Hyhoneydv6: A hybrid Honeypot Architecture for IPv6 Networks

Sven Schindler[1], Bettina Schnor[1], Thomas Scheffler[2]
[1]*Department of Computer Science University of Potsdam, Germany*
[2]*Department of Electrical Engineering Beuth Hochschule, Berlin, Germany*

Abstract

This paper presents a new hybrid honeypot archi-tecture which focuses on the coverage of large IPv6 address spaces. Results from a 15-months darknet experiment verify that attackers and researchers utilise various approaches to scan wide and unforeseeable IPv6 address ranges which cannot be managed with current honeypot solutions. The huge IPv6 address space not only makes it hard for attackers to find target hosts, it also makes it difficult for a honeypot to get found by an attacker. We solve this challenge through the use of dynamically configured high-interaction honeypots that can cover large chunks of the IPv6 address space. A new proxy mechanism is used to transparently handover and forward traffic from low-to high-interaction honeypots on demand to provide the best possible service granularity. Measurements with our prototype implementation show that the proposed approach performs well on off-the-shelf hardware and has low maintenance costs.

1. Introduction

In May 2015, the global IPv6 usage of all Google users reached a peak of more than 6 percent[1], which is a growth of more than 100 percent over a single year. Individual countries, such as Belgium, observe up to 33 percent of IPv6 traffic. These numbers motivate our growing interest to determine the current threat level in IPv6 networks.

Honeypots are useful security tools to monitor malicious activities, judge the network threat level and gain insight in the application of known and novel attack methods. The two major IPv6-capable low-interaction honeypot projects are Dionaea[2] and Honeydv6[3][16]. Dionaea focuses on the emulation of well-known services, such as SMB or SIP. Honeydv6 extends the well-known honeypot Honeyd [14] and provides a framework to simulate entire IPv6 networks.

IPv4 networks do not require any special provisions to make sure that a honeypot will be found by an attacker. Tools like ZMap and Masscan can be used to scan the entire IPv4 Internet within minutes [7]. This stands in stark contrast to the deployment of honeypots in IPv6 networks, where the huge address space makes brute force network scans impossible. The following section presents the results of a 15-months IPv6 darknet experiment which shows that attackers apply various scan approaches to explore IPv6 networks. We will show that these scan approaches require new honeypot design concepts that are not provided by current honeypot implementations. In the following, we introduce a new hybrid honeypot architecture called *Hyhoneydv6* that is designed to cover huge IPv6 address spaces.

[1]http://www.google.com/intl/en/ipv6/statistics.html
[2]http://dionaea.carnivore.it/
[3]https://redmine.cs.uni-potsdam.de/projects/honeydv6/

2. Results of an Darknet Experiment

We conducted a 15-months darknet experiment using a /34 network to find out what kind of scan approaches attackers use to explore IPv6 networks.

2.1. Experimental Setup

We used the packet analyser tcpdump[4] to capture the entire darknet traffic in our /34 darknet. We analysed the received network scans and visualized the targeted address spaces by converting each probed address into a 128-bit integer value that we could depict on a *gnuplot*[5] diagram. It is important to note that the applied gnuplot version 4.6.6 does not support large 128-bit values. For this reason, we downsized the generated values without destroying the proportions.

2.2. Scan Patterns

Table I shows the total number of received packets grouped by protocol. ICMPv6 represents the majority of the received traffic. With over 31,600 received packets, TCP amounts to about 12 percent of the captured traffic. The amount of received UDP traffic is comparatively low. With 226 UDP packets, the protocol takes up less than 0.1 percent of the total traffic.

Table I
PROTOCOL DISTRIBUTION AND TOTAL AMOUNT OF CAPTURED PACKETS DURING THE EXPERIMENT

Total	255,840	
ICMPv6	224,010	87.56%
TCP	31,604	12.35%
UDP	226	0.09%

The ICMPv6 traffic was dominated by Echo Request-based network scans and the TCP traffic contained mostly TCP SYN

[4]www.tcpdump.org
[5]http://www.gnuplot.info

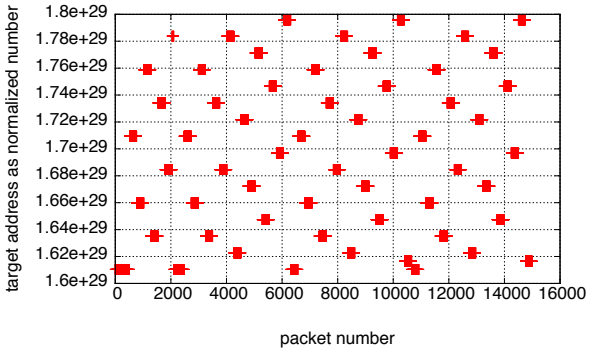

Figure 1. Sample for Scan Pattern 1: ICMPv6 Echo Request message-based network scan, apparently sent from the Karlsruhe Institute of Technology, scans multiple adjacent address spaces.

