsolicited scanning campaigns targeting your network [22], the standard version of SIPVicious is by far the most popular tool used to scan for open SIP servers. The alterations of the SIPVicious toolkit mostly change the *user-agent* field of the tool to either a name from a list of user-agents or a single genuine-looking SIP user-agent such as *Cisco* or *PBX*. While changing the *user-agent* will allow an attacker to circumvent some detection methods, tool 2 neglects to also change the other obvious mentions of SIPVicious in the *from* and *to* fields, where *sipvicious* is written into. Tool 3 and 4 are more thorough, as both change the fields to respectively a randomly chosen (but valid) agent name or the string "*AmooT*".

While tools based on SIPVicious change their fields and add more randomization, tools based on NMap keep the normal structure where no SIP header field is randomized and instead only change the values present in the fields. The sophistication of the change in the tool seems to be dependent on the sophistication of the tool itself, as randomization is harder to implement in a tool statically sending the same packet than in a tool already including some form of randomization.

The best way to avoid detection of artifacts created by commonly used tools is to create a tool from scratch, or significantly change the tool such that no original fields are included in the packet anymore. 13 of the 21 tools discovered are not traceable back to a base version of NMap or SIPVicious and highly vary in implementation, with more customization in SIP header fields but also changes in scanning strategy. Most tools aim to spoof the *user-agent* field to something believable, but some tools such as tool 9 selects a random string for each packet. While this does not generate believable user-agents, it does circumvent any blacklisting approach as the string will be different on each probing attempt.

Most custom-made tools change header fields to expected values for a SIP packet and pick believable user-agents that change every packet. Some tools however contain implementation details that make it trivial to spot that packets do not originate from a normal SIP client. NMap based tool 8 changed the SIP packet header details by including the misspelled key *content-lnegth* (sic!), which a normal SIP parser would not be able to match with an existing option in the protocol. Tool 14 even implements the SIP protocol using a dummy example found in many online tutorials, with names as *Alice* and *Carol*.

B. Scanning artifacts

Scanning tools do not only create artifacts in SIP header fields but also fields in the IP and UDP header such as the destination IP address are set by the scanning tool. If the packets are injected on the network interface without using

