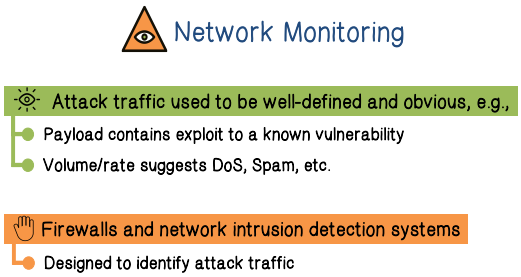
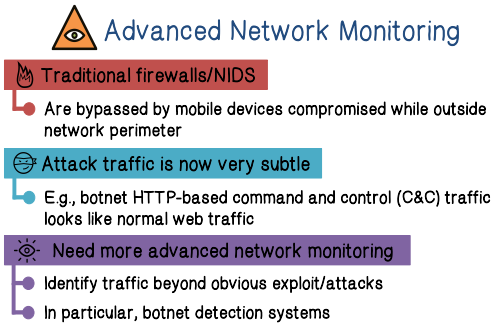
Recommended readings for this lesson:
BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee. In Proceedings of the 16th USENIX Security Symposium (Security'07), Boston, MA, August 2007.
http://faculty.cs.tamu.edu/guofei/paper/Gu_Security07_botHunter.pdf

You can find the summary of the paper at the end of the document.



The goal of network monitoring is to detect and prevent attack traffic. Attack traffic used to be obvious. For example, the payload of a packet may contain an exploit to a known vulnerability and therefore a signature can be used to detect such attack, or a network monitor can detect a deny-of-service attack or spam activity by analyzing the volume and rate of network traffic. The typical network monitoring systems are the firewalls and the network intrusion detection systems.



Increasingly, the traditional firewalls and network IDS are becoming less effective. First of all, mobile devices are now widely used. A mobile device can be compromised when an employee is on travel. And then when the employee brings the mobile device back into the company's network, it effectively has bypassed the perimeter defense. In addition, attack traffic now is very subtle and they often look like normal traffic. For example, a botnet's HTTP- based command-and-control traffic would look like normal legitimate web traffic. Therefore, we need more advanced network monitoring systems to detect this new generation of attacks. In this lesson, we are going to discuss botnet detection systems.

Fill in the blanks with the correct answers.

A Bot is often called a zombie because it is a compromised computer controlled by malware without the consent and knowledge of the user.

Here is a quiz on the definition of a BOT.

A bot is often called a zombie because it is a compromised computer controlled by malware without the consent and knowledge of the user.

Fill in the blanks with the correct answers.

A Botnet is a network of bots controlled by a Bot Master (*attacker*). More precisely, a coordinated group of malware instances that are controlled via command and control (C&C) channels. C&C architectures: centralized (e.g., IRC, HTTP), distributed (e.g., P2P)

It is a key platform for fraud and other for-profit exploits.

Here is a quiz on Botnet.

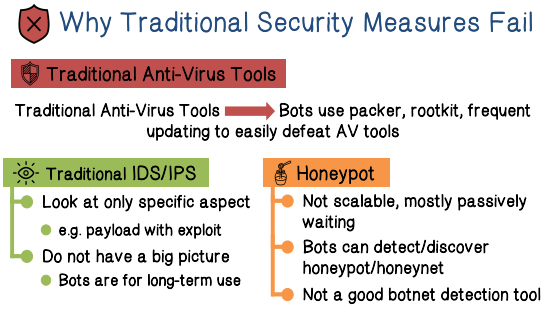
A Botnet is a network of bots controlled by a bot master or an attacker. More precisely, a botnet is a coordinated group of malware instances that are controlled via command-and-control (C&C) channels. C&C can use centralized architectures or distributed architectures. Botnet is a key platform for fraud and other for-profit exploits.

Select all the tasks that botnets commonly perform:

- [x] More than 95% of all spam
- [x] All distributed denial of service (DDoS) attacks
- [x] Click fraud
- [x] Phishing & pharming attacks
- [x] Key logging & data/identity theft
- [x] Distributing other malware, e.g., spyware
- [x] Anonymized terrorist & criminal communication

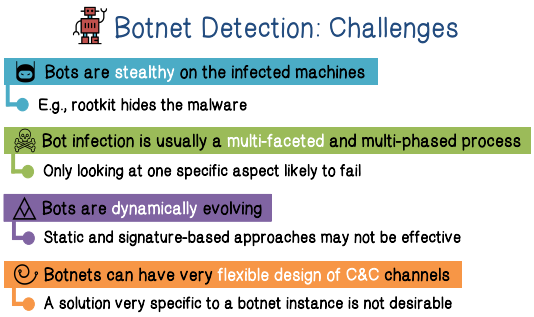
Here is a quiz on botnet activities. Select all the activities that botnet commonly perform.

All of these tasks are commonly performed by botnets. Other than spam and DDoS, these other attacks can look a lot like normal traffic.

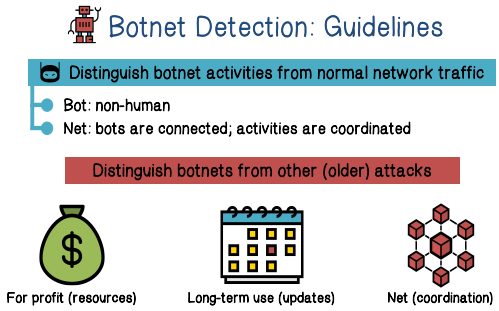


Let us analyze why the traditional security measures cannot detect botnets effectively. First of all, traditional signature-based antivirus systems are not effective, because bot codes are typically packed, and they can use rootkit to hide. And they also use frequent updates to defeat antivirus tools. The traditional IDS and IPS are not effective, because they typically look at the specific aspect of an attack, for example, a specific exploit method. Whereas, botnet typically performs multiple kinds of activities, because they are for long-term