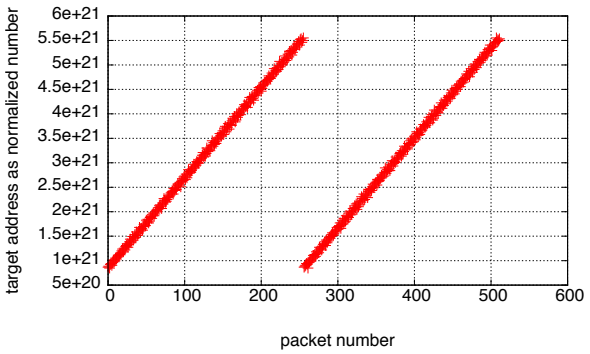

Figure 2. The first two linear scans of the scan set shown in Fig. 1.

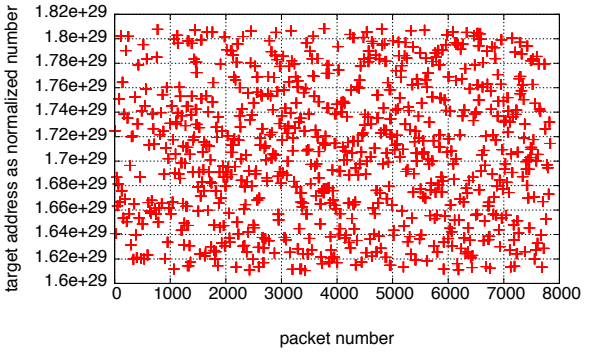

Figure 3. Sample for Scan Pattern 2: Visualization of the address space that was apparently contacted by AS34288.

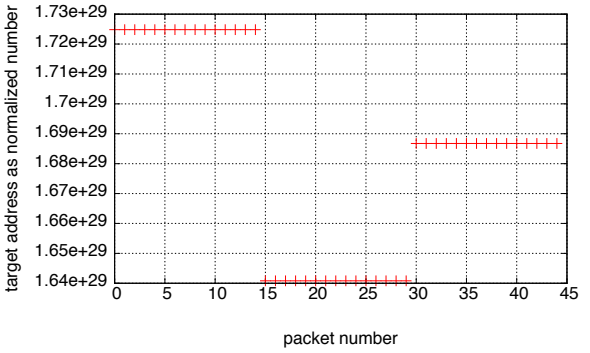

Figure 4. A closer inspection of the scan set shown in Fig. 3.

scans. We observed that the majority of these network scans matched one of the following two patterns:

Scan Pattern 1: The first scan pattern was mostly produced by a source pointing to the Karlsruhe Institute of Technology (KIT). Multiple individual Echo Request message based network scans sent a total of 14994 packets to 4352 unique destinations within a wide address space range. Except two of the received packets, all packets arrived with the same hop limit value of 57. Noticeable is the fact that all scans tried to contact only low-byte addresses ending with ::1 in a large number of different subnets. Fig. 1 visualizes the scanned address space.

A common characteristic of these scan samples is that the address space appears to be probed in an uniformly distributed manner. Individual scans having a small temporal distance scanned entirely different address spaces whereas, in some cases, temporal distant scans probed adjacent address spaces.

A closer inspection of the scans, depicted in Fig. 2, reveals that individual scans of this pattern sent probes to multiple adjacent destinations in an ascending order.

Scan Pattern 2: Various sources utilised a scan pattern which appeared to be much more scattered than the first pattern. An example scan, sent through the ISP AS34288[6], is shown in Fig. 3. Similar to the first scan pattern, the probes targeted low-byte addresses only.

[6]https://as34288.net

Although the target address space also appears to be uniformly distributed, a large number of individual scans covers relatively small independent address spaces. This can further be observed in Fig. 4. The Figure depicts the scan of three contacted destinations. In contrast to the first pattern, each individual scan sent fifteen packets to the same destination before trying a different, not necessarily adjacent destination. A closer look at the individual packets reveals that, in most cases, each host received packets with different hop limit values. Usually, a hop limit value between 1 and 5 was used, whereby a destination received three packets for each hop limit value in an ascending order.

2.3. Summary

We received many further network scans with similar characteristics, but we could not observe any more sophisticated scanning techniques.

Most of the received packets belong to large ICMPv6 network scans. It appears that many of these scans are conducted by universities or other research institutions. We found that the scanning methodology differs between multiple scans although some scans from different sources seem to apply the same scanning algorithm. A common characteristic of the majority of the observed ICMPv6 scans is that they only contacted low-byte addresses in various, not necessarily adjacent networks.

Although we received more packets than expected, only one in about $6 * 10^{23}$ addresses in our /34 network was contacted.

This proportion shows, that new architectural approaches are required to successfully deploy honeypots in sparsely populated IPv6 networks.

3. IPv6 Honeypot Requirments

The results of our darknet experiment show that IPv6 network scanners aim at a wide and unforeseeable address space range. Therefore, we present a new and efficient hybrid honeypot architecture for IPv6 networks which focuses on the handling of large IPv6 address spaces and which fulfils the following requirements:

R1: IPv6 Address Space Coverage - Honeydv6 supports large IPv6 address spaces by starting new low-interaction honeypots based on attackers' requests. We extend this approach to allow the dynamic deployment of high-interaction Honeypots.

