Scan Prediction and Reconnaissance Mitigation through Commodity Graphics Cards

Christian Doerr, Mourad el Maouchi, Sille Kamoen, Jarno Moree
Delft University of Technology
Cybersecurity Group, Department of Intelligent Systems
Delft, The Netherlands

***Abstract*—In order to protect ICT systems against remote attacks and exploitation, insight into which systems are targeted is necessary as soon as possible. Given the lack of advance information, current network-based attack detection and mitigation techniques, such as virus scanners or intrusion prevention systems, are typically aimed at countering the delivery and exploitation. This paper presents a novel approach capable of detecting threats while they scan a local network for potential targets and even before an intrusion attack has been made. This allows the defender to single out scan traffic and selectively deny access to an attacker performing reconnaissance while maintaining the availability to other users. We implement a proof-of-concept on commodity graphics cards, and demonstrate fast prediction of scanner behavior on a /16 network telescope.**

I. Introduction

When connecting a new host to the Internet, usually within seconds to minutes the first scan packets arrive that probe the machine for open ports and the presence of common services. These network scans are usually the precursor for a subsequent exploitation attempt or attack, either by the scanning party itself or by other cyber criminals which have bought lists of potentially exploitable hosts on the black market [1]. In the following attack phase on services such as SSH, telnet, FTP, HTTP, DNS or mail, the attacker then aims to gain control over the machine through an insufficiently secured login, misuse the host as a dropbox for files or the delivery of malware, act as an amplifying mirror in DDoS attacks, or as a relay for SPAM.

The hosts and the level of sophistication by which they are exploited depends on the setup, capabilities and intentions of the adversary. In order to be effective, cyber defense thus needs accurate, actionable intelligence on threats and their capabilities, and the likely targets pursued by a particular attacker. Such information is hard to come by. Additionally, many of the network and communication security tools are tailored to detect threats when they are actively trying to exploit a system (e.g., virus scanners, IPS) or aiming to prevent information exfiltration (e.g., diodes). In terms of the "kill chain" model by Hutchins et al. [2] shown in figure 1, this however means that threat detection happens at a much later stage than desirable: First, unsuccessful detection during the delivery and exploitation phase means that a system has been compromised. Second, if the defender is building up the first line of defense at these stages in the attack progression chain, little, if any, advance warning time is available about an immanent attack. It would thus be beneficial if intelligence about an adversary and a potential attack could be collected beforehand, that can either be used to break the attack chain as early as possible, or to inform the processes in later stages to maximize detection rates.

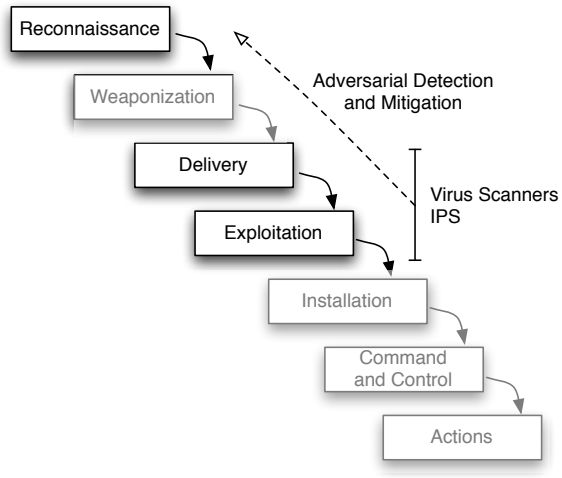

Fig. 1. By enabling an attack detection already during the adversary's reconnaissance phase, the attack progression can be identified and stopped earlier than in case of other defense methods. Alternatively, the defender can use this knowledge and gain further intelligence during the expected subsequent delivery and exploitation.

Regardless of their motivations and different capabilities, both script kiddie and cyber criminal alike first need to probe the network which hosts and services are active prior to subsequent steps. With network scans being the precursor to network-based attacks and exploitation[1], an effective mining of incoming probe packets can provide insights into which services and hosts are targeted. In order to load-balance and minimize the impact on remote networks, high performance port scanners such as ZMap [3] or masscan [4] generate the sequence of to-be scanned candidates using a pseudo-random number generator. If the next likely random number and targeted host can already be determined in real-time during a scan, a defender can thus interrupt and influence the remote

[1]We note that an adversary has still many other attack vectors ranging from social engineering to hardware modifications to establish a presence in the network, which are not in scope of network-based attacks outside the defense perimeter and need to be treated differently.

reconnaissance by the adversary.

This paper presents a novel approach to exploit the IP selection behavior of the common scan tool ZMap to predict the next host that will be scanned in real-time. This opens up new mitigation strategies to the defending network:

1) to selectively drop expected probes by temporarily adjusting the firewall and prevent the remote party from gaining insight into which resources are available locally, and
2) to obtain further knowledge about the intentions of the remote party by analyzing target's hosts and ports and preemptively react by spinning up a honeypot at the next targeted host.

The approach is especially attractive since the scan prediction and reconnaissance mitigation does not require new equipment, but is deployable using today's typical networking hardware. In this paper we report an efficient implementation on commodity graphics cards and results in a /16 network block that demonstrate that scan targets may be predicted and mitigated within a few minutes.

This paper is structured as follows: in section II we describe the state of the art in network scan analysis. Section III introduces the concept of port scanning and the target-generating algorithm of ZMap. Section IV outlines the network telescope used in our measurements. Section V elaborates on the techniques for an efficient real-time prediction of scan sequences on stream processors of commodity graphics cards. Section VI reports results and evaluates the feasibility of this approach for different network sizes and available resources, Section VII outlines potential mitigation options based on these results. Section VIII concludes our work and summarizes our findings.