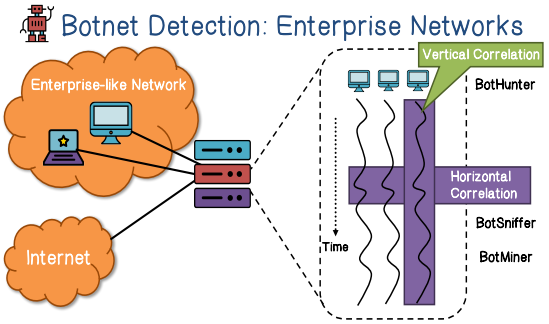
use. That is, although we can detect that a host has been compromised by an exploit, we do not know that it belongs to a botnet because we would need to analyze its command-and-control traffic and daily malicious activities. Honeypots and Honeynets are also not effective. First of all, since they only passively waiting for incoming connections, they have to be lucky to capture botnet activities. In addition, sophisticated bot malware can detect a honeypot because of the lack of realistic user activities. And since a Honeypot is a single host it cannot detect a network of bots.



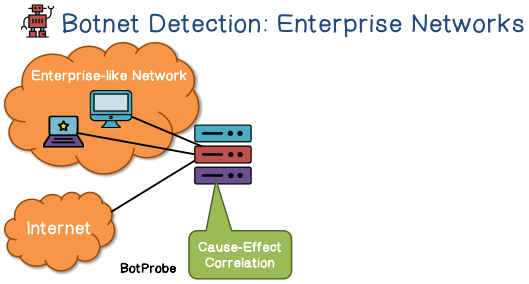
What are the challenges in botnet detection? First of all, bots try to hide themselves. Second of all, bots are also involved in multiple activities over a period of time. Bot malware can also get updates frequently. Botnets can have many different command control methods. In fact, a bot malware can be programmed to select one of several C&C methods at run time.



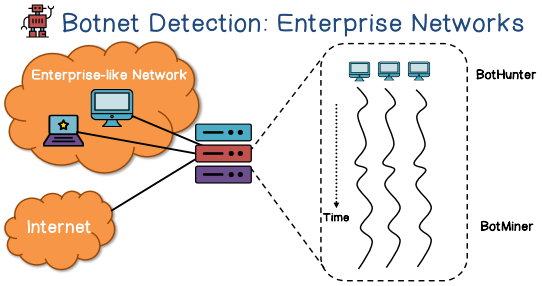
So, how do we go about detecting botnets? We need to first focus on the characteristics that botnets are different from normal traffic. First of all, a bot is not a human. That is, the activities by bots may look different from the activities by human. Second of all, the fact that botnet is a network means that the bots are connected, and their activities are somehow coordinated. We can also distinguish botnets from other traditional attacks. Botnets are for profits, and they most likely are going to use the compromised computers as resources. Botnets are for long-term use, and therefore, there will be frequent updates to the bot malware. And again, there must be coordination among the bots to form a botnet.



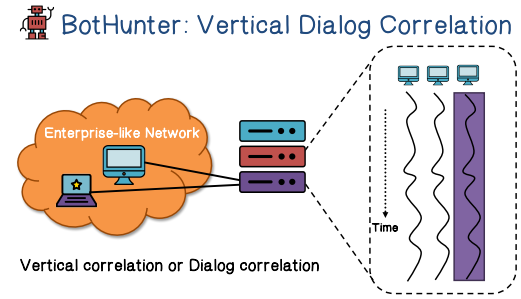
Let us first discuss how we detect botnets in the enterprise network. We can deploy a botnet detection system at a gateway or router. This is how we deploy firewall in IDS. There are several detection approaches in a so-called vertical correlation. We are looking for correlated events across a time horizon, even if a bot has multiple activities in its life cycle. In horizontal correlation, we are looking for similar or coordinated behaviors across multiple bots.



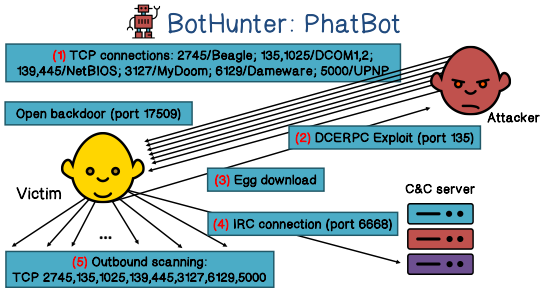
In cause-and-effect correlation, we inject traffic to play with the bot to conform that the traffic is generated by bot versus human.



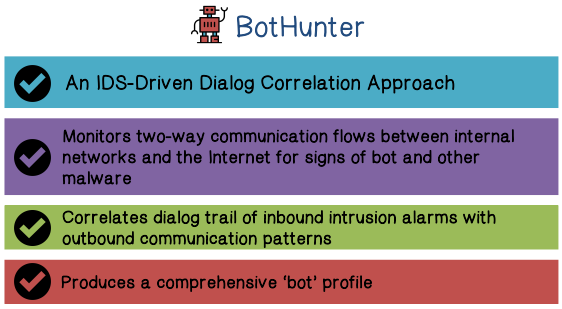
In this lesson, we aregoing to discuss two systems, one is BotHunter, the other one is BotMiner.



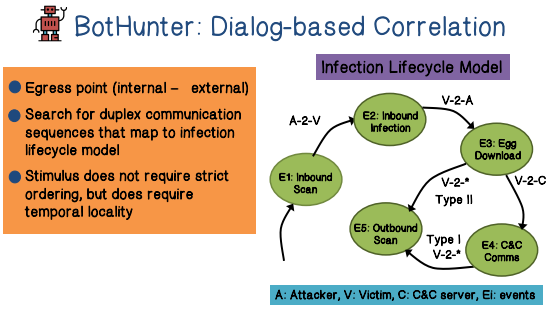
BotHunter is a system that performs vertical correlation. We also call it dialog correlation. That is, BotHunter correlates multiple events that belong to the life cycle of a bot.