R2: Genuine Service Emulation - Attacks to complex network services are handled by high-interaction honeypots running real operating systems with authentic network services.

R3: Price/Performance - The presented hybrid honeypot system utilises low-interaction honeypots to reduce the load on high-interaction honeypots. This reduction allows CERTs, universities and students to use the honeypot architecture without hiring cloud-based data centers.

R4: Honeypot Concealment - The handover between the low- and high-interaction honeypot-systems must be seamless in order to conceal the honeypot from an attacker.

4. Hyhoneydv6 Architecture

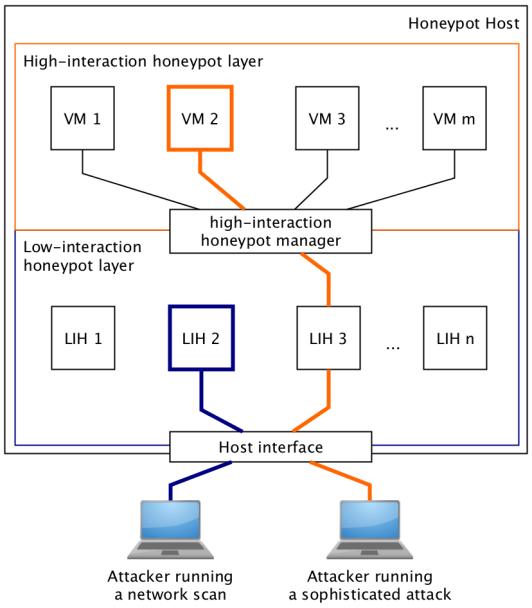

Figure 5. The hybrid IPv6 honeynet architecture processes network-scans on low-interaction honeypots and sophisticated attacks on virtual machine-based high-interaction honeypots.

Fig. 5 provides an overview over the components that are used in the architecture. A low-interaction honeypot layer dynamically instantiates honeypots based on attacker's destinations (R1). The instantiation is done randomly, based on a configurable probability. Simple network scans and attacks to less complex network services are processed in a low-interaction honeypot layer and therefore require a minimum of system performance. Sophisticated attacks are *transparently* forwarded to a high-interaction honeypot layer (R4). The high-interaction honeypot layer contains a limited number of virtual machine-based high-interaction honeypots which will be configured on-demand according to the requested destination address (R2). The entire architecture is placed on a single machine running off-the-shelf hardware (R3).

This section describes the main concepts and their implementation. Our hybrid architecture extends the low-interaction honeypot Honeydv6 [16] and is called *Hyhoneydv6*.

4.1. Dynamic Instantiation of high-interaction Honeypots

A newly implemented high-interaction honeypot manager starts and dynamically configures virtual machine-based high-interaction honeypot instances on demand. Our mechanism works similar to the dynamic machine instantiation of Honeydv6 which creates low-interaction honeypots based on attackers requests [16].

Fig. 6 shows the main components that are involved in our architecture. Incoming connections from an attacker first arrive at a low-interaction honeypot, which in this case is implemented by Honeydv6. Providing that the Hyhoneydv6 configuration defines a high-interaction honeypot mapping for the requested host and service, the high-interaction honeypot manager is requested to dynamically furnish a corresponding virtual machine instance. When this assignment between attacker and high-interaction honeypot is finished, the low-interaction honeypot starts to proxy all incoming packets to the high-interaction honeypot. Packets coming from the high-interaction honeypot targeting the attacker are still passed through Hyhoneydv6's customised network stack to keep control over the exchanged packets.

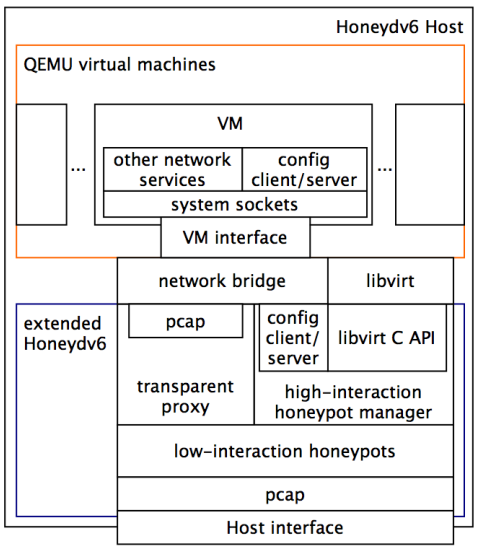

Figure 6. Components of the implemented architecture.

The honeypot manager uses the virtualisation library libvirt to create, backup and destroy QEMU-based [3] virtual machine instances [9]. We avoid a long machine startup time by working with a machine pool that contains already running machines. All virtual machines in the pool are booted automatically within the initialisation process of Hyhoneydv6. After a configurable timeout, a machine that interacted with an attacker is considered unused and an automatic memory backup will be created for forensic analyses.

4.2. Remote IPv6 Address Configuration

The forwarding of complex attacks to high-interaction honeypots requires a remote address configuration of the corresponding virtual machine. We decided to implement a custom IPv6 address configuration server which runs on the high-interaction honeypots and which interacts with the honeypot manager to reconfigure IPv6 addresses of the underlying host on-demand. The server installation is temporary and removes itself after finishing its tasks.