II. Related Work

As discussed in the introduction, the majority of work in network defense has focused on detection and mitigation techniques in the later stages of the "kill chain" than the work presented in this paper. Virus scanners typically aim at a discovery in the delivery phase, while for example host-based intrusion detection and prevention systems try to prevent the actual exploitation happening. Typically these systems such as bro, snort, deny hosts or fail2ban define specific thresholds, which – if exceeded within a certain time frame – trigger a predefined action. Network diodes intend to limit or at least slow down network traffic associated with the remote controlling of malware in the C&C phase.

Obtaining intelligence about adversaries already through their reconnaissance phase has received almost no attention yet in the past. Yegneswaran et al. [5] propose and introduce one of the first setups to obtain threat intelligence by collecting packet traces on a honey network, a collection of individual honey pots. They demonstrate the scanning behavior exhibited by select malware, which by software shows random or highly structured IP exploration patterns.

Allman et al. [6] conducted a longitudinal survey for scanners over the period of 12 years. They report a massively increasing volume of scans in the Internet, and also find two classes of scanners being present: (1) "heavy hitters", scanners that send massive amounts of packets in as little time as possible, and (2) a group which scans hosts at a slow continuous speed. The authors noted the difficulty in collecting and analyzing such traces due to the constraints in memory, as they could only process and correlate data in chunks of 24 hours each. This makes it challenging to identify adversaries which deliberately limit the amount of packets sent and act over longer periods of time.

A significant body of work has been done on detecting specific types of attacks, for example brute-forcing of SSH servers. Javed and Paxson [7] report the existence of stealthy, low-rate attacks, where advanced adversaries coordinate from multiple resources and minimize the amount of probing to stay under the radar of the defender. As such brute-forcing attempts are typically precursed by a port scan identifying the existence of these services on select hosts in the first place, a stealthy brute-forcing action might actually be prevented by identifying the reconnaissance scan.

Durumeric et al. [8] are among the first to exploit specific header fields to detect scanners. They note that ZMap (which that group authored) may be identified by the use of 54321 as the IP ID. They cluster scan probes based on source IP and perform statistics which ports scanners have been targeting historically, but neither conduct a real-time analysis to obtain situational awareness nor dissect the scanning patterns themselves in further detail.

III. High-Speed Port Scanning

Whether or not a port is open and a service exists behind it answering requests can be determined remotely based on the hosts' reaction to a probe packet. As shown in figure 2, TCP sessions begin with a three-way handshake, where the initial TCP SYN by the client is responded to with a SYN+ACK by the server if it is willing to establish a connection. In case no service exists at the requested TCP port, following RFC793 [9] the operating system will reply to an incoming SYN packet with a RST. Since UDP packets do not involve an application-independent connection establishment phase, a UDP frame to an open port may or may not be answered by the attached application. However, a request sent to a closed port is typically answered by an ICMP Port Unreachable reply by the operating system unless filtered out.

While SYNs are by far the most dominant TCP scan method[2], other methods exist testing transport-layer specific differences for closed and open ports. These methods are sometimes applied to prevent detection by default rules in intrusion detection systems and firewalls. Since each of these methods involves sending a packet to the probed host, these nuances do not matter for the detection methodology described in this paper.

Building a high-performance port scanner is not trivial. The naive approach of attempting a connection through the sockets

[2]The longitudinal survey in [10] shows that 88% of TCP scans use SYNs.

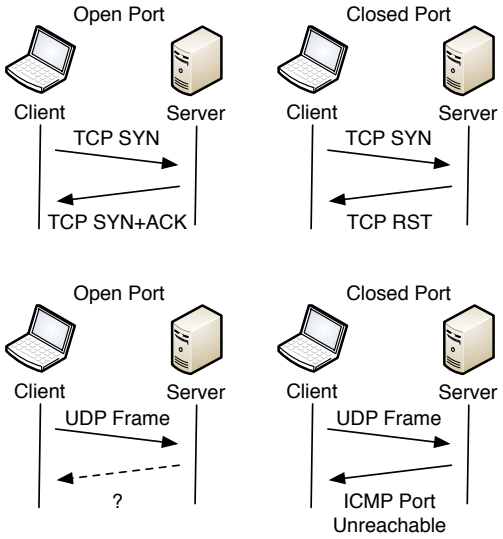

Fig. 2. A TCP packet with a SYN set will be answered with a SYN+ACK by a server with an open port, and a TCP packet with a RST flag in case of a closed port. In UDP, a frame to a closed port generates an ICMP Port Unreachable response unless filtered.

provided by the operating system is impracticably slow. This is caused by the internal timeouts, the maximum connection pool sizes and, if successful, the clog up of resources such as transmission control block allocations. This is why network probes are normally sent using specialized, dedicated port scanner applications. While tools such as nmap are nearly two decades old and well established, even they are not efficient enough to scan large blocks of IPs or even the entire IPv4 space within a relatively short time.

This has recently led to the emergence of a few next generation network scanners, specifically tailored to scan the entire Internet in a short period of time. One of the major scanners, ZMap [3], is capable of scanning the entire IPv4 space on a 1 Gbps connection in under one hour. In order to achieve such throughput, a variety of optimizations are required, such as: the reduction or elimination of connection state, the injection of packets as low as possible within the host's operating system and network stack, often bypassing the stock network stack and replacing it with its own streamlined version, or even the implementation of optimized network interface drivers. Given this significant amount of knowledge and investment necessary to create a high-performance scanner, scanning activity in the Internet almost exclusively relies on these publicly available open source tools [10]. In this paper, we focus on ZMap, which together with Masscan [11] is the major high-throughput scanning software.