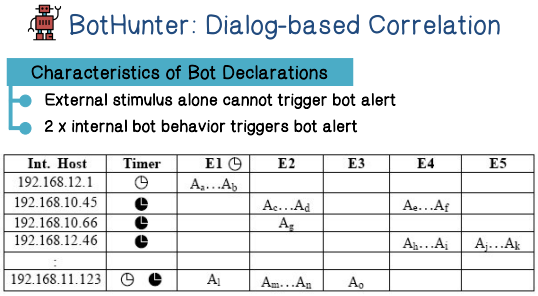
Let us use an example to illustrate the multiple steps or types of activities in a bot's life cycle. First, the attacker scans the network and identifies vulnerable hosts. It then sends the exploit to compromise a victim host and opens a back door. The compromised computer then downloads the real bot malware, it then connects to a command-and-control server. And from there, it can perform a number of activities, for example, port scanning, to find other vulnerable hosts.



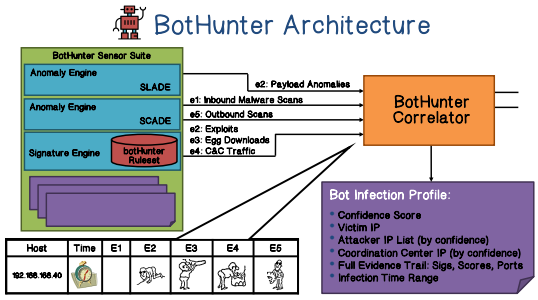
In short, BotHunter is based on IDS technologies, and it uses a table-driven correlation algorithm to correlate evidences to decide if a host is compromised as a bot. The evidences are obtained by monitoring traffic and identifying botnet related command-and-control and malware related activities. For each bot, BotHunter outputs a description of its behavior profile.



From the phatbot example, we can extract the life cycle of a botnet. You can include inbound scan and inbound infection, and then outbound traffic to download the egg or the bot malware and command-and-control traffic, as well as other activities, such as outbound scan. So, the main idea behind BotHunter is to analyze network traffic to detect patterns that suggest any of these activities belonging to the botnet lifecycle. These observations do not have to follow this strict order, but they do have to appear within the same period of time.

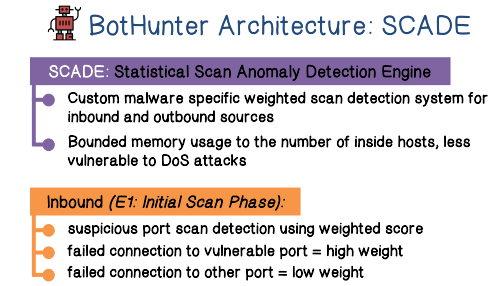


BotHunter uses a table to keep track of the evidence that it collects for each host. And here is an example: for each internal host, BotHunter keeps track of the specific activities that belong to each steps of the botnet life cycle. There are timers associated with these observations. That is, they are valid as long as a timer has not expired. The intuition is that within a period of time that is before the timer expired, if you see multiple evidences belonging to the botnet life cycle then we can determine that this host is a bot. And we give more weight to evidences that suggest that an internal machine has been compromised and it is participating in botnet activities such as egg downloading, outbound scanning and outbound spamming.

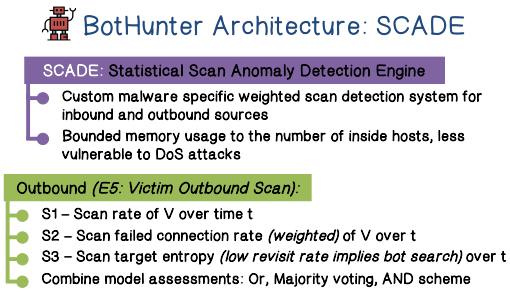


Here is the architecture of BotHunter. It has a number of detection engines. Each of these engines are disposable for detecting certain activities of the botnet lifecycle. And the correlator correlates evidences of these activities and makes detection that an internal machine

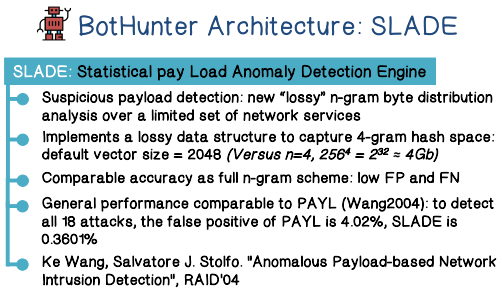
has been compromised and has become a bot and produces a bot infection profile.



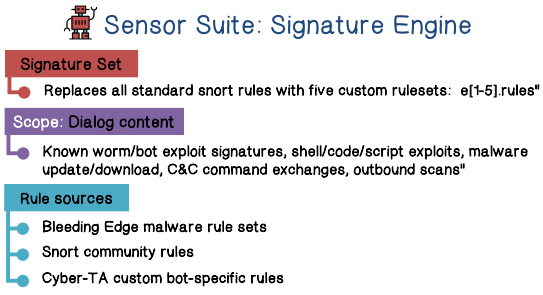
Let us discuss the BotHunter detection engines. The first is SCADE. It is for scan detection. Recall that in botnet lifecycle inbound scan is a first event. SCADE uses different ways for different inbound scan connections, in particular, it gives a higher weight to vulnerable ports.