4.3. Attack Assignment

In [6], the authors create a new machine for every distinct source IP address. This approach allows observing attacks from different adversaries in an isolated manner because a stored machine state can be correlated to a single adversary. While this approach works well in IPv4 networks, it is not suitable for IPv6 networks. Due to IPv6 privacy extensions [13] and the allocation of larger address spaces, it is more probable that an attacker uses different IPv6 source addresses when executing an attack. We therefore decided to assign incoming network traffic to high-interaction honeypots based on the target addresses without considering the source addresses. We accept the fact that we may forward the traffic of more than one adversary to the same machine in the unlikely event that multiple attacks are targeting the same IPv6 destination.

4.4. Transparent TCP Proxy

A core component of the presented architecture is a newly implemented TCP proxy which extends the customised network stack of Honeydv6 to transparently hand over complex attacks from low- to high-interaction honeypots.

Our approach allows to configure a high-interaction honeypot with the same IPv6 address as an already existing low-interaction honeypot. All the main IPv6 header fields are copied when proxying traffic to a high-interaction honeypot without any changes to the high-interaction honeypot operating system.

Fig. 7 shows the operation of our transparent proxy mechanism based on a short example. An attacker first conducts a simple TCP SYN scan and aborts the 3-way TCP handshake with an RST packet. This is handled completely by the low-interaction honeypot. In a second attempt, the attacker establishes a TCP connection to the honeypot node with the IPv6 address *2001:db8::5* so that Hyhoneydv6 starts the IPv6 address reconfiguration of an available high-interaction honeypot.

An attacker may send further packets before the connection to the high-interaction honeypot is established. These packets are buffered and sent to the high-interaction honeypot when it becomes available. When the connection process has finished, Hyhoneydv6 sends all packets from the packet queue to the high-interaction honeypot before processing any further packets from the adversary.

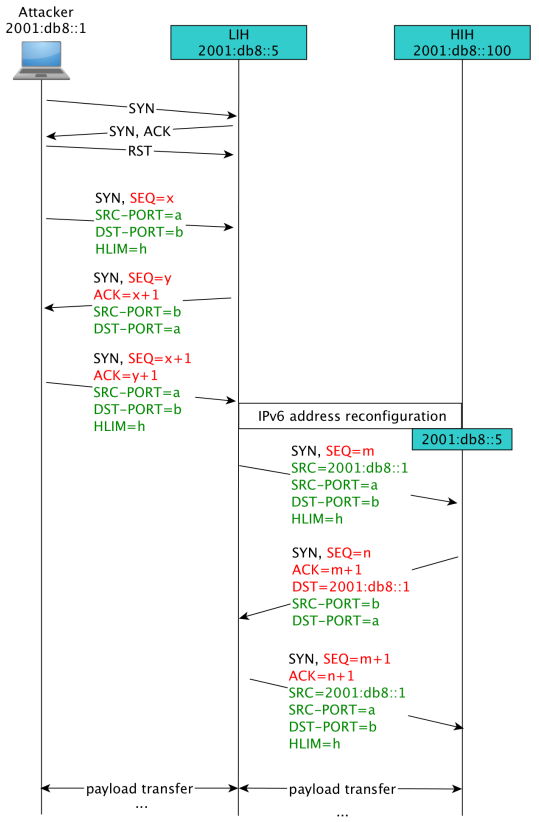

Figure 7. Implementation of a transparent TCP handoff.

The packet flow in Fig. 7 shows in green all header fields that are identical in both connections. This includes the source and destination addresses, the source and destination ports as well as the hop limit field. The TCP design does not allow us to communicate with the same sequence numbers without modifying the honeypot operating systems. This is a drawback of our architecture that we accept in favour of an uncomplicated deployment, requiring no network layer modifications.

5. Performance Evaluation

This section presents the results of our performance measurements and proves that simple off-the-shelf hardware is sufficient to run a hybrid honeynet which covers entire IPv6 networks.

5.1. Test Setup and Hardware Specifications

We conducted our tests with Hyhoneydv6 running on a desktop-class computer which is specified in Table II. Depend-

ing on the executed measurement, the Hyhoneydv6 host was running a varying number of QEMU-based high-interaction honeypots.

Table II
HARD- AND SOFTWARE SPECIFICATIONS OF THE HYHONEYDV6 HOST

Device/System	Specification
Operating system	Ubuntu 12.04 LTS
Qemu	1.0
Motherboard	EP45-DS3
CPU	Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz
Memory	4GB (2x2) 800 MHz
Network	RTL8111/8168/8411 PCI Express GE Ctrl. (r8169 Gigabit Ethernet driver 2.3LK-NAPI)
HD	SanDisk SDSSDP25 (read: 490MB/s write: 350MB/s)

Table III
EMULATED HARD- AND SOFTWARE SPECIFICATIONS OF THE VIRTUAL HONEYPOTS

Device/System	Specification
Operating systems	Windows XP (IPv6 support) / Debian 7.5 kern. 3.2.0-4-686 pae
Memory	256 MB
Network	Realtek Semiconductor, RTL-8139/8139C/8139C
CPU	QEMU virtual CPU

5.2. Initiating TCP Connections and Machine Reconfiguration

It is a crucial aspect in our architecture that the establishment of a connection can be done in a reasonable amount of time. A delayed response time caused by the IPv6 address reconfiguration may expose the honeypot infrastructure and should therefore be avoided.