A. Scan Target Selection

If scanning a range of IP addresses, the naive approach of moving through the space in a monotonically increasing sequence will lead – due to the block-based allocation of addresses to organizations – to temporary targeting of a particular

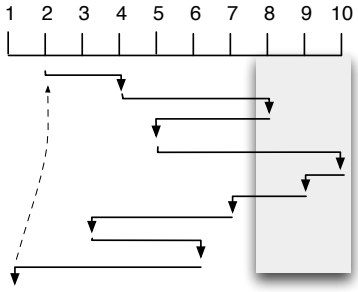

Fig. 3. Sequence of numbers generated by 2 in the multiplicative group modulo 11.

organizations with probes. This is undesired for two reasons:

First, as remote scans (for example for security purposes as described in [3]) are not commissioned by the remote party but typically only tolerated, it is good practice to keep the resources consumed as low as possible and minimize the chance to interfere with the normal operation of the network, both in terms of total packets sent over the duration of a network scan but also in terms of the maximum amount of probes launched into a single organization's network at any given time. This is especially true for high-speed scanners which can easily generate volumes exceeding a million packets per second.

Furthermore, firewalls, log monitors and intrusion detection systems may be monitoring data traffic and are frequently configured to block or generate an alert if a threshold of a certain number of "unusual" packets is exceeded. In this case, unusual packets may mean anything from traffic originating from unauthorized source IP addresses to packets sent to hosts and ports that do not exist. For this reason, it is also in the interest of a scanner foregoing detection to minimize the amount of probes sent into a given network within a particular time interval.

Scanners that select targets at random would hence need to keep record of which IP addresses already have been scanned. Nmap, for example, keeps such a list and also logs which hosts have and have not responded to a query to account for and retry probes potentially lost in the network. However, saving information about the connection state comes at a significant expenditure. ZMap avoids this and instead generates scan candidates algorithmically using a multiplicative group of integers modulo p.

ZMap's source code defines a set of primes and generators to be used for scans of different network sizes. In case of a /0 scan, i.e., a test of the entire IPv4 space, the list of IP addresses is obtained using the primitive root 3 and the next prime p larger than 2^{32}, 4,294,967,311 ($2^{32}+15$). As shown in figure 3 for the generator 2 in the multiplicative group modulo 11, this number-theoretic approach ensures that all IP addresses are scanned exactly once during a scan cycle while keeping minimal internal state.

Since the same generator would recreate the exact same

sequence of IP addresses in every scan and on every host running the ZMap software, networks at the beginning of the sequence would be over-proportionally targeted due to configuration tests, trial runs and aborted scans. ZMap thus finds a value k such that 3^k is co-prime to p, which results in a valid generator. The candidate k is derived from an AES-encrypted word modulo p, and successively incremented by one until a valid generator is found. Each valid k will thus generate a unique sequence $3^{k \cdot 1}$, ... $3^{k \cdot (p-1)}$ in the IPv4 space.

IV. Scanning the Scanners

When observing incoming traffic on a host directly connected to the Internet, one can differentiate between three types of traffic: (1) user traffic sent for example in response to previous request, (2) backscatter from attacks reflecting off other hosts where the attacker has randomly spoofed the source IP, and (3) probing traffic from hosts in the Internet that scan the local network for available local machines and open ports.

While firewalls typically weed out (2) and (3), they can be mined to provide insights into ongoing attacks and adversarial intentions. In this paper we utilize network probes which can be separated from backscatter at the IP layer and transport protocol layer. TCP backscatter from a DDoS attack has the TCP SYN+ACK flag set, while a scan packet will feature only the SYN packet. Separating UDP packets into backscatter and scan traffic requires some more advanced protocol-based heuristics, determining for example for DNS whether the UDP payload contains a DNS request (scan) or DNS answer (backscatter). We have implemented protocol parsers for the most commonly occurring applications and protocols. In addition to protocol- or payload-based methods, additional hints also exist to identify network probes. Stock ZMap for example also identifies itself by inserting 54321 as the IP identification number of its generated packets.[3]

Obtaining scan probes for threat mining can be achieved in two ways: First, a port mirror on a pre-firewall router could copy all incoming packets to a separate host for analysis. As this can be accomplished using only a configuration change in most enterprise routers, this allows for an easy, immediate deployment of the methods described in this paper as no additional network components or dedicated IP addresses are necessary. Second, traffic traces may also be collected by observing a block of dark IP addresses – at which thus only backscatter and network scans would arrive.

The results in this paper are based on incoming data from a /16 IP block, which has been used as a network telescope for about 15 months. In the absence of any user data, the telescope receives approximately 15 GB of backscatter and network scans per day. From this 7 TB repository, backscatter was excluded by above methodology and the month of April 2015 selected for analysis. 190 million scan packets were received over the course of this month that could be attributed to ZMap by IP ID, 14.4% of which were UDP frames.

[3]While it is of course feasible to change the default IP ID in the source code – and we in fact do see several parties implementing such alternations to their copy of the open source software –, we can still detect and identify scan probes through other behavioral means in incoming traffic which is beyond the scope of this paper.

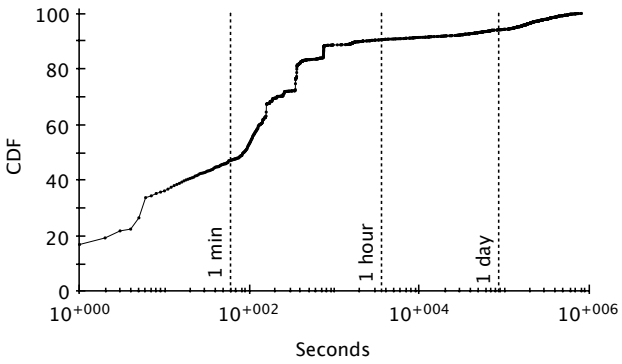

Fig. 4. Cumulative density function (CDF) of the average interarrival time between scan probes by scanning source IP.