	ID	Variant	First seen	Packets	Selection of SIP header fields used to identify the tool	Scan strategy	Known
Sipvicious	1	Base	2016	694 million	contact:sip:100@SourceIP:SourcePort from / to:sipvicious<sip:100@1.1.1.1> user-agent:friendly-scanner	Sequential	✓
	2	Changed user	2016	26 million	user-agent:[Random agent] from / to:sipvicious<sip:100@1.1.1.1>	Sequential	
	3	Changed fields	2016	19 million	user-agent:[Random agent] from / to:[Random name]<sip:100@1.1.1.1>	Sequential	
	4	AmooT	2018	26 million	user-agent:[Random agent] from / to:AmooT<sip:100@1.1.1.1>	Sequential	
NMap	5	Base	2016	3.6 million	call-id:50000 contact:<sip:nm@nm> from:<sip:nm@nm>;tag=root to:<sip:nm2@nm2>	Random sequential	✓
	6	Test	2020	238,623	contact:<sip:test1@test1> from:<sip:test1@test1>;tag=root to:<sip:test2@test2>	Random sequential	
	7	sip:a@a	2020	7.5 million	call-id:97 contact:<sip:test1@test1> from:<sip:a@a>;tag=1 to:<sip:b@b>	Random sequential	
	8	sip:ag@ag	2019	40,883	call-id:5000 contact:<sip:ag@ag> from:<sip:ag@ag>;tag=root content-lnegth (sic!):0 to:<sip:ag2@ag2>	Random sequential	
Other	9	100rel	2016	501,487	contact / to:<sip:[8 characters]@SourceIP:SourcePort> from:<[8 characters]<sip:[8 characters]@DestIP:DestPort>;tag=hetfgeeb max-forwards:30 supported:100rel	Random	
	10	Random user	2016	11 million	call-id:[10 characters]@SourceIP contact / from / to:[8 characters]@SourceIP:SourcePort user-agent:[8 characters]	Random	
	11	a123	2020	9 million	call-id:a123-bc2547-a159 user-agent:Avaya one-X Deskphone	Random	
	12	pplsip	2018	8 million	call-id:a123-bc2547-a159 contact:<sip:me@SourceIP:SourcePort> from:me<sip:me@DestIP> to:me<sip:me@DestIP> user-agent:pplsip	Sequential	✓
	13	StarTrinitySecurity	2019	360,363	call-id:startrinityvoipsecuritytestsuite contact:sip:s3@SourceIP:SourcePort from:sip:startrinityFriendlyScanner@SourceIP to:sip:penetrationTest@DestIP user-agent:StarTrinityFriendlyScanner	Random	
	14	a84b4c	2019	2 million	call-id:a84b4c76e66710 contact:<sip:alice@pc33.atlanta.com> from:Alice<sip:alice@atlanta.com> to:<sip:carol@chicago.com> via:pc33.atlanta.com	Random sequential	
	15	sip:b@localIP	2016	2,512	call-id:[32 characters]@127.0.0.1 contact / from:<sip:b@127.0.0.1> to:<sip:DestIP> date:[Current date] user-agent:Asterisk PBX	Random	
	16	sip:99999	2020	555,146	contact:<sip:99999@SourceIP:SourcePort> content-length:209 from:<sip:99999@DestIP:DestPort> to:<sip:[phonenumber]@DestIP>	Sequential	
	17	x-nat	2019	7,941	call-id:[9 numbers]@[5 letters] from / to:[5 numbers]<sip:[5 numbers]@SourceIP:SourcePort> x-nat:nothing x-serialnumber:[10 characters] server:o2-IAD_6741-2.6.3.32	Random	
	18	adoom	2018	2 million	contact:<sip:adoom@SourceIP:SourcePort> from:<sip:adoom@DestIP>[random string] to:<sip:adoom@DestIP> user-agent:Cisco-SIPGateway/IOS-12.x	Sequential	
	19	VOIP	2019	362,437	contact:sip:[phonenumber]@0.0.0.0:SourcePort from / to:[phonenumber]<sip:[phonenumber]@DestIP> user-agent:VOIP	Sequential	
	20	01146441408560	2019	3 million	contact:sip:DestIP:SourcePort from:DestIP<sip:01146441408560@DestIP> to:01146441408560<sip:01146441408560@DestIP> user-agent:[Random agent]	Sequential	
	21	Nessus	2017	5,778	contact:<sip:DestIP> from / to:Nessus<sip:DestIP:DestPort>	Sequential	✓

TABLE I: Top scanning tools used and a selection of their most prominent SIP header fields. Fields changing each packet are enclosed in square brackets. Header fields with exactly the same values are grouped together.

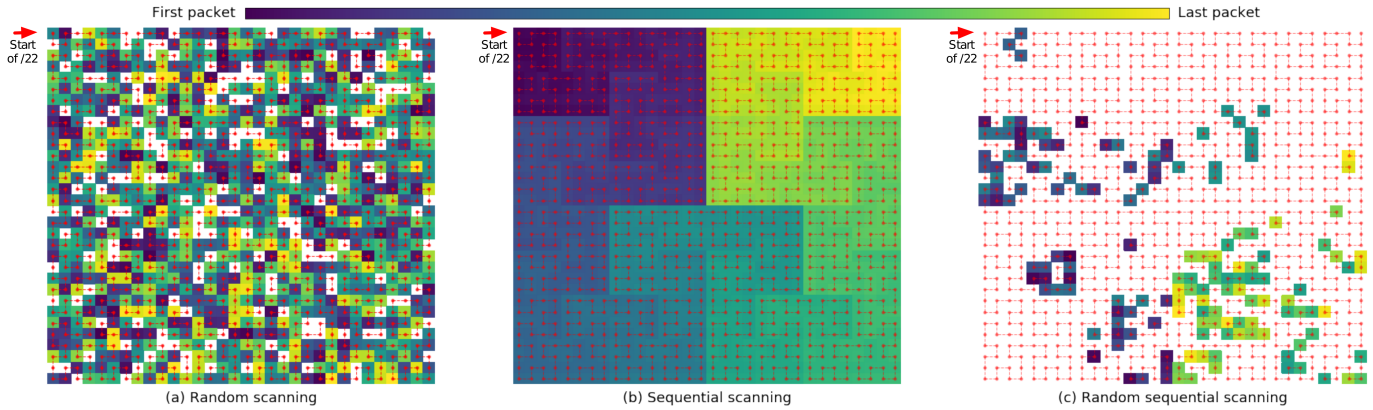

Fig. 2: Destination IP scanning patterns observed from different scanning tools in-the-wild mapped on a Hilbert curve containing a /22 netblock.

an Operating System socket, which is mostly used for high-performance scanning [23], fields such as the Time-To-Live or the IP identification number will also be set by the tool itself.

