Concordia University
Ivan Pustogarov
March 14, 2022

Lecture 8

DoS attacks (cont'd), botnets — Coremelt, Crossfire, Spamhaus

Part 1

Slide 1. Hello everyone, today we will continue speaking about Denial of service attacks, and also speak about botnets. In the previous lectures we were speaking about cases when a denial of service attacks directly targets a specific server on the Internet. In this kind of attack you either send a lot of traffic to congest the server bandwidth or you send a lot of connection requests to exhaust server's memory or CPU.

Today we will consider a bit different way to take down a server or a group servers. Intend of directly targeting the endpoint, an attacker can try congest network links that surround the server. In this case the server will not really experience high load but at the same time clients that want to connect tot the server will not be able to do that because the traffic will be dropped at the congested links.

Slide 2. This type of attacks were known before, but one of the first academic papers describing this kind of attacks is coremelt paper. This attack does not target specific server, but tries to disconnect to parts of the network by congesting specific links between autonomous sets. Autonomous sets are roughly can be described as

This attack assumes that the attacker controls a botnet that are able to send traffic to each other. Since there is not address spoofing and each of the bots sends moderate amount of traffic this type of attack are much harder to detect because all the traffic looks like legitimate traffic. If you have N bots in your botnets then you can create in the order N^2.

Slide 3. The attack itself looks as follows. First, the attacker selects some link as a target link. This can be for example a links between some area and the rest of the Internet. Next the attacker needs to identify what bots can send traffic between each other so that this traffic goes through this link. For example, on the slide, nodes S1 and S2 belong to two subnetworks for which the path between them does not include the target link. This means that S1 and S2 there is no point in sending traffic between them. On the other hand, the path between S1 and S3 includes the target link, this means that the data send between these two bots will contribute to congesting the target link.

The efficiency of this attack would depend on two factors. First, the number of bots and their distribution. The more bots you have, and the more they are distributed, so that there are many pairs than can send traffic over the target link, the higher chance you have to congest the target link. Second, the success of the attack depends on how much data the bots can send. If the bots are able to send more data the you will need less bots to saturate the target link and other way round. At the time when this attack was documented, with about 400,000 geographically distributed bots that are able to send data at the speed of 128 kbps, it would be possible to saturate links between deferent autonomous sets.

One important thing about this attack is collateral damage. When your bots send data, the also put other link in the network under heavy load. This means two things. First, the data send by some of a subset the bots will be limited by those previous network links. And seconds, the more intermediate links you saturate the more noticeable you attack will be.

One of the goals of this attack is to be as stealthy as possible, this is why you bots mimic normal users.

Slide 4. Once we know that we can DoS network links let's see how we can further extend it to actually target server and geographical areas, specifically, let's look at crossfire attack. This attack has many similar properties. It also uses a large botnet where each bot send a low-volume traffic with the goal o congest specific links. There is one important difference. Instead of sending traffic between each other, bots choose public servers for example HTTPS servers to send send and receive data from. These server are chosen in such a way so that the traffic goes through the target links.

Slide 5. Let's see in a bot more detail how this attack works. First what is the target for this detail of service attack? The target is either server or some institution, such as a university, and government department. Or it can be some geographical area such as a city district, or a even a city. On the slide the target are is in he middle, inside the read circle.

In order to disable this target area, the attacker will try to congest all the network links that connect this target are with the rest of the Internet. In the slide these are marked as read lines. As before we call them target links. Now we assume that the attacker controls a number of geographically distributed bots that would need to send traffic over these links. In order to do this, the attacker needs to identify a number of servers that are geographically close to the target area. This servers are called “decoy servers” in crossfire attack. If the bots then mimic normal users and send traffic to the decoy servers, then some of the connection will go through the target links.

Slide 6. Let’s look at each step at a bit more detail. The attack consists of the three main steps. During the first step the attacker tire to learn the network topology that connects his bots to the larger area. Or more specific the attacks tries to identify IP paths that connect his bots to the servers is the target area. In order to do this each bot can run a “traceroute” command. Just to remind you, traceroute command allows you see the IP address of routers that are between your machine and some server. [SHOW TRACEROTUE]

The way traceroute works is quite simple. When you send a packet to a some server, this packet travels from one router to another until it reaches the destination. Each IP packet contains a field called “time to live” or TTL that is set to some value. Whenever a packet is forward to the next hop, the corresponding router subtracts one from TTL. When the TTL value reaches zeroth router sends back an error message. So by setting TTL to 1, the first router in the path will send us the error message. Then we set TTL to 2 and resend the packet. In this way, the second router in the path will send us the error message. In this way we find the path to a server hop by hop.