During the 1 month observation period, a total of 13664 unique IPv4 addresses scanned the /16 network. The amount of packets injected by source IP varies drastically. There exist some which only inject a handful of packets over the course of one month, while a few hosts send thousands of packets per hour. From the perspective of the existing network defense portfolio, the latter ones are of less concern as their noisy appearances will trigger existing IDSes. However, a sizable number of scanners exists that send probes significantly spaced apart that would not result in any action by normal network defense systems. Raising an alarm or applying a counter-measure on the basis of one packet received in a 5 minute interval would be impractical. It is thus especially interesting to find new methods that can discover such slow scanners operating almost continuously at a very low pace, either out of courtesy or to evade detection. In our dataset, we have observed that some scanners reduced their impact even further by cycling through blocks of IP addresses in a round-robin fashion. However, this may be detected due to their interleaved arrival and the fact that their union can be uniquely traced back to a sequence.

Figure 4 shows a plot of the cumulative density function (CDF) of the interarrival time of scan packets, averaged by the scanning source IP and, in case multiple scans were launched by one IP, also separated by runs. The figure shows that the slowest 50% of scanners had an average interarrival time of more than 90 seconds between probes, while the bottom quantile of scanners were injecting packets spaced over 6 minutes apart. The adversarial reconnaissance detection technique presented in this paper will be particularly of use in situations of slow (and potentially distributed) scans which cannot efficiently be detected or prevented using the current state of the art. We notice that the ability to predict and react to probes within a time frame of 1-2 minutes will be sufficient to effectively stifle 50% of the scanner population. If not all IP addresses in one's organizational network are used or it is acceptable that a scanner may successfully probe a handful a packets before being shut out, a scan prediction and reconnaissance mitigation needs to determine the scan seed in

the order of 10 minutes to be sufficiently reactive.

V. Real-Time Prediction of Multiplicative Group Generators

Since the generator for the multiplicative group, $\mathbb{Z}_p$, is chosen randomly by the scanning host as discussed in section III, neither the starting seed nor the exact resulting sequence is known by the targets. With the prime p and the primitive root 3 specifically chosen and hard-coded within ZMap, knowledge of which IP addresses have been scanned in which order is however enough to determine the starting point of the sequence. This in turn predicts which other hosts will be targeted. In combination with the interarrival time of the previously observed probes – if proportional to their position in the generated sequence – an approximation when the next IP in the organization's network will be scanned can also be provided.

Consider the situation that a network is monitoring a set of IP addresses A, ... Z for incoming probes. As shown in figure 5, each valid $k \in (0, 2^p)$ will generate a different sequence of IP_1^k, ..., $IP_{2^p-1}^k$. While the worst case computational complexity is in theory $\mathcal{O}(2^{64})$, the search for a seed becomes practical due to a number of computational and algorithmic optimizations discussed in this section.

A. Bounding the Search Algorithmically

Suppose scan probes have been received by a local network first by host A, then host C, followed by host B and D. Figure 5 displays the evaluation of three k up to a generation depth of j and the three situations that may occur:

1) **Out-of-order termination.** Imagine that first IP the sequence generated by i addresses that are observable to the receiving network in will be A followed by D. Since a probe had been received by A and at least one other host before D was targeted, the evaluation of this generator can be immediately terminated.
The likelihood of out-of-order sequence hits drastically increases with the size of an organizational network, and in result the probability that individual seeds can be abandoned early on. Consider a scan with probes sent to k IP addresses of which n have been observed by the defender. The likelihood for an IP address that has received a scan packet to be not in sequence is $\frac{n-1}{n}$, the likelihood that a seed may be abandoned after 2 trials is $\frac{1}{n} \cdot \frac{n-2}{n-1}$, and for a mismatch at the third observed IP $\frac{1}{n} \cdot \frac{1}{n-1} \cdot \frac{n-3}{n-2}$. Generalizing for a mismatch at the j's incoming probe, the probability of seed abandonment becomes
$$p_a(j) = \frac{(n+1-j)!}{n!} \cdot \frac{n-j}{n-j+1},$$
which as shown in figure 6 already eliminates close to 98% of all valid seeds after two received packets. The scans of the locally observed IPs occur with the sequence of, in the worst case, 2^{32} IP addresses per generating seed. This naturally poses the question about the expected length of a sequence that needs to be evaluated before a starting value may be abandoned. If the first IP out of n total observed IPs appear at position j of the sequence, the length of the sequence until the first element can be derived as
$$\sum_{j=1}^{k-n} \frac{n}{k} \cdot j \prod_{i=0}^{n-1} \frac{k-j-1}{k-i},$$
or in the general case for position l
$$\sum_{j=l}^{k-n+l-1} \frac{n}{k} \cdot j \prod_{i=l}^{l-1} \frac{j-i}{k-i} \prod_{i=l}^{n-l} \frac{k-j-1}{k-i} \binom{n-1}{l-1}.$$
Figure 7 shows the average number of packets that need to be evaluated per sequence as a function of the received packets in the network. While naturally no sequence may be abandoned after the first probe, having received four packets by a scanner enables the defender to make a decision on average after approximately 10^9 evaluations, a 75% speed-up.