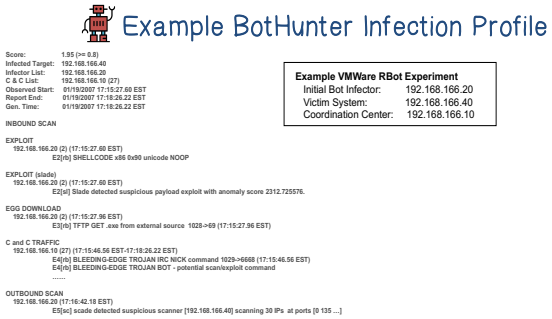
SCADE also detects outbound scan. It looks at the rate of outbound connections, how likely such connection fails and the distribution of the destinations of these outbound connections. Collectively, this can suggest outbound scan.



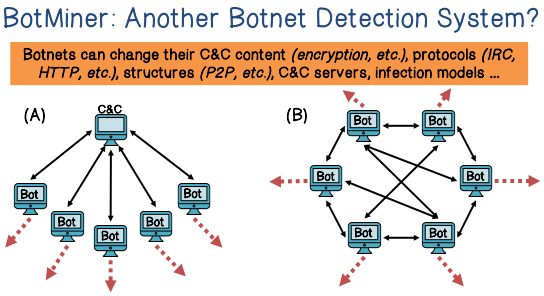
Another BotHunter detection engine is SLADE. It can detect anomalies in network payloads. The main idea is that we can establish the normal profile of a network service by looking at the n-gram byte distribution of the traffic payload of this network service. That is, an attack such as an exploit or egg download will cause deviation from this normal profile because the n-gram byte distribution of the attack traffic will be different from the normal traffic. SLADE has a very efficient implementation of payload anomaly detection.



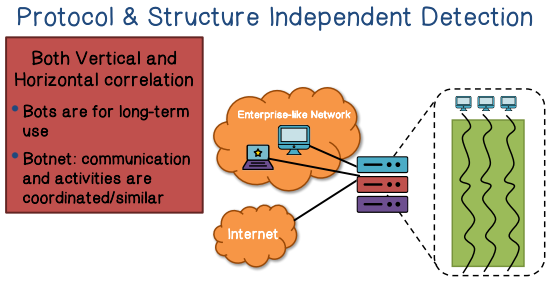
BotHunter also includes a signature engine. It can detect known exploits and known patterns of command control. The signature rules come from multiple open sources.



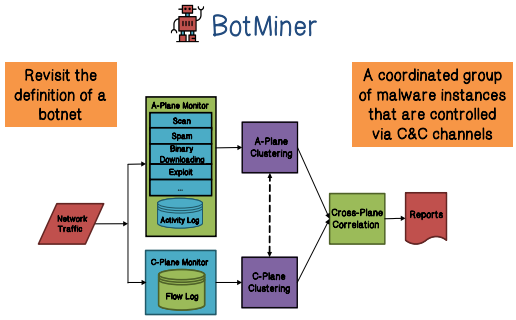
Here is an example of infection profile produced by BotHunter. It lists the initial exploit that infects the internal machine, that is, the internal machine that has become a bot, and the command-and-control server. It also lists evidences of the lifecycle steps.



Now let us discuss another botnet detection system, Botminer. The first question is, why do we need another botnet detection system. Notice that BotHunter is based on some specific botnet infection life cycles. But botnet can have different infection life cycles and they can change the protocols and structures of the command-and-control. For example, botnet can use a centralized command-and-control system or fully distributed peer-to-peer control system.

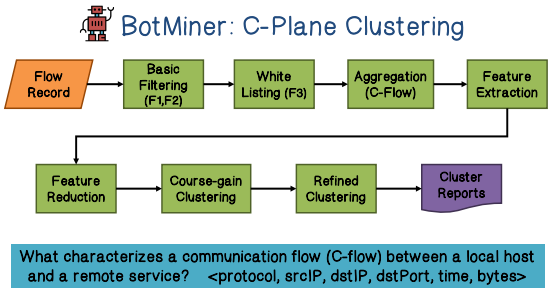


Our goal is to have a Botnet detection system that is independent of the command-and-control protocol and structure. In order to achieve this goal, we need to focus on the intrinsic properties of botnet. In particular, bots are for long-term use, and bots within the same botnet have similar or coordinated communication and activities. Therefore, we need to perform both vertical and horizontal correlations.

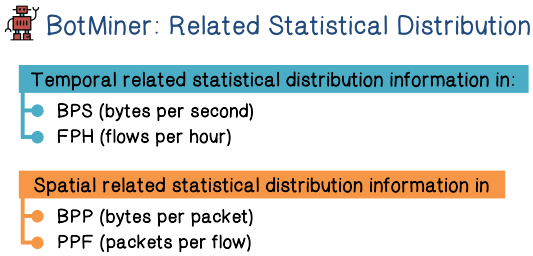


Here is the architecture of BotMiner. We arrive at this architecture based on the definition of a botnet. We say that a botnet is a coordinated group of malware instances, and they are controlled via command-and-control channels. The C-Plane monitor here is for monitoring command-and-control traffic and A-Plane here is for monitoring malicious activities because these are malware instances. On both planes, we perform clustering to detect groups that are in correlated or similar ways. Then we use

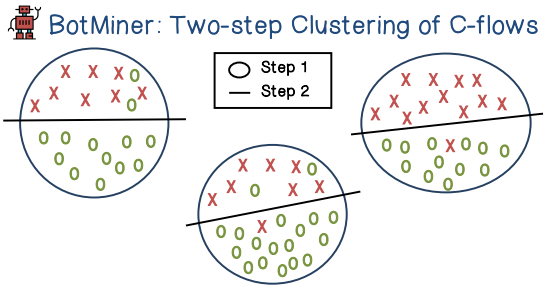
cross-plane correlation to detect a group of machines that perform similarly in both command control activities and malicious activities. And these are parts of the same botnet.