We measured the time needed to establish SSH connections to Debian-based high-interaction honeypots. We conducted this measurement with a fully emulated and with a KVM-based Debian machine. The machines run an OpenSSH server to which we opened a connection while measuring the response time. We used two different measuring points in this experiment to determine the overall connection time from the attacker's point of view and the internal connection time from low- to high-interaction honeypot. For the overall connection time, we measured the duration starting from the first TCP SYN packet that leaves the attacker's interface card until the arrival of the first TCP packet that carried actual payload. This measurement was done using the network packet tracing tool Wireshark[7]. The blue bars in Fig. 8 show the corresponding results. The green bars depict the amount of time that Hyhoneydv6 consumes in the internal honeypot reconfiguration and connection stage. An internal timer measures the time needed to create the transparent proxy connection to the high-interaction honeypot as soon as Hyhoneydv6 received a connection request. In case of the first connection request, this

[7]https://www.wireshark.org/

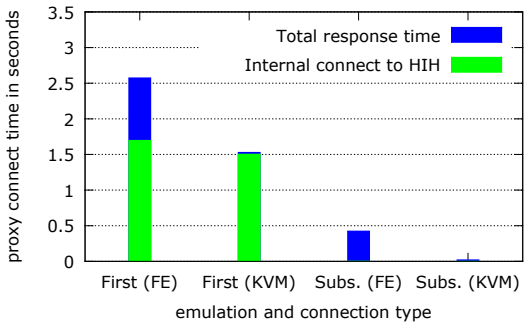

Figure 8. Time needed to establish SSH-connections to Debian-based high-interaction honeypots.

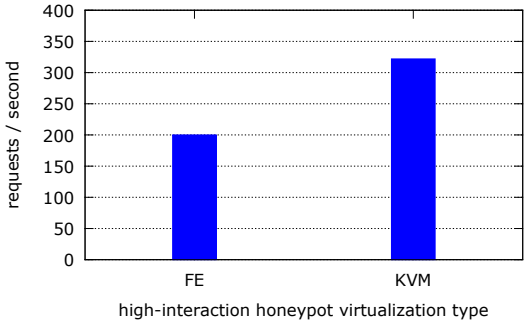

Figure 9. HTTP requests/second.

duration includes the time needed for the address configuration of the high-interaction honeypot. Each measurement was repeated 5 times and the median values are shown in the Figure. It takes approximately 2.5 seconds until a client receives the first TCP payload after sending the initial TCP SYN packet to the fully emulated machine. In contrast, about one second less is needed on the KVM-based virtual machine. However, we could observe only an insignificant acceleration of the internal high-interaction honeypot connection establishment and the address reconfiguration when using KVM. All subsequent requests could be processed in less than 0.5 seconds. By using KVM, the response time could be further reduced to about 0.02 seconds.

Another performance metric that influences a realistic user experience is data throughput. We choose to measure the number of HTTP requests that our architecture is able to process for an open connection. We installed the Apache webserver on a Debian high-interaction honeypot and used the publicly available benchmark servload version 0.9 [10]. We booted a single Debian machine instance and configured servload to request the default HTML test page up to 400 times per second. Fig. 9 shows the measurement results. The fully emulated Debian machine was able to process 200 requests per second while the KVM based machine performs about 320 requests per second. Hence, both machines types are more than sufficient for our application scope.

Table IV
CAPABILITY COMPARISON OF MAJOR HYBRID HONEYPOT AND HONEYCLOUD ARCHITECTURES.

Honeypot	IPv6 Support	Scan Handling Support	IP Address Updates	Dynamic Instantiation
Hyhoneydv6	**yes**	**yes**	**yes**	**yes**
A Hybrid Honeypot Architecture for Scalable Network Monitoring [2]	-	yes	no	no
GQ [6]	-	no	no	yes
Honeybrid [4]	no	yes	no	no
VMI-Honeymon [12]	-	-	no	no
Honey@home [1]	no	yes	no	no
A Hybrid System for Wireless Mesh Networks [15]	no	yes	-	no
Collapsar [8]	-	-	no	no
Potemkin [17]	-	no	yes	yes
An adaptive honeypot system to capture IPv6 adress scans [11]	yes	no	yes	yes
HoneyCloud [5]	no	no	-	yes

6. Related Work

We surveyed major existing hybrid honeypot architectures and large-scale honeyfarms and compared them against our architecture in terms of their IPv6 capabilities, their handling of network scans and their high-interaction honeypot deployment strategies. We evaluated the following properties:

IPv6 Support: Determines whether an architecture can be deployed in IPv6 networks. Possible table values are either *yes*, if an architecture supports IPv6 networks or *no*, if IPv6 is not supported. A dash character indicates that the IPv6 support could not be determined.