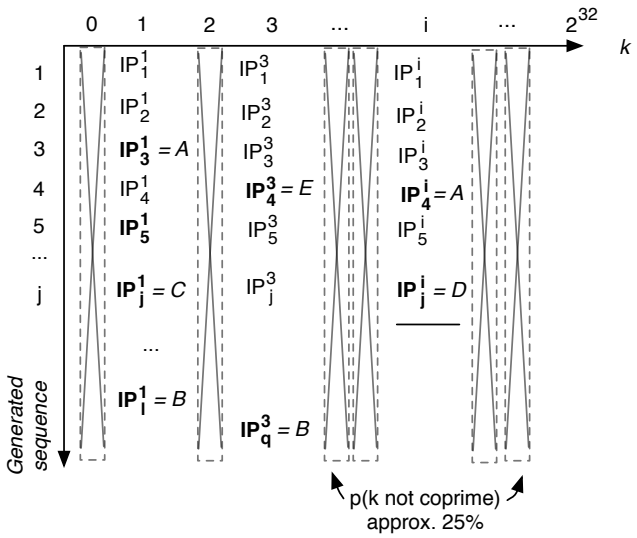

Fig. 5. Parallel exploration to identify the generator using incoming the sequence of probes.

2) **Accommodation of packet loss.** While the first IP to fall within the range of the target network that is resulting from $k = 3$ is E, a probe packet has not been received by this host. Since the scan to E could have been lost en-route in the network, a sequence with an IP at which no scan was observed needs to be continued, and may only be terminated in step q for $IP_q^3 = B$.
As the size of the observed block increases, the probability that multiple probes were lost and thus not observed by the network decreases drastically. Given a packet loss probability of p, a threshold t can be set so that correct sequences are prematurely and incorrectly terminated with less than probability m: $p^t < m$.

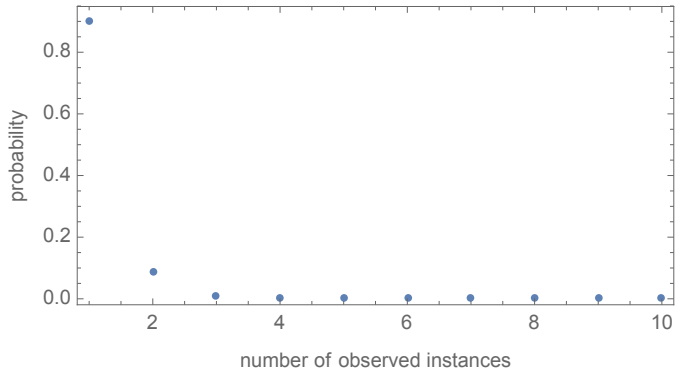

Fig. 6. Abandonment probability for k=2^{32} and n=2^{16} as a function of received packets.

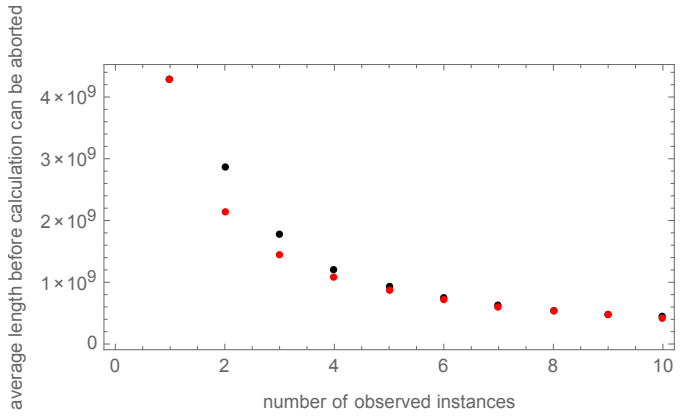

Fig. 7. Average length of sequences that need to be observed as a function of observed packets.

3) **(Sub-)sequence match.** A k delivering the exact sequence of probes observed at the network's IP addresses may indeed be the original generator value used by the remote party. Given a small network size, there may however exist more than one generators that would produce A, C, B, D. While the likelihood of such events can be bounded by monitoring more IPs analogue to the two cases above, our implementation continues all valid sequences and feeds them into the mitigation chain described in section VII. While with each additionally received packet more and more candidate sequences can be abandoned, the parallel evaluation of a limited set of chains ensures a timely proactive reaction while minimizing the amount of treatment options.

B. *Efficient SIMT-Seed Search on GPUs*

Given the independence of all possible k and the minimal state that needs to be kept for the computation, this problem lends itself naturally for concurrent evaluation using parallel computing platforms provided by modern graphics card. In addition to algorithmic pruning discussed above, additional significant speed-ups may be reaped through implementation-specific optimizations when implementing the seed search in the single-instruction-multiple-threads (SIMT) model provided by CUDA.

The basic idea behind the discovery of the seed and predicting the sequence of the scan is simply exploring each seed value to find the one that would generate the observed sequence. While there are in principle $2^p - 1$ starting points, the search for a seed may first focus on a exploration of $2^{30} + 62836735$ initial values, as the primitive root and prime are hard-coded into ZMap and approximately 25% of all k are not co-prime. Instead of saving these co-prime values, these can be computed before a GPU call on the CPU. Since the memory on GPUs is limited, the amount of seeds is split up in chunks and transfered to the GPU. With multiple GPU cards, the computation of the chunk of seeds can be done right before the call on the next GPU. Since this can be done asynchronously, maximizing the available time.

While the algorithm in itself is trivial – for each candidate seed, compute all powers from 0 to 2^{32} modulo p, compare the resulting value with the observed IPs, and continue or abandon seed as a result of the comparisons –, implementing such algorithm on the stream processors of consumer graphics cards and their SIMT computation model would provide unacceptable low throughout values, as (a) in case of branches all cores need to evaluate all options and (b) groups of threats (in CUDA-terms warps, or wavefronts in OpenCL-lingua) – typically 16 or 32 – execute the same instruction. As from above theoretical analysis we know already that close to 98% of all seeds may be abandoned after 3 iterations, this results in practice in the majority of threats inside a warp to be idle after a minuscule fraction to time. Thus, all cores in a warp can only start on a new thread when all other cores are finished.