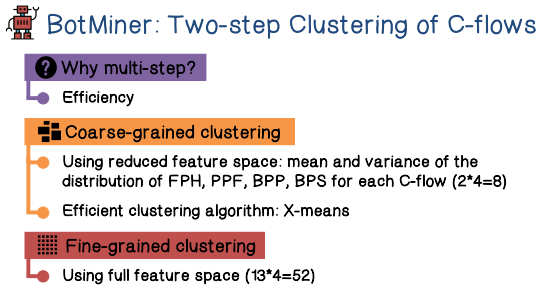
Here is a workflow of C-plane clustering. First of all, a flow record for connection between a local host and a remote service is defined by the protocol, source and destination IP addresses and destination port, time, and number of bytes. All flow records go through a number of steps that include filtering, aggregation, feature extraction, and clustering.



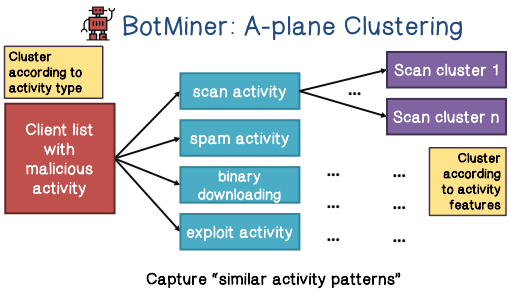
Here are some examples of our features, look at bytes per second, flows per hour, bytes per packet, and packets per flow and we perform clustering in two steps.



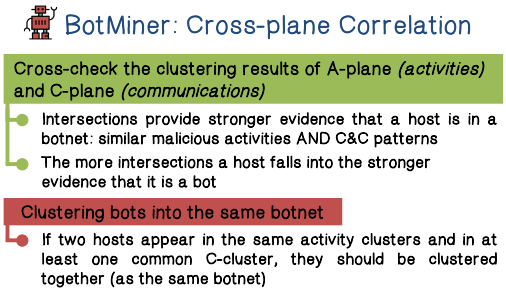
In step one, we group C-flow records into course-grained clusters. In step two, within each of these course-grained clusters, we further position them into finer grain groups.



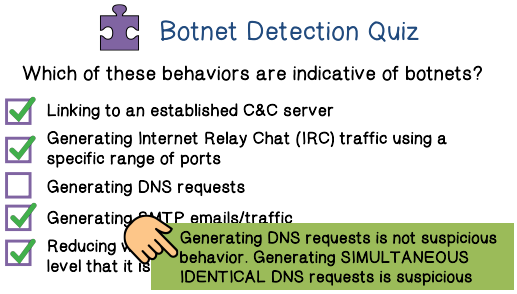
The main idea here is that we can use a small set of features to perform course-grain clustering. Because the number of features that we use is small, this step is very efficient. There within each course-grained cluster, you can afford to use the full feature space to perform fine-grained clustering.



In A-plane clustering, we first cluster based on activity type, for example, this is scan, spam, binary downloading or exploit. Within each activity we use the traffic features to perform further clustering. For example, for scan, we can use the destination subnets and the ports and for spam we can use a spam template. Again, the main idea of A-plane clustering is to capture similar activity patterns among the hosts.

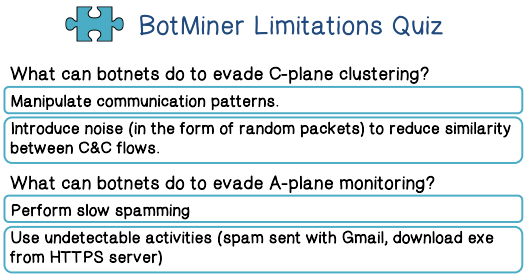


In cross-plane correlation, we are looking for the intersection between A-plane and C-plane clusters. Intuitively, hosts in the intersections have similar malicious activities and similar C&C patterns. In particular, if two machines appear in the same activity clusters and in at least in one common C-cluster. That means they should be clustered together because they are in the same botnet.

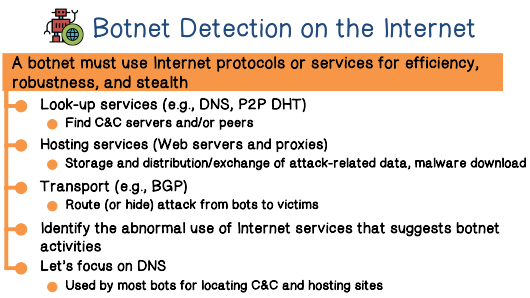


Now, let us do a quiz on botnet detection. Which of these behaviors are indicative of botnets?

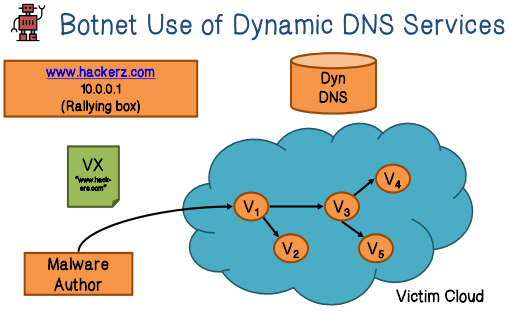
Generating DNS requests by itself is not indicative of Botnet activities. However, if multiple machines looking up the same domains at the same time and the domain is not on a whitelist, that is quite suspicious.



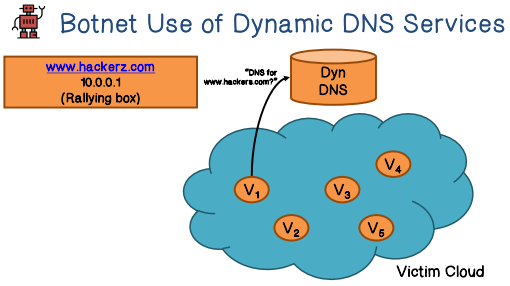
Now, let us consider the limitations of BotMiner. What can botnets do to evade the C-plane clustering? And, what can a botnet do to evade A-plane clustering? Botnets can vary the C&C communication patterns and they can introduce random noise in their communication traffic. For A-plane clustering, the botnets can also vary their activities to evade our detection heuristics.