A tool can select the next target to scan in various ways, for example by randomizing the destination IP address or increasing it by 1 every packet. To visualize these differences, we map our /16 ranges onto a Hilbert Curve, which is a 2D projection of a list that retains the distances between most points, where points close to each other in the list are also close to each other in the projection. We assign all points of the projection a color depending on when the particular IP address was scanned inside a flow. Figure 2 shows this projection for three different strategies seen in scanning tools. For visibility, we only include a /22 netblock in the figure.

Figure 2(a) shows a random scanning pattern, implemented by some custom-made tools which randomize the destination IP address over the entire IPv4 space for every packet. By doing so, a scanner will hit a specific netblock with lesser intensity, as the probability of picking an IP address in this netblock is $\frac{netblock-size}{2^{32}}$. Figure 2(b) shows a scanning behavior where the next scanned IP address is sequential, which is implemented in SIPVicious and is thus also present in all tools based on this source code. This functionality can be changed in SIPVicious by providing a *–random* flag when running the program, we have however not been able to identify any SIPVicious based activity that has used this functionality instead of the sequential scan. While a sequential scan is simple to implement, this method will scan all IP addresses in a netblock in a small time frame, making it easier for IDS systems to identify a scan is going on by matching the number of packets originating from a single source IP address in a small time window. Figure 2(c) shows the SIP scanning behavior of NMap and similar tools, where the IP address is randomized but generated for small netblocks in sequence, leading to all IP addresses in a netblock being scanned at once before moving on to another part of a netblock.

Another field a scanner can customize is the source port from which the packet is sent. For NMap based scanners these source ports are randomized on every packet sent, without customization from users. In SIPVicious however, a user can choose a port on startup of the scan with a default of 5060 where the program will bind to in the operating system. When the program is unable to bind to a specific port for any reason, it will automatically move on to the next port until the program can bind to the port. Here there is again little customization as 94% of all flows are initialized with the default source port. 3% of the flows are initialized with port 5061, indicating that another process is already bound to the standard SIP port.

Message	**Response**
INVITE	100 Trying
REGISTER	401 Unauthorized
BYE / CANCEL	481 Call / transaction doesn't exist
OPTIONS	200 OK (Options in headers)

TABLE II: SIP message types used for scanning and the responses from a server.

C. SIP messages

As described in section III, there are six general types of SIP requests: *INVITE*, *ACK*, *BYE*, *CANCEL*, *OPTIONS* and *REGISTER*. Except for the *ACK* message, all of these messages can be used for the detection of SIP servers, as a SIP server will respond. Table II shows responses of a plain SIP server to the message types. As every message type elicits a different response, the messages provide an adversary with different information after a server is identified. In our measurements, we have observed the *INVITE*, *CANCEL*, *OPTIONS* and *REGISTER* commands being used to scan for open services. Most scans use the *OPTIONS* message type, which is the default scan type of most tools including NMap and SIPVicious, and notifies the scanner about server functionality.

While SIPVicious and NMap based scanners mainly use the *OPTIONS* type in the message, custom-made tools instead prefer to use *REGISTER* messages, with 6 out of the 13 custom tools solely using this message type. By using this message type, the scanner will immediately find whether an additional login is needed on the server, or that it can be used to perform