Given this unusual, yet independent, workload for graphics cards, we thus designed a seed sieving algorithm that optimizes the high seed-abandonment likelihood with the underlying SIMT model. Figure 8 shows a schematic overview of the sieving procedure. As the data transfer between CPU and GPU is comparatively slow, the CPU feeds lists of seeds asynchronously into a dedicated memory region on the GPU. Whenever a thread group has terminated, it fetches the next block of seeds from the seed region while decreasing the counter $c1$ of stored elements. Instead of pursuing each starting values for up to 2^p iterations or until a sequence match or mismatch has been found, each thread group uniformly pursues each new seed for only l iterations. If a seed is discarded, a particular thread is thus only idle for the at most remaining $l - 1$ iterations; if a seed is still a feasible option at the end of the exploration, the original seed s_i, the current value (s_i^l) (or $s_i^{i \cdot l}$ after i runs) as well as a map of which IPs have been hit is serialized into a separate buffer. The number l may be dynamically chosen and the optimal value depends, as shown above, on the total number of observed IP addresses, the level of acceptable packet loss, and false negative ratios.

Threads first feed off the list of available seeds, clearing the space for the CPU to asynchronously repopulate the seed region to eliminate slowdowns from the comparatively slow

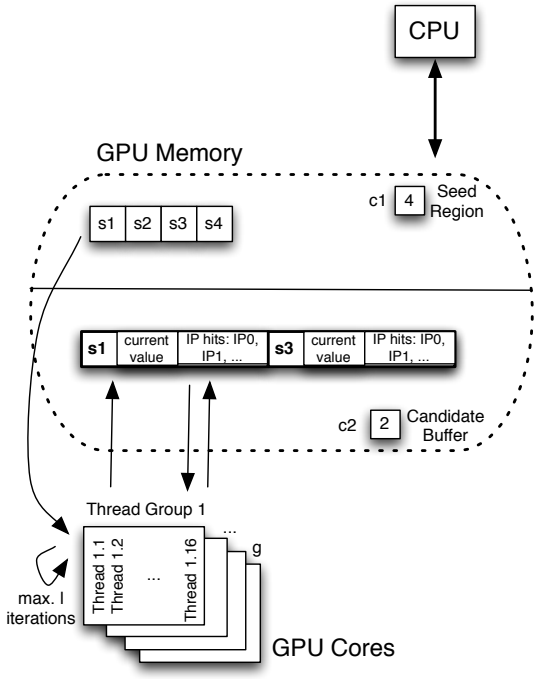

Fig. 8. A two-phase sieving process optimized for the specific workload of the seed exploration, optimizing for CPU/GPU latency, high abandonment ratios, and SIMT instruction.

memory transfer. If counter c_1 shows the absence of seeds, threads pop seeds off the seed buffer and re-push the seed after l iterations if no decision could yet be made. The comparatively high abandonment ratio (for l below a few hundreds of thousands of iterations) means that the two lists stay balanced and the seed buffer is not overflown. The search can stop if both $c1$ and $c2$ are 0, indicating that in this block no feasible candidate exist.

The split into two phases of a fixed iteration count obsoletes the need for any thread group coordination and expensive mutex operations. Such actions might consist of keeping a counter on how many threads are still active and stop processing on high idle ratios, while at the same time reducing thread divergence as much as possible. The above algorithmic- and platform-specific optimizations together with a few implementation-specific tweaks result at a problem of in theory $\mathcal{O}(2^{64})$ can be efficiently computed in practice and effective lyused for defense purposes. In the next section, we report performance benchmarks obtained based on commodity cards and data from our /16 network telescope.

VI. Experimental Evaluation

The basis for our evaluation platform and proof-of-concept mitigation prototype was the nVidia GTX 680 card with 1536 cores. While not optimized for integer operations such as the Maxwell chip featured in say the Titan X (through which the benchmarks in this section are significantly improved), the no longer state-of-the-art GTX 680 features a major benefit for an academic lab: cost per unit.

Since the workload can not only be parallelized across graphics cores but also across graphics cards themselves with proportional speedup, a scale-up using slightly older but more consumer cards provides a powerful computing platform at a comparatively low expense. In this section, we report evaluation benchmarks from a scaled-up parallelization execution on GTX 680s worth approximately 5000 Euros at the time of writing, buying a cluster of 51 consumer cards. This is sufficient to achieve fast turnaround times for the scan prediction and threat mitigation described in this paper, while at the same time being a negligible expense by the standards of organizations that have a large enough network footprint and public IP range where scan prediction is necessary in the first place. As the results scale perfectly linear, the reader may scale-up or down the performance results presented below for differently-sized installations.

The evaluation in this section will focus on four critical important aspects determining the applicability of the presented approach in practice: First, as an increasing number of packets will rule out more and more seeds, we compute the overhead reduction in possible solutions as the scan progresses. Second, the next subsection provides for our utilized network telescope the scan prediction run times. Third, for the above GPU cluster and the distribution of the bottom half of scanning speeds in figure 4, we will analyze the expected reaction time and mitigation speeds towards scanners. Finally, as a sizable organization may be scanned by multiple sources at the same time, we demonstrate in the fourth subsection that a scan prediction for multiple scanners does not require a multiple of the run time of the individual seed searches, but that also here the prediction of multiple threats may again be efficiently parallelized.