Scan Handling Support: The filtering or restriction of network scans unnecessarily prevents an attacker from finding hosts in the honeypot address space. If an architecture is able to handle arbitrary network scans and does not require filter mechanism, then this value is set to *yes*. A value of *no* indicates that the architecture either requires scan filters to reduce the system load or that the architecture restricts the scan support to certain address ranges.

IP Address Updates: Indicates whether the address of a high-interaction honeypot corresponds to the address that is expected by an attacker. Architectures which utilize NAT or similar mechanism to establish a connection to an attacker have an entry value of *no*, otherwise it is set to *yes*.

Dynamic Instantiation: This field is set to *yes* in case an architecture dynamically creates high-interaction honeypot instances. A fixed number of honeypots constrains an architecture to forward multiple attackers to the same honeypot instance or to restrict the number of attackers to the number of available machines. If an architecture uses a fixed number of high-interaction honeypots, then this field is set to *no*.

In the following, we elaborate these properties individually for the evaluated honeypot architectures. Table IV summarizes the results.

IPv6 Support

The survey shows that, even after more than 15 years after the publication of the initial IPv6 specification, there is still a lack of IPv6-compatible honeypot projects. Nine of the eleven architectures do not support or consider IPv6 networks at all. The only other architecture with IPv6 support is presented by Kishimoto et al. [11]. However, the presented solution relies on the IPv6 Neighbour Discovery (ND) protocol which limits its applicability to /64-networks, whereas Hyhoneydv6 supports arbitrarily large IPv6 networks.

Scan Handling Support

Only five of the eleven evaluated architectures are able to handle network scans without the requirements for filters. However, except Hyhoneydv6, none of the projects with scan support also support IPv6 networks.

All of the architectures which support the processing of network scans apply low-interaction honeypots for this purpose. For example, Bailey et al. use low-interaction honeypots to process so-called uninteresting traffic [2]. Traffic is considered to be uninteresting, if it contains known payloads or packets

that do not belong to existing connections, which includes network scans. Honey@home is another architecture which proposes the use of low-interaction honeypots to process network scans [1]. In contrast to the architecture by Bailey et al., only traffic that indicates malicious activity is forwarded to high-interaction honeypots. Similar to Honey@home, the hybrid honeypot architecture for wireless mesh networks applies low-interaction honeypots to process network scans and only malicious traffic is forwarded to high-interaction honeypots [15]. In contrast to the other architectures, Honeybrid provides a honeypot framework instead of a fully implemented architecture and leaves the handling of network scans to the actual implementation [4]. The example implementation presented by the authors uses Honeyd [14] to implement the low-interaction honeypot layer and may therefore fully support the handling of network scans.

The network scan filter approaches vary between the honeypot architectures. GQ uses multiple filtering mechanisms to reduce the load on the high-interaction honeypots [6]. For example, uninteresting traffic, such as network probes coming from a single source, is filtered while interesting traffic is forwarded to a small number of high-interaction honeypots. The Potemkin honeyfarm architecture dynamically instantiates new honeypots for each newly requested target address [17]. Potemkin implements traffic filters to filter out uninteresting traffic like network scans in order to limit the number of dynamically created machines. The architecture presented by Kishimoto et al. restricts the instantiation of new machines to addresses which are generated using the default EUI-64 algorithm [11]. Therefore, networks scans targeting arbitrary address spaces, which may include addresses generated with IPv6 privacy extensions [13], will never cause the instantiation of a machine. HoneyCloud instantiates a new virtual machine for each pair of source and destination IP address [5]. It only accepts traffic on the SSH port 22. This way, the architecture avoids the massive instantiation of new virtual machines through network scans to other ports.

In case of the Collapsar [8] and the VMI-Honeymon [12] honeypot architecture, it is not entirely clear whether the architectures are able to process network scans. In case of the Collapsar architecture, attackers communicate via so-called redirectors to a Collapsar backend, which contains a number of virtual machine-based honeypots. The authors do not provide any information about the handling of probe packets by redirectors. VMI-Honeymon builds upon Honeybrid and could theoretically be able to efficiently process network scans. However, the sample implementation of VMI-Honeymon differs from the proposed Honeybrid implementation and it is not clear whether the authors added a mechanism to filter network scans.

IP Address Updates

Three of the eleven evaluated architectures make sure that the IP address of a high-interaction honeypot matches the address that was probed by the attacker. Six of the evaluated architectures use proxy techniques, such as NAT, which result in different IP addresses on the high-interaction honeypots than originally requested. In two cases, the IP configuration was not further specified.