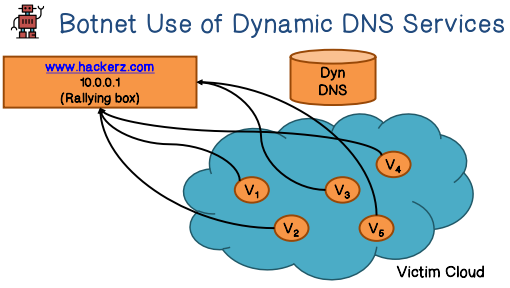
So far, we have discussed botnet detection in the enterprise network. Now, let's discuss how we detect botnets on the Internet. We observe that a botnet must use Internet protocols and services in order to maintain a network infrastructure. For example, in order to maintain its network structure, it must use some sort of look-up services to find command-and-control servers or the peers. And you know that to support its various kinds of malicious activities, a botnet must use hosting services, for example, to store and distribute attack data and for malware download. A botnet can also use transport services to route or hide its attack traffic. Therefore, by identifying the abnormal use of internet services, we can detect botnet activities on the internet. In this lesson, let us focus on DNS. And the reason is that most bots use DNS to locate command-and-control and hosting services.



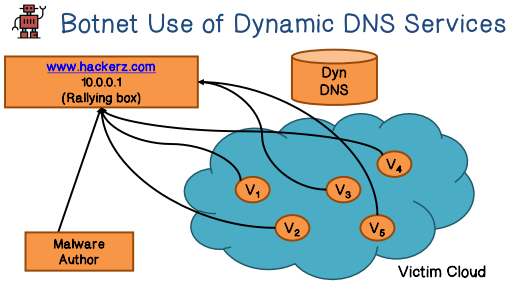
Many botnets use DNS for command control. A key advantage is that DNS is used whenever a machine on the internet needs to talk to another, because DNS stores the mapping between domain name and IP address. That is, DNS is always allowed in a network and using DNS for command control will not standout easily. Suppose a malware infects many machines, that is, many machines now have become bots. The question is how can these bots be organized into a botnet?



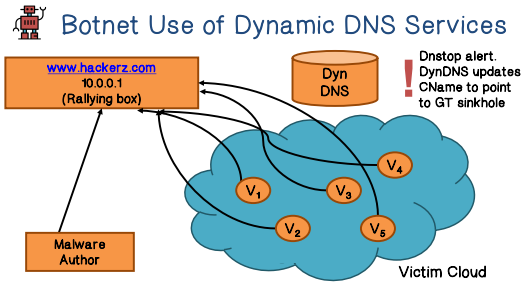
The bot malware has instructions to connect to its command-and-control server. But in order to connect to the command-and-control server, it will perform a DNS lookup first.



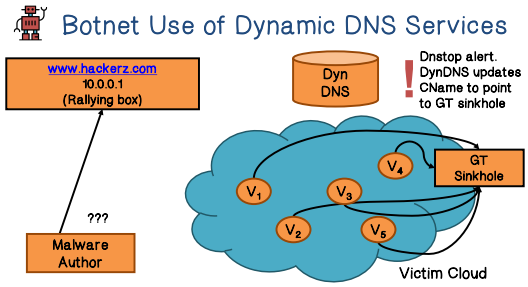
And with the IP address, the bot can connect to the command-and-control server and that is how the bot becomes part of a botnet.



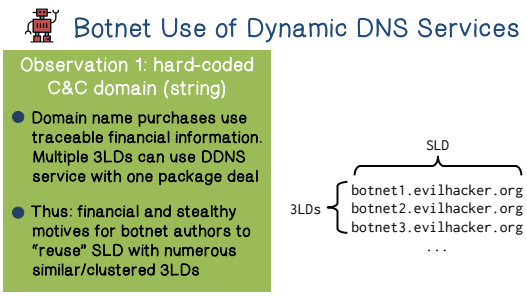
The DNS service providers preferred by botnets are Dynamic DNS providers, because they allow the frequent changes of the mapping between DNS domain name and IP address. That is, the botmaster can change to use another machine on the Internet for command-and-control and all he needs to do is log into his Dynamic DNS provider, and make changes.



If we can detect that a domain is used for botnet command-and-control, then we can detect which machine connects to this domain and this machine is a bot. But how do we know this domain is used for botnet command-and-control? It turns out that the way the bots look up the domain is different from how machines look up Internet domains such as a news website because of normal user activities. For example, a botnet C&C is looked up by hundreds of thousands of machines across the Internet. And yet, it is so-called unknown according to Google search and that is an anomaly. We can use anomaly detection at the Dynamic DNS service provider by examine queries in DNS domains to identify botnet CNC domains.

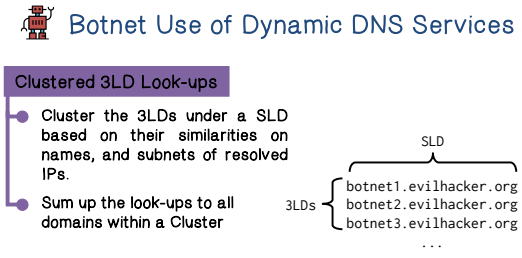


And once we identify a domain is used for botnet C&C, then a number of responses are available. One is for the service provider to disable a domain, and so when a bot looks up the domain it gets no IP address back. Another option is for the provider to set a mapping of a domain to a single address. So that instead of connecting to the botnet command-and-control server, the bots are now connected to a sinkhole. The sinkhole, in addition to disabling the botnet, allows security researchers to learn where the bots are by looking at the origins of the connections to the sinkhole.