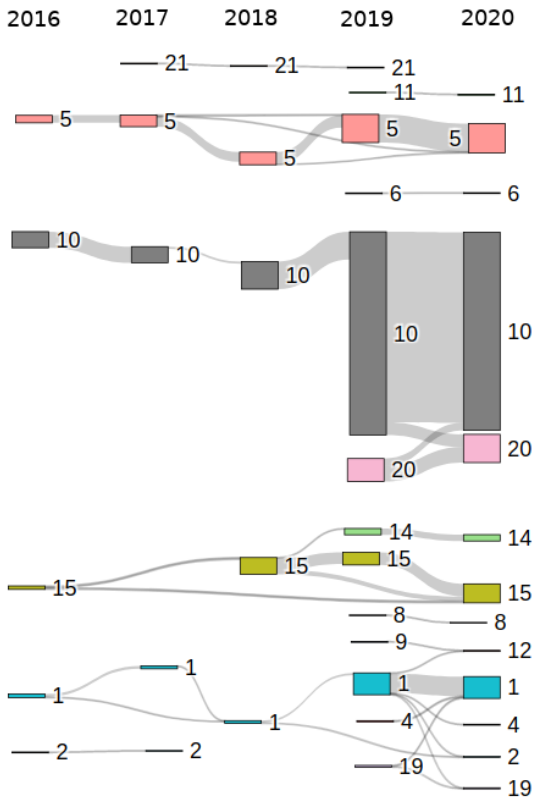

Fig. 3: Sankey diagram of the yearly evolution of tools. Boxes show the relative number of IP addresses using a tool and the flows between boxes show the overlap between years.

for example call fraud without having the server credentials. *CANCEL* is the least often used with only tool 17 sending this message type. While a packet containing *CANCEL* could be used to forcefully stop a SIP connection, it is very unlikely for this to happen in a scan due to the entropy of call-ids. *INVITE* messages are only used by scanners 11, 12, and 20 and do not only obtain a reply from the server to identify new SIP servers, but they also instruct the server to request the establishment of a SIP session when a phone number is provided in the packet, which only occurs in 3.5 million packets. In total, 234 unique numbers are included located all over the world. We do not find any indication why these specific numbers are called, except for two numbers that are included in over 100K packets each and are reported to be scammers by a Norwegian phone number reputation database.

D. Tool evolution

When scanning the Internet for a long time, actors might opt to redesign their tools to make them faster or change the payload to prevent their probes from being fingerprinted and subsequently dropped. To understand how SIP scanning actors have evolved, we identify the 21 tools found using the fingerprints in-network telescope data of the first two months of every year from 2016 to 2020 and compare the tools used

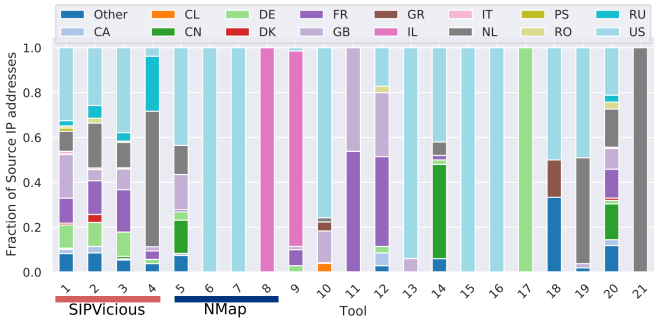

Fig. 4: Geographical distribution of source IP addresses running specific tools.

by source IP addresses over the years. Figure 3 shows the evolution of tools used by source IP addresses, where only the IP addresses that are observed to be scanning for more than one year are included in our data. When scanning for periods spanning multiple years, it would be expected that they update their tools over time, yet we see surprisingly little evolution in tooling. Most actors never change their tool even when newer tools are available that are better suited for circumventing detection systems by for example randomizing destination IP addresses. Even when IP addresses are observed years apart, such as an IP address that was only online in 2016 and 2020 running the *sip.b.local* tool, the same tool is used when they come online again.

E. Geographical distribution of tools

While scans originate from 75 countries, 65% of scanning traffic originates from only 5 countries: Germany, the United States, the United Kingdom, France, and the Netherlands. Interestingly, this does not follow usual IP allocation; China has a large address space but only 1.4% of hosts scanning SIP originates from China. Germany on the other hand has a much smaller address space while 30% of all SIP scanning hosts originate from this country. Overall scanning traffic mostly originates from countries such as China or the US [24], which is not the case for SIP scanning traffic.