The Potemkin honeyfarm architecture [17] and the architecture presented by Kishimoto et al. [11] dynamically instantiate new machines on-demand. Both architectures deploy high-interaction honeypots based on incoming requests without the utilization of low-interaction honeypots. Kishimoto et al. remotely configure IPv6 addresses via SSH before forwarding an attacker to a machine. This approach, however, requires an SSH daemon running on the high-interaction honeypots which may reveal honeypot systems where an installed SSH daemon is unusual. It is also not clear whether Kishimoto et al. hide all traces of the IP address reconfiguration, for example, by deleting the corresponding shell history entries. Potemkin uses a different approach to reconfigure IP addresses. The Potemkin architecture includes a clone manager which creates new honeypot instances by cloning an already running Xen base machine image. As soon as the cloning process is finished, the clone manager sends an unspecified configuration packet to the newly instantiated machine. This configuration packet contains the desired IP address and triggers the address reconfiguration.

In six of the eleven evaluated projects, the addresses of the high-interaction honeypots differ from the originally requested addresses. Bailey et al. apply a proxy component, which forwards new and unknown connection flows from low-interaction to a fixed number of high-interaction honeypots [2]. The IP addresses on these honeypots stay unaltered and the proxy implementation is required to tunnel connections to the high-interaction honeypots. GQ employs NAT to translate the public IP addresses to the actual honeypot addresses [6]. Honeybrid's prototype implementation works with a fixed number of high-interaction honeypots and implements a Redirection Engine which proxies connections from low- to the high-interaction honeypots [4]. The process does not include an address reconfiguration. The same applies to the VMI-Honeymon architecture which is built upon Honeybrid [12]. Collapsar is another architecture which applies a fixed number of high-interaction honeypots in its backend. Network packets are directly injected into the virtualized NIC of the virtual machine-based high-interaction honeypots in order to avoid an IP address reconfiguration.

In case of the HoneyCloud [5] architecture and the hybrid system for wireless mesh networks [15], it is not clear whether an IP reconfiguration is applied. In case of the HoneyCloud architecture, the system description indicates that the public IP addresses are only configured on the HoneyCloud gateway, which is located between attackers and high-interaction honeypots.

Dynamic Instantiation

Honeypot architectures which work with a fixed number of honeypot instances may either restrict the number of parallel attackers to the number of available machines or forward multiple attackers to the same machine. This limitation can be avoided by the implementation of a dynamic instantiation

mechanism which creates new honeypot instances on-demand as required. Five of the eleven evaluated architectures, including Hyhoneydv6, use such a dynamic machine instantiation for high-interaction honeypots whereas all other architectures employ a fixed number of honeypot instances.

The architectures with dynamic machine instantiation support apply very different approaches. The GQ architecture utilizes a replay proxy component which learns from the communication between attackers and dynamically instantiated high-interaction honeypots [6]. The replay proxy handles known and interesting traffic as far as it has learned the corresponding traffic flow. As soon as GQ observes an uncommon and new packet flow in an attack, it replays and continues the communication with a high-interaction honeypot so that the replay proxy can observe and learn the new traffic flow. The high-interaction honeypots are controlled by a honeypot manager which is responsible for starting and resetting the machine instances. The Potemkin [17] architecture utilizes high-interaction honeypots only. It employs a sophisticated cloning and data transfer mechanism, called *flash cloning* and *delta virtualization*, to rapidly clone an already running base machine. The IPv6-specialized architecture presented by Kishimoto et al. [11] starts a number of virtual machine-based high-interaction honeypots in advance. An IPv6 router with ND-support connects these high-interaction honeypots to the Internet. Depending on various filtering mechanisms, an incoming Neighbor Solicitation message may trigger the address reconfiguration so that the corresponding honeypot is ready to interact with an attacker. An address assigning manager dynamically resets and reboots the honeypot instances into a clean state after successfully observing an attack. HoneyCloud provisions a new virtual machine-based high-interaction honeypot for each attacker on-demand [5]. To achieve this in an acceptable duration, a Cloud Controller component employs a pool of already started virtual machines. The Cloud Controller starts new machines when required to fill the machine pool and stops machines after a certain interaction timeout.

Survey Summary

At the time of writing, there is only one other honeypot architecture with limited IPv6 support available. All other presented architectures either do not support IPv6 or do not provide any information about their deployment in IPv6 networks. Less than half of the presented architectures are able to process network scans. A majority of the architectures either require scan filters to reduce the load on the honeypots or do not provide information about the processing of network scans. Although forty percent of the surveyed architectures dynamically instantiate honeypots, only a minority configures their IP addresses to the addresses that were initially probed by the attackers.

Hyhoneydv6 is the first hybrid honeypot architecture which supports all evaluated properties. The architecture can handle large-scale IPv6 network scans and provides dynamically instantiated high-interaction honeypots to cover entire IPv6 address spaces.

7. Conclusions

We have observed various network scans to unforeseeable addresses ranges in a 15-months darknet experiment. A common property of all major scans is a wide destination range, containing many different /64 subnets. This paper presents a hybrid honeypot architecture, called *Hyhoneydv6*, which can manage huge IPv6 address spaces and which performs well on off-the-shelf hardware. Low-interaction honeypots are used to process simple network scans while high-interaction honeypots handle sophisticated attacks. The high-interaction honeypots are dynamically created and configured on demand.

The transition from low- to high-interaction honeypots is supported by a proxy mechanism which allows low- and high-interaction honeypots to own the same IPv6 address in the same network. This mechanism gives the attacker the impression that he or she is working on a real host after being forwarded to a high-interaction honeypot.