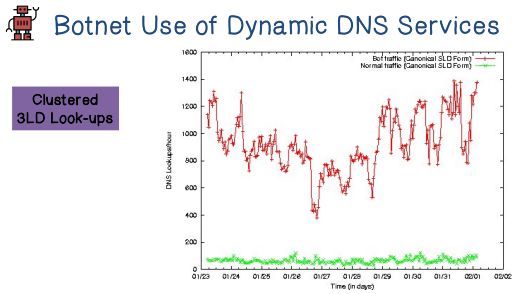


Now, let us focus on how we detect that a domain is used for C&C. There are a number of heuristic-based on observations. First, when a bot master mass purchases a domain, he must use a credit card, and that leaves traceable financial information. And in order to limit such traceable information as little as possible. Therefore, he may do the so-called package deal, where for one

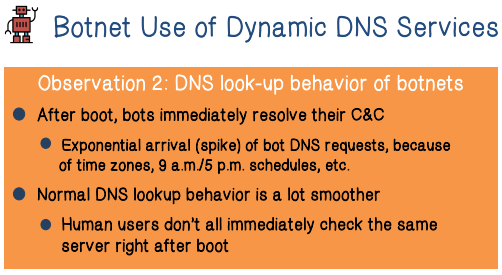
second level domain, he is going to use it for multiple three-level domains for botnets. For example, here for the same package deal for the second-level domain evilhacker.org, there are multiple botnets using different three-level domains, the point is that a single financial transaction can support multiple botnets.



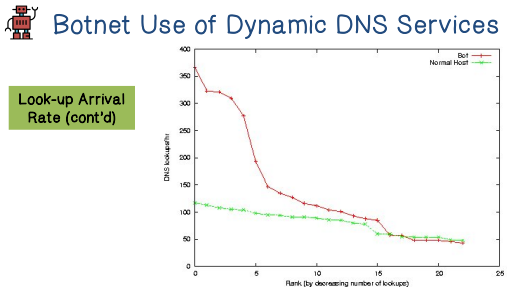
Therefore, we can cluster the three-level domains under the same second-level domain that looks similar in their names or they resolved to similar subnets of IPs because they are likely to be botnets that are related. And we sum up the lookups to all of these domains within a cluster.



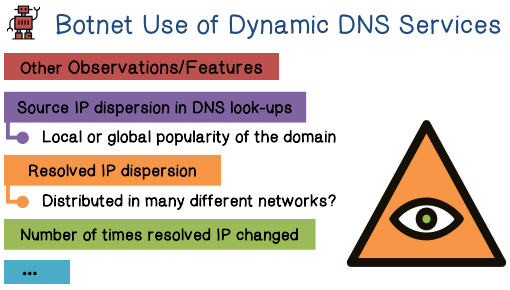
By doing so we will compare the lookup patterns of legitimate domains. They are different from the lookup patterns of the botnet domains because the later tend to have larger lookup volumes. And remember these domains are in dynamic DNS providers and the domains tend to be small and medium sized businesses, therefore, legitimate domains tend not to have a very large lookup volume.



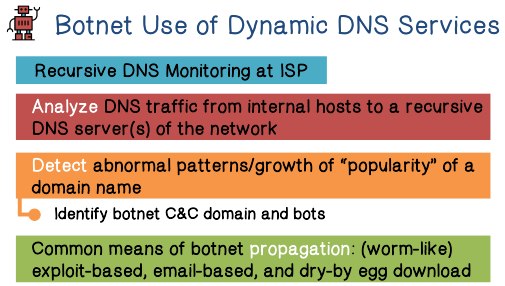
Another observation is that bots tend to look up their command-and-control service as soon as their host machines are connected to the Internet. The intuition is that bots must maintain communication with the command-and-control server and since they have no control over when the host machines connect to the internet, they must take their first opportunity. The result is that there is an exponential arrival of DNS requests from the bots, whereas legitimate DNS lookups by normal user activities is a lot smoother. For example, Human uses do not all immediately check the same side.



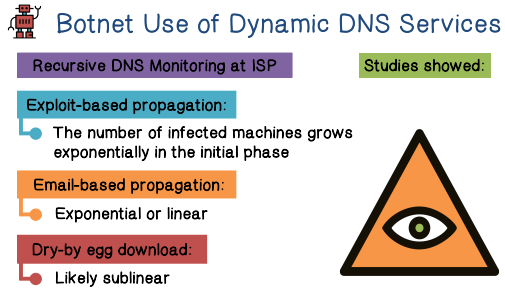
Therefore, if you sort the lookout volume per hour by 24 hour windows, you can see the exponential arrival rate of the bot's queries, whereas human queries are a lot smoother.



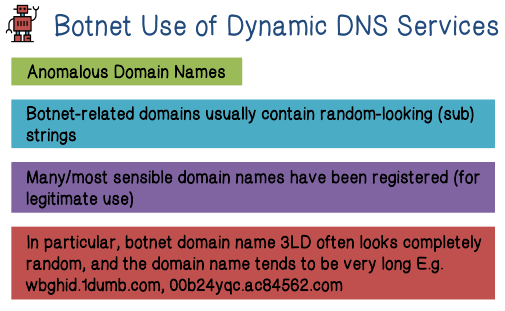
There are other detection heuristics. For example, the lookups are from all over the Internet. And a C&C server is resolved to many different IP addresses across the Internet. And the resolved IP address changes frequently and so on. Any of these observations alone cannot effectively detect a botnet command-and-control domain. And therefore, we combine them in a detection system.



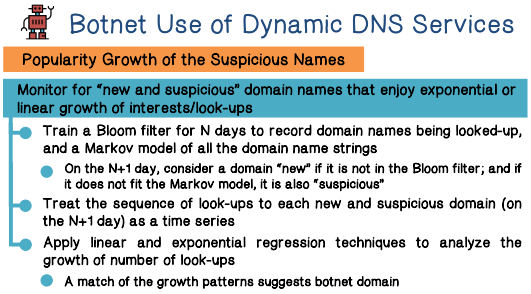
Now let us discuss how we detect botnets in large networks such an Internet Service Provider (ISP). Here, we can focus on the DNS queries that can be observed by the recursive DNS server and the ISP. That is, we can analyze the Internet traffic from the internal host to the DNS server. In particular, we can detect any abnormal growth of the popularity of a domain name. It can suggest that this domain name is used for botnet command-and-control. Intuitively, the reason that a botnet will grow is that more machines become infected and become bots. Therefore, the growth pattern of a botnet corresponds to the propagation pattern of an infection.