Figure 4 shows the geographical distribution of sources using one of the tools we can fingerprint. The figure shows that tools based on SIPVicious are highly distributed, with countries with large IP space such as the US holding a significant part of the scanning activity. Tool #4 is also highly distributed in countries, but a disproportionate amount of sources is located in the Netherlands and Russia. We can trace back the scanning activity of this tool to datacenters and bullet-proof hosting providers. NMap-based tools show less geographical distribution, with only the base tool and not its derivations being used from multiple countries. Unlike SIPVicious, NMap is also used from Chinese sources. The only other two tools used from China are tool #14 which uses a standard tutorial packet and tool #20 which uses *INVITE* messages. As tool #20 is mostly distributed according to geographical IP allocation and only sends *INVITE* messages we suspect this tool to be used to place phone calls rather than

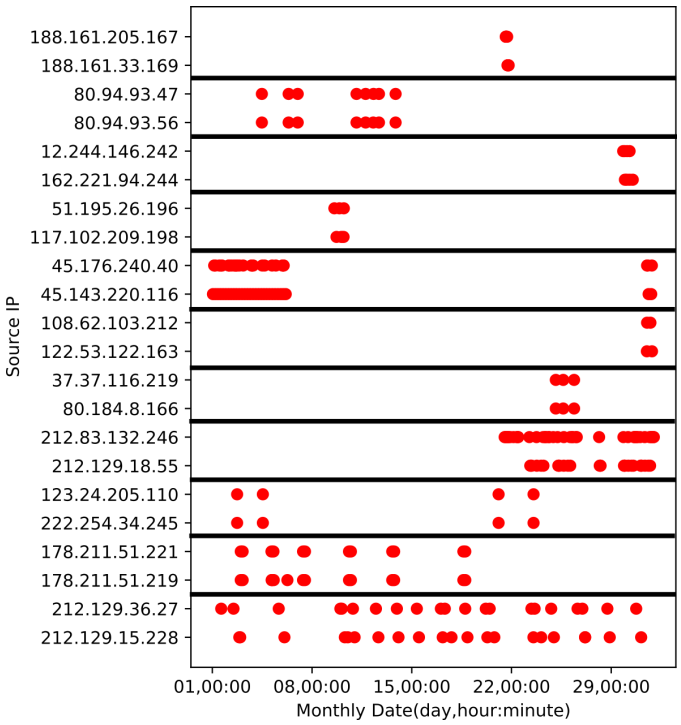

Fig. 5: Examples of tuples of scanning IP addresses.

discover SIP services, and therefore randomizing the Source IP address of each packet preventing the answer to these requests being observed by the actor. Custom tools are as expected much less distributed than common tools, as common tools are used by various actors around the world, and custom tools are most likely used by a single actor or small groups of actors.

VI. Campaign analysis

While the fingerprints identify a tool used by a source IP address, it does not allow for direct identification of scanning campaigns as the same tool might be used by different actors. In this section, we will identify behavioral traits allowing for the identification of scanning campaigns targeting SIP and analyze the behavior of these scanning campaigns.

A. Behavioral clustering

When scanning the Internet, an actor will either send all packets from one single source IP address or farm out the operation over multiple controlled hosts. As it is impractical to create a new tool for every collaborating host, the actor will most likely use the same tools on all machines. To verify this claim we first create clusters of source IP addresses by grouping them based on their activity and verify that campaigns we find are conducting scans with a single tool.

A scanning campaign consists of multiple distributed hosts scanning the Internet at the same time. Figure 5 shows timelines of similar scanners, creating a significantly unique behavioral fingerprint used to cluster similar scanners. We cluster source IPs in three steps: First, we create a binary vector spanning our entire measurement period for every source IP address where a '1' is added on every day the source IP has been observed and a '0' when it is not observed. Second, we perform HDBScan clustering using pairwise Euclidean distances on the resulting vectors and select the clusters with 5 or more hosts. Third, we remove all clusters of IP addresses that are always active as they do not provide us with a distinct behavioral fingerprint but will rather cluster all campaigns scanning every day into one big campaign.