One important thing here is that network paths may change over time. Moslty due to load balancing. Because of this the bots might want to run traceroute command multiple time to eliminate non-persistent paths. After this step, the attacker will learn what persistent paths connect his bots to the servers in the target area.

Slide 7. At the next step, the attacker needs to find a set of links that, if congested, would disconnect the target are from the rest of the Internet. The rule here is that you would want to have links with higher bandwidths, as supposedly most of the clients will try to connect the target are using these high-bandwitdth links.

At this step you also need to choose decoy servers. These are servers that are geographically close to the target area. And the target links should be in the paths between bots and the decoy servers. For example, assume you target a government department in some city. In the same city you probably will have universities that run a number of web-servers that can be used as decoy servers.

Slide 8. Finally you lunch the actual attack by instructing the bots to send requests to the decoy servers. Based on each target link capacity you will assign the amount of traffic sent by each bot. If you properly flood all the target links, they will become congested. As the same links are used to connect Internet users to the servers in the target area, these servers will be either become unreachable to the latency will be become too large to be usable.

In order to make this attack stealthier or to deal with automatic load balancing for those links, you might want to choose a different disjoint set of target links and alternate between flooding the first set and the second set of target links.

Slide 9. This attack has several important properties.

First since the attack is not attacking the target are directly, it is not possible to detect this attack on them: the servers will not experience higher load. On the contrary, the load will be lower.

Second, the traffic flows generated in this attack are very hard to distinguish from legitimate Internet traffic. Because each bot has a different non-spoofed IP address, every connection looks different for the router and it's hard to apply defence mechanisms.

Third, this attack can be made persistent by changing bots and decoy servers.

And finally, the availability of decoy servers makes this attack applicable to different target areas.

Slide 10. Let's now see a historical example of this type of attack where the attack targets network infrastructure instead of specific server.

The attack initially Spamhaus service in 2013. In simple words Spamhouse is an Internet company that builds and manages a list of domains names and IP address that are involved in sending spam. Other companies then can used this list to filter incoming traffic and reduce the amount of spam and malicious emails. Apparently one hosting provider was included into this list and was not happy about it and decided to initiate DDoS attack on Spamhaus.

Slide 11. The attack turned out to be very powerful and gradually attacked different parts of the network when it could not take down Spamhuaus itself. And because of this and huge volume of traffic it affected many unrelated services.

Slide 12. The attackers' goal was to flood Spamhaus to overwhelm their capacity to handle incoming network packets. The attack used a large botnet and was a DNS amplification attack [Do you remember what is DNS amplification attack?]. At the beginning the attack targeted directly Spamhaus which became unreachable. But shortly after it hired a Cloudflare which was able to redistribute the attack load over a large number of servers.

The attack stoped for a bit, but then the attackers changed their strategy and attacked Cloudflare infrastructure. In general in hard to attack CND providers as they have a lot of resources to redistribute the traffic. So the attackers decided to attack network infrastructure itself. Specifically they London Internet exchange. Internet exchanges a large internet hubs that connect different networks. And specifically in connected CloudFlare with its customers.

So in the end the attackers decided to target a specific link in the network infrastructure instead of

Part 2

Slide 1. When we were discussing DDoS attacks, we often assumed that an attacker is able to control a botnet. So let's now speak about then a bit more.

Slide 2. Here are some of the resources that you can use.

Slide 3. We can define a botnet as a network of infected end-hosts, called bots, that is under the control of person or a group of people called botmaster. Each infected host runs some sort of bot software which gives the botmaster access to this host. The level of sophistication of this software will vary depending on the botnet. As you remember when we spoke about Mirai botnet, it was relatively complex and was able to launch a number of dismal of service attacks. But in general a bot would have the following functionality: it should be able to infect other vulnerable devices; it provide some sort of anti detection functionality; it should be able to carry out some attacks; and it should have a way to communicate with the botmaster and receive commands from him.