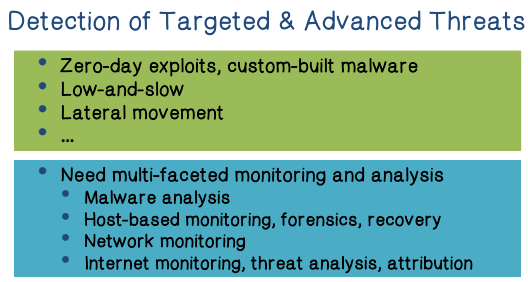
According to studies, exploit-based infection grows exponentially. Email-based infection grows exponentially or linearly. And drive-by download's growth is likely to be sublinear.



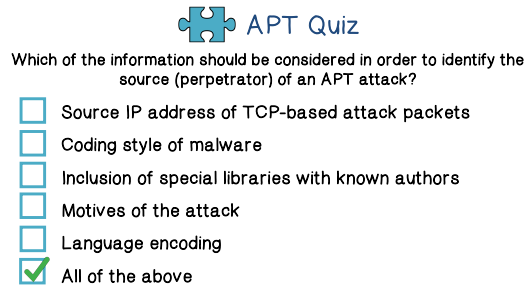
In a large ISP, there are many domain names being looked up every day. But we do not have to analyze all domain names. In fact, we can focus on a few anomalous domain names. These are domain names that look suspicious. In particular, many regularly spelled, easily sounding domain names have been taken up by businesses and individuals already. Therefore, botnets are forced to use very random looking domain names. In other words, these random looking domain names are suspicious.



And then we need to analyze the growth pattern of these suspicious domain names. In particular, we look for the exponential or linear growth of their popularities. Here is a sketch of our ideas. We assume a baseline of N days, and we assume that, within these N days, all domains are legitimate. Therefore, we record these domains In a Bloom filter. A Bloom filter is a very efficient representation of set. That is, we use the Bloom filter to record a set of domains observed within these N days. We also use a Markov model to model how these domain names are spelled. After this baseline, whenever we observe a new domain, that is, a domain that is not in our Bloom filter on a set of recorded domains, if this domain does not fit our Markov model, that means it is spelled in a different way and looks suspicious. Then we know that we have a new and suspicious domain. Then we analyze the lookup sequence to this new and suspicious domain. And if the growth is linear or exponential, then we know that this domain is used for botnets.



So far, we have discussed botnet detection. The latest threats are more targeted and more advanced. For example, they use custom-built malware on zero-day exploits, their activities are low-and-slow, and they move within network and covering their tracks. Even the existing botnet detection systems are not effective against these targeted advanced threats. In order to counteract these targeted and advanced threats, we need multifaceted monitoring and analysis. That is, we need malware analysis, host-based monitoring, forensics, and recovery, network monitoring, Internet monitoring, threat analysis, and attribution.



Now let us do a quiz on APT. Which of the following information should we consider in order to identify the source of an APT attack?

We need all of them.

BotHunter: Detecting Malware Infection through IDS-Driven Dialog Correlation

BotHunter monitors the two-way communication flows between hosts within your internal network and the Internet. It aggressively classifies data exchanges that cross your network boundary as potential dialog steps in the life cycle of an ongoing malware infection. BotHunter employs Snort as a dialog event generation engine, and Snort is heavily modified and customized to conduct this dialog classification process. Dialog events are then fed directly into a separate dialog correlation engine, where BotHunter maps each host's dialog production patterns against an abstract malware infection lifecycle model. When enough evidence is acquired to declare a host infected, BotHunter produces an infection profile to summarize all evidence it has gathered regarding the infection. In short, BotHunter helps you rapidly identify infected machines inside your network that are clearly and helplessly under the control of external malicious hackers.

Dialog correlation is an algorithm that classifies each network traffic exchange that occurs across your network boundary as potential infection sequence steps that occur when a host is infected with malware. Not all network traffic events generate a dialog event. Dialog events are fed directly into a separate dialog correlation engine, where each host's individual dialog production pattern is mapped and scored against an abstract malware infection life cycle model, including following potential dialog transactions;

- E1: External to Internal Inbound Scan
- E2: External to Internal Inbound Exploit
- E3: Internal to External Binary Acquisition
- E4: Internal to External C&C Communication
- E5: Internal to External Outbound Infection Scanning

When the dialog correlation algorithm determines that a host's dialog production patterns maps sufficiently close to the life cycle mode, the host is declared infected, and an infection profile is generated to summarize all evidence regarding the infection.

Overview

Distinguishing a successful malware infection from the vantage point of the network egress position requires a command of the two-way dialog flow that occurs between a network's internal hosts and the Internet. While many malware infections start with an initial external-to-

internal infection, malware may use a wide range of options to infect a host, including indirect host infections through email, direct exploit-based infection, or drive-by infections that are launched from malicious network servers. Furthermore, with the growth in popularity and capability of mobile laptops, direct infection of an internal asset needs not necessarily take place behind a well-administered network. Malware may inject itself into a host opportunistically from any Internet access point the hosts happens to associate or may be executed voluntarily by a victim who inadvertently accesses a Trojan binary, multimedia file, or other infected transmission source. Regardless of how malware enters a host, once established inside the network perimeter the challenge remains to identify the infected machine and remove it from service as quickly as possible. Refer Section 4 for system design in detail and Section 5 for evaluation results.