After clustering the 5,691 source IP addresses we obtain 250 campaigns containing in total 2,787 (49%) IP addresses, which are all scanning the Internet using only one tool. Figure 6 visualizes these campaigns in a bubble chart where every bubble corresponds to a campaign, colored by the specific tool that was used to perform the scan. We find that common tools such as SIPVicious are used to scan the Internet at high speeds (mean of 520 packets per second in our telescope), and only limited actors are actively throttling their scanning speed to avoid detection by IDS systems. While only one campaign using SIPVicious is sending at a rate where we observe less than 5 packets per second, we observe highly distributed campaigns from custom tools that are actively scanning at very slow rates, with 4 campaigns sending even less than 200 packets per second if we extrapolate over the entire IPv4 space. We would expect that actors adapt the number of packets sent based on the number of hosts used to perform the scan, and thus send half as many packets per host when scanning with 2 IP addresses as opposed to one. To test this we correlate the size of the cluster with the number of packets sent per source IP address and find a much weaker relation than we expect with $R = -0.134$ $(p < 0.05)$ as many campaigns scan all IP addresses from all of their hosts. For scanning speed, we do not find a significant relationship, but there is an indication of a small negative effect of the amount of source IP addresses used in a campaign and the average scanning speed ($R = -0.09, p = 0.13$), indicating that there is a slight reduction in scanning speed when distributing scanning activities, but most distributed scanners are farming out their operation to obtain a significant speedup of the campaign. The largest campaign we identify is using tool #10 and is highly coordinated, evenly distributing the operation over 283 IP addresses and limiting scanning speed.

B. Scanning activity

While scanning the Internet once for open SIP servers might be enough for adversaries that aim to perform a one-time exploit, some actors will periodically update the list of open servers by performing regular scans. Campaigns using tools #10 and #17 are for example scanning the Internet every month, while tool #15 is used very sporadically and no clear pattern emerges in its scanning behavior. From all IP addresses scanning SIP, 2,634 (46%) only scan the Internet once and never come back. From the 54% of IP addresses generating multiple flows, only 642 (21%) are located in Germany, where the bulk of SIP scanning IP addresses are located. Instead, most recurring scanning activity originates from the US, where 1,345 (44%) of the IP addresses generating multiple flows are located. Additionally, we find that campaigns using

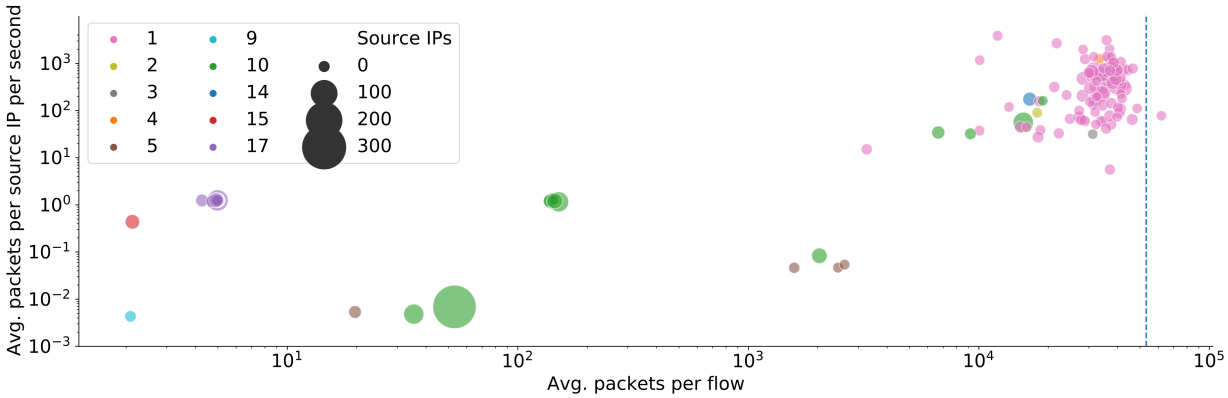

Fig. 6: Bubble plot of scanning campaigns identified in the data colored by tool, with the observed amount of packets per IP address and the average speed observed in the telescope. A blue dotted line shows the addresses in the network telescope.

custom tools are more likely to perform periodic scanning with 76% of campaigns being periodic whereas only 29% of SIPVicious-based scanning campaigns show recurrence. While over time Internet speeds increase, with only in 2017 already a global speedup of 30% [25], we do not see a significant increase in scanning speed over the years ($p = 0.79$). Instead, average scanning speeds remain constant as the variation between clusters becomes larger, where increases in scan rate are countered by campaigns that are actively throttling their operation. The number of packets sent per source IP address in a campaign however sharply drops over the years ($R = -0.41,\ p < 0.001$), showing that the distribution of scanning campaigns becomes much more frequent.