Performance measurements show that *Hyhoneydv6* is able to configure a fully-emulated high-interaction honeypot and establish a new connection for a request to an arbitrary IPv6 address in about 2.5 seconds. By using a KVM-based high-interaction honeypot, the setup time is reduced to about 1.5 seconds. Once configured, even a fully-emulated high-interaction honeypot is able to process 200 transparently proxied HTTP requests per second on commodity hardware.

Hyhoneydv6 has the benefit that it needs no additional knowledge about typical scan methods in IPv6 networks and that it reduces hardware costs since an IPv6 network can now be simulated on a single host. Further, it integrates the superior analysis capabilities of high-interaction honeypots with network-wide monitoring capabilities of low-interaction honeypots. A survey of large-scale honeypot architectures shows that Hyhoneydv6 is currently the only available architecture providing these facilities.

8. References

[1] ANTONATOS, S., ANAGNOSTAKIS, K., AND MARKATOS, E. Honey@Home: A New Approach to Large-scale Threat Monitoring. In *Proceedings of the 2007 ACM Workshop on Recurring Malcode* (New York, NY, USA, 2007), WORM '07, ACM, pp. 38–45.

[2] BAILEY, M. D., COOKE, E., WATSON, D., JAHANIAN, F., AND PROVOS, N. A Hybrid Honeypot Architecture for Scalable Network Monitoring. Tech. Rep. CSE-TR-499-04, University of Michigan, Ann Arbor, Michigan, USA, October 2004.

[3] BELLARD, F. QEMU, a Fast and Portable Dynamic Translator. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2005), ATEC '05, USENIX Association, pp. 41–41.

[4] BERTHIER, R. G. *Advanced honeypot architecture for network threats quantification*. PhD thesis, University of Maryland, College Park, MD, USA, 2009. AAI3359256.

[5] CLEMENTE, P., LALANDE, J.-F., AND ROUZAUD-CORNABAS, J. HoneyCloud: elastic honeypots - On-attack provisioning of high-interaction honeypots. In *International Conference on Security and Cryptography* (Rome, Italy, July 2012), pp. 434–439.

[6] CUI, W., PAXSON, V., AND WEAVER, N. C. GQ: Realizing a System to Catch Worms in a Quarter Million Places. Tech. rep., University of California, Berkeley, CA, 2006.

[7] DURUMERIC, Z., BAILEY, M., AND HALDERMAN, J. A. An Internet-Wide View of Internet-Wide Scanning. In *23rd USENIX Security Symposium (USENIX Security 14)* (San Diego, CA, Aug. 2014), USENIX Association, pp. 65–78.

[8] JIANG, X., AND DONGYAN, X. J. Collapsar: A VM-Based Architecture for Network Attack Detention Center. In *Proceedings of the 13th USENIX Security Symposium* (2004), pp. 15–28.

[9] JONES, M. T. Anatomy of the libvirt virtualization library. *IBM developer Works* (2010), 97–108.

[10] JÖRG ZINKE AND JAN HABENSCHUSS AND BETTINA SCHNOR. servload: Generating Representative Workloads for Web Server Benchmarking. In *International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS)* (Genoa, Italy, July 2012), vol. 44, IEEE Communications Society, pp. 82–89.

[11] KISHIMOTO, K., OHIRA, K., YAMAGUCHI, Y., YAMAKI, H., AND TAKAKURA, H. An Adaptive Honeypot System to Capture IPv6 Address Scans. In *Cyber Security (CyberSecurity), 2012 International Conference on* (Dec 2012), pp. 165–172.

[12] LENGYEL, T. K., NEUMANN, J., MARESCA, S., PAYNE, B. D., AND KIAYI, A. Virtual Machine Introspection in a Hybrid Honeypot Architecture. In *5th Workshop on Cyber Security Experimentation and Test* (Berkeley, CA, 2012), USENIX.

[13] NARTEN, T., DRAVES, R., AND KRISHNAN, S. Privacy Extensions for Stateless Address Autoconfiguration in IPv6. RFC 4941 (Draft Standard), Sept. 2007.

[14] PROVOS, N., AND HOLZ, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Addison-Wesley, 2008.

[15] RAWAT, P., GOEL, S., AGARWAL, M., AND SINGH, R. Securing WMN Using Hybrid Honeypot System. *International Journal of Distributed and Parallel Systems 3*, 6 (2012).

[16] SCHINDLER, S., SCHNOR, B., KIERTSCHER, S., SCHEFFLER, T., AND ZACK, E. IPv6 Network Attack Detection with HoneydV6. In *E-Business and Telecommunications*, M. S. Obaidat and J. Filipe, Eds., vol. 456 of *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2014, pp. 252–269.

[17] VRABLE, M., MA, J., CHEN, J., MOORE, D., VANDEKIEFT, E., SNOEREN, A. C., VOELKER, G. M., AND SAVAGE, S. Scalability, Fidelity, and Containment in the Potemkin Virtual Honeyfarm. In *Proceedings of the Twentieth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2005), SOSP '05, ACM, pp. 148–162.