A. Candidate List

Upon the receipt of the first scan packet at an IP, all of the $2^{30} + 62836735$ seeds could have been used to generate the sequence that is followed by the scanner. As the second probe comes in, half of them will be out of order and may be abandoned per the rules described above, and so on. This however means that while at the beginning of a scan the relatively sparse information will result in a large number of possible starting values and thus also many “next IPs” to protect.

Figure 9 shows the expected number of matching sequences from the generator as a function of received probes. The sequence count falls extremely fast from an initial astronomical number, after 11 scan packets a unique solution exists statistically.

This means that from the perspective of the defenders, it is actually not in their best interest to stop scans as early as possible, as this will inflate the number of targeted hosts to monitor. If the scanner is allowed to proceed with a limited number to packets – 11 for the network of 65000 IPs in our example –, an initially more reluctant mitigation will ultimately make the detection and defense much more focused already in the short term. We can thus conclude that receiving 9 or more scan probes – with the search for a solution already beginning

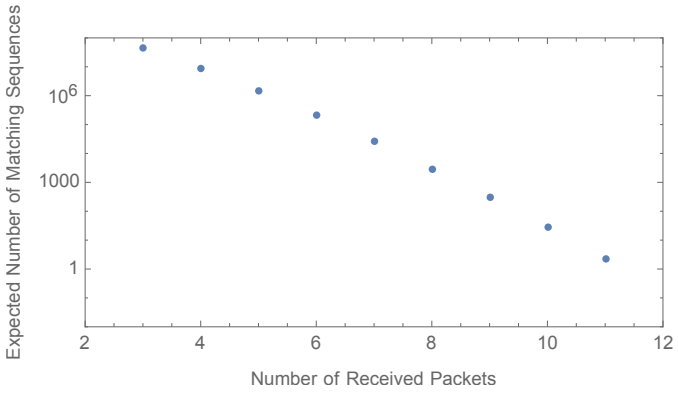

Fig. 9. Expected number of matching sequences as a function of received network probes.

TABLE I. WORST-CASE RUN-TIME FOR A FULL EXPLORATION ON A SINGLE CONSUMER GRAPHICS CARD

Model	Run-Time	cost / unit	speed-up
GTX 680	6h 20 min	95 Euro	–
GTX Titan	4h 39 min	350 Euro	25%
GTX 970	3h 38 min	500 Euro	42%

as soon as two packets come in – before responding to the scan is actually desirable.

B. Run Time

Since algorithmically-speaking only non-matching sequences – either due to out-of-order subsequences or threshold-exceeding packet loss – may be abandoned, as long as the seed is still a valid generator to the sequence it must be kept.

This however means that unless enough packets have been received to obtain a unique sequence as discussed above, the average run time will actually be constant, as valid sequences for storage may exists among any of the $2^{30}+62836735$. While exploration depth will differ per seed, long-term averaged over all seeds this variation disappears. From a defense perspective, this may actually be an attractive feature as the absence of variation implies a guaranteed time to find a solution.

Table I shows a comparison of the worst-case run times for the entire space of $2^{30} + 62836735$ seeds for three consumer graphics cards, the four-year old GTX 680, the three-year old GTX Titan and the one-year old GTX 970. The one-year more modern GTX Titan with its 2880 cores slices off another 25% of the run time of the GTX 680, the GTX 970 with its new Maxwell architecture reduces the run time by 42%. When running in a cluster configuration as described above, each individual card only needs to process a single-chunk, which means that a solution is obtained in a mere 7.4 minutes.

Although cluster size may be significantly reduced, if a solution may arrive slower or when relying on more modern architectures (or at equal size traded in with run time), older generations dominate the price-per-performance ratio. This is because those generations are being phased out and available at a significant discount. When factoring in total cost of ownership (TCO) with electricity and system administration, the modern chip however wins.

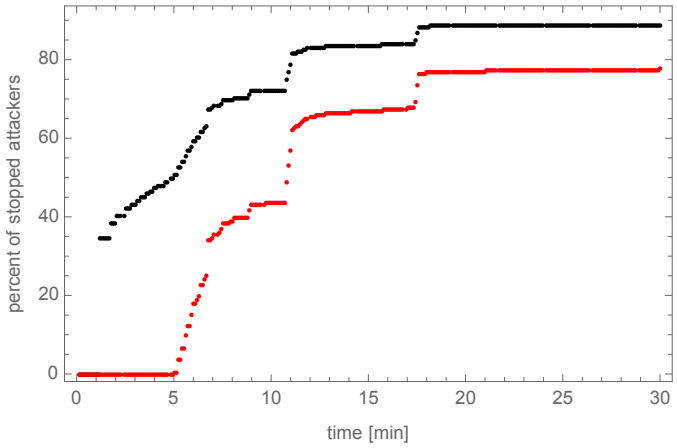

Fig. 10. Success percentage to detect scanners as a function of time.

C. Reaction Time and Mitigation Success for Scanning Speeds

Referring back to figure 4, it was shown that the bottom 50% in terms of scanning speed show an inter-arrival time of 90 seconds or more. At this threshold – we assume that faster and more noisy scanners can be efficiently detected by for example an IDS –, this means that within the 7.4 minutes in the worst case 5 probes have been received. For slower scans – say an inter-arrival time for a few minutes –, the cluster is actually "too" fast at computing the sequence than scan probes: while a set of possible solutions has been found and can be handed over to the scan mitigation block, the defending network actually has to wait for a couple of more packets to apply the right sequence in order to minimize the number of mitigation rules (see next section).