VII. Conclusion

In this work, we analyze tools used for SIP scanning from 2016 until 2020, finding that most actors perform large single-source scans using standard tools. We analyze who is scanning and provide a method to identify which tools are used to perform a scan, uncovering 21 different tools used by various actors. We find evidence of distributed and persistent scanning campaigns, where actors scan the Internet periodically and actively reduce the number of probes sent per second by their measurement infrastructure. Over time, scanning campaigns become increasingly more distributed, making these scans harder to detect for defensive infrastructures.

References

[1] D. Hoffstadt, A. Marold, and E. P. Rathgeb, "Analysis of sip-based threats using a voip honeynet system," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 541–548, IEEE, 2012.

[2] N. Miramirkhani, O. Starov, and N. Nikiforakis, "Dial one for scam: A large-scale analysis of technical support scams," *arXiv preprint arXiv:1607.06891*, 2016.

[3] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "Rfc 2543," *SIP: Session Initiation Protocol*, 1999.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "Rfc3261: Sip: session initiation protocol," 2002.

[5] A. Roach, "Rfc3265: Session initiation protocol (sip)-specific event notification," 2002.

[6] R. Sparks, S. Lawrence, A. Hawrylyshen, and B. Campen, "Rfc5393: Addressing an amplification vulnerability in session initiation protocol (sip) forking proxies," 2008.

[7] A. Roach, "Rfc6665: Sip-specific event notification," 2012.

[8] S. Donovan *et al.*, "The sip info method," 2000.

[9] D. Geneiatakis, T. Dagiuklas, G. Kambourakis, C. Lambrinoudakis, S. Gritzalis, K. S. Ehlert, and D. Sisalem, "Survey of security vulnerabilities in session initiation protocol," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 3, pp. 68–81, 2006.

[10] U. U. Rehman and A. G. Abbasi, "Security analysis of voip architecture for identifying sip vulnerabilities," in *2014 International Conference on Emerging Technologies (ICET)*, pp. 87–93, IEEE, 2014.

[11] H. J. Abdelnur, R. State, and O. Festor, "Kif: a stateful sip fuzzer," in *Proceedings of the 1st international conference on Principles, systems and applications of IP telecommunications*, pp. 47–56, 2007.

[12] H. Abdelnur, O. Festor, *et al.*, "Advanced fuzzing in the voip space," *Journal in Computer Virology*, vol. 6, no. 1, pp. 57–64, 2010.

[13] T. Alrahem, A. Chen, N. DiGiuseppe, J. Gee, S.-P. Hsiao, S. Mattox, T. Park, I. Harris, *et al.*, "Interstate: A stateful protocol fuzzer for sip," *Defcon*, vol. 15, pp. 1–5, 2007.

[14] S. Taber, C. Schanes, C. Hlauschek, F. Fankhauser, and T. Grechenig, "Automated security test approach for sip-based voip softphones," in *2010 Second International Conference on Advances in System Testing and Validation Lifecycle*, pp. 114–119, IEEE, 2010.

[15] G. F. Lyon, *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, 2009.

[16] S. Gauci, "Sipvicious." URL: https://github.com/EnableSecurity/sipvicious.

[17] M. Nassar, R. State, and O. Festor, "Voip honeypot architecture," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 109–118, IEEE, 2007.

[18] A. Aziz, D. Hoffstadt, E. Rathgeb, and T. Dreibholz, "A distributed infrastructure to analyse sip attacks in the internet," in *2014 IFIP Networking Conference*, pp. 1–9, IEEE, 2014.

[19] A. Dainotti, A. King, K. Claffy, F. Papale, and A. Pescapé, "Analysis of a /0 stealth scan from a botnet," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 341–354, 2014.

[20] E. Raftopoulos, E. Glatz, X. Dimitropoulos, and A. Dainotti, "How dangerous is internet scanning?," in *International Workshop on Traffic Monitoring and Analysis*, pp. 158–172, Springer, 2015.

[21] J. M. Ceron, K. Steding-Jessen, and C. Hoepers, "Anatomy of sip attacks," *; login:: the magazine of USENIX & SAGE*, vol. 37, no. 6, pp. 25–32, 2012.

[22] S. Gauci, "Svcrash." URL: https://github.com/topics/svcrash.

[23] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman, "Zippier zmap: internet-wide scanning at 10 gbps," in *8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14)*, 2014.

[24] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, pp. 65–78, 2014.

[25] Speedtest, "Internet speed increase in 2017." URL: https://www.speedtest.net/insights/blog/global-speed-2017.