Figure 10 shows the percentage of stopped attacks as a function of time into the scan, the black curve for all scanners observed in our /16 network and the red curve for specifically those with a slow scanning speed. As can be seen in the figure, the method proves highly effective in stopping scanners in their tracks, after 11.5 minutes 50% of the seed and sequences of all slow scanners have been identified. The red curve trails off after approximately 18 minutes, as for the approx. 10% of scanners with an interarrival time longer than 20 minutes not enough packets have yet been received to do an effective search.

Although the focus for this prediction technique is to especially find slow scanners which are currently not caught in existing mitigation techniques, we do find that the technique is also applicable for those sending substantially more packets. The black curve demonstrates that within 5 minutes, more than 50% of the entire scanner population can be eliminated; while the worst case running time of the cluster would be 7.4 minutes, fast scanners have at this point already sent so many packets that most threads terminate significantly earlier and the single matching candidate can be immediately identified.

D. Parallelization of Search against Multiple Scanners

As discussed above, the use of the GPU's parallel SIMT structure excels when thread divergence and branching can be minimized as much as possible. Micro-benchmarks on the GTX 680 card showed that after an 8.6 second initial initialization period with the first chunk of seeds (which is avoided for subsequent blocks due to the asynchronous repopulation of the seed area), an individual thread displays a worst case run time of 560 ms, if all algorithmic optimizations discussed above are switched off. Within a run, approximately 22 ms are spent to compare the list of IP addresses at which probes have been received with the currently generated sequence.

As the sequence comparison only accounts for 3.9% of the overall worst-case exploration, another dimension of the seed search may thus be efficiently parallelized: the identification of multiple scanners. As an organization is most likely being subjected to a number of adversaries scanning at the same time, the overall cost to identify their scanning pattern is thus not a multiple of the overall runtime, but only comes at a slight performance penalty.

A concurrent identification of 10 adversarial patterns does require an additional 35% run time, resulting that the sequence can be computed on the cluster in just under 10 minutes. In an operational setting, batches of probes from various source IPs should thus be accumulated and processed in blocks in regular intervals to reduce the cost per identified sequence as much as possible.

VII. Mitigation

With the ability to predict the next targeted IP already during the scan, this opens up new mitigation options and possibilities for network defense. This section exemplarily introduces two such options.

A. Selective Firewall Adjustments

As most of an organization's IP addresses are actually assigned to hosts potentially continuously engaged in packet exchanges, a feasible mitigation option against an expected scan may not be to interrupt or interfere with the ongoing operation. However, as the amount of scanners is comparatively small – for our /16 network, approximately 13000 IP addresses performed a network reconnaissance over the course of a month – it is feasible to meet expected probes by selectively adjusting the firewall for the source IP addresses detected to be in a slow scan. This adjustment could either be targeting the entire network block, i.e., given a detected scan the source IP is blocked from any further interaction with the defender's network, or selectively – and potentially differently – for each expected scan target.

While the former approach of complete blockage would certainly seem the most effective mitigation option, note that the complete absence of any reply – including publicly available web, mail and DNS servers – would be noticed by the scanner. If the organization was specifically targeted, the scan could be resumed from another IP address. Given the location and value of assets in the defender's network, it might thus be more promising to only selective drop probes while admitting others. As some results are being returned, the scanner would typically assume that the silent blocks are non-existent, deterring activities away from more important assets.

B. Intension-Detection through Target Honeypots

As for a limited number of scanner IPs and targeted machines, individual firewall rules are feasible to implement. Traffic expected to arrive soon for the next IPs in line could be diverted to a honeypot. This honeypot would present open ports and accompanying services at those ports the scanner was previously interested in, and allow the defender to observe the next actions and possible intension of this actor selectively. Such a diversion, based on source and target IP address pairs, also ensures an impact-less operation of the proposed detection and mitigation schemes towards normal ongoing traffic.

VIII. Conclusion

In this paper we introduce a new type of network defense method and demonstrate that it is possible to detect and mitigate remote parties, performing a scan against the local network, using the scan tool ZMap. While an identification of the attacker's state would normally be a prohibitively expensive search, we introduce a number of algorithmic optimizations that together with an efficient implementation on commodity graphics cards provides a solution within minutes, fast enough to react against slow- and APT-style attackers that are not caught using today's rule- and threshold-based intrusion detection systems.

References

[1] N. Ianelli and A. Hackworth, "Botnets as a vehicle for online crime," *The International Journal of Forensic Computer Science*, vol. 1, 2007.

[2] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," in *Information Warfare and Security*, 2010.

[3] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide scanning and its security applications," in *Proceedings of the 22nd USENIX Security Symposium*, Aug. 2013.

[4] R. Graham, "Masscan: Mass ip port scanner, https://github.com/robertdavidgraham/masscan."

[5] V. Yegneswaran, P. Barford, and V. Paxson, "Using honeynets for internet situational awareness," in *HotNets*, 2005.

[6] M. Allman, V. Paxson, and J. Terrell, "A brief history of scanning," in *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, (New York, NY, USA), pp. 77–82, ACM, 2007.

[7] M. Javed and V. Paxson, "Detecting stealthy, distributed ssh brute-forcing," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), pp. 85–96, ACM, 2013.

[8] Z. Durumeric, M. Bailey, and J. A. Halderman, "An internet-wide view of internet-wide scanning," in *23rd USENIX Security Symposium*, 2014.

[9] Information Sciences Institute, "Transmission control protocol (rfc793)," tech. rep., IETF, 1981.

[10] N. Blenn, V. Ghiette, and C. Doerr, "Mining scan traffic for threat intentions," in *NCSC*, 2016.

[11] R. D. Graham, "Masscan: Mass ip port scanner," *URL: https://github.com/robertdavidgraham/masscan*, 2014.