Slide 4. There are a number of ways to infect the hosts, and in many ways they are the same as when you want to infect a machine with a virus. Maybe one of the differences is that with normal virus that do something malicious to your computer, bot software stays dormant until the botmaster needs to use it. So once the machine is infected, the user would not really care because it does not directly harm him or her.

The infection strategy will depend on the infected host. If it a computer running normal operating system then the infections can be through application or OS-level vulnerabilities. For example if adobe acrobat used to have lots of vulnerabilities, so it was possible to exploit them by distributing a specifically crafted PDF document. Or with browser vulnerabilities, your computer can be infected by visiting a malicious web site. Until recently java script engines contains a lot of exploitable vulnerabilities.

A virus can be disguised as a useful program which would be distributed over files shares or p2P protocols such as Bittorrent. Botnet software could also be installed on previously infected machines by uploading new bot functionality. One could also use various social engineering techniques to trick users to install the malicious software. Or it could be installed as a part of a cracked software, for which one would otherwise need to pay.

In the case of Internet of Things devices or routers, you can either use software vulnerabilities which are plentiful because such devices only rarely updated. Or, since they often provide some sort of control interface protects by a password, you can try to brute force the passwords as was in the case of Mirai botnet.

Slide 5. When you design a botnet software, you need to think how you are going to communicate with your bots. The computer system from where you control your bots and send commands in called command and control center. The most common is centralized one. It's also the easiest to manage. For this purpose you can use a number of protocols such as IRC or HTTP.

But there are also other ways which include P2P command and control center, which is more robust against take downs, but also less efficient and unreliable from the attacker's view point. There is also locomotive approach and hybrid approach. Let's discuss each of them.

Slide 6. Centralized approach is the most simple but also efficient. In this approach the botmaster uses one Command and Control server to which each bot periodically connects and listens for commands. Each bot also provides status information to the server. Whenever is the botmaster wants to initiate a denial of service attacks, he sends corresponding command to the bots from this command and control centre. There is not constraints on the protocol that is used for communication between the bots the C&C server, its up to the botmaster. But the most commonly used protocols are IRC, HTTP or DNS.

Just to remind, IRC stand for Internet Relay Chat. It works by having a server and clients that connect to the server, and these clients can join chat rooms and exchange text messages. In the case of C&C server, the server creates a chat room to which the bots connect and wait for text commands posted by the server.

In the case of HTTP or DNS protocol, the the bots can send HTTP or DNS requests to the server and receive HTTP and DNS responses.

Slide 7. In P2P or decentralized architecture, the botmaster communicates with a small number of bots, and then the bots broadcast these command to other bots. This means its command and control server is not limited to a single, centralized machine, but rather can be done from every machine in its distributed network. In other words, every host running the malware process becomes part of the network, and is capable of sending, receiving, and executing the commands.Usually this is done by opening a listening on some port. The protocol for data exchange is up to the botmaster. For example FritzFrog botnet used its own proprietary protocol but other botnets used existing P2P protocols.

P2P architecture is more resilient but also it is harder to construct. One of the challenges is once a host is infected by a malware, how does not know where how to find the the botnet and join it. The way it is done is not much different from normal P2P networks such as bitcoin, BitTorrent, or Gnutella.

In general there are two way for new bots to join the P2P network

First you can encode an initial list of peers into the malware. When a new host is infected, it will try to contact each peer in that initial list to update its neighboring peer information.

Second, you can store a list of bots at some place on the Internet and you put the location in the bot malware code. In this way new bots can refresh its peer list by going to this web location and fetch the latest list.

These two methods can become a single point of failure. Law enforcement can reverse engineer you malware code, find out either the hardcoded list of peers or the web-location of the cache.

To overcome this vulnerability, botnet attackers may think of other ways to avoid introducing bootstrap procedure in P2P botnet construction. For example, when a bot A compromises a vulnerable host B, A passes its own peer list to this newly infected host B, and B will add A into this neighboring peer list. And any two bots that have found each other (e.g., through Internet scanning) will exchange their peer lists to con- struct new lists. In this way, the P2P botnet avoid bootstrap procedure relying on hard-coded lists.

Another thing worth mentioning is that is in general there are two way for bots to receive commands from Command and Control centre: Push and Pull. With push method, the bots actively listen for the command, and with pull, they periodically contact the server. For example in the case of centralized C&C centre, the server publishes the commands or updates, and bots send HTTPS requests. With push methods in P2P botnets, one of the challenges is which peers should a bot forward a command to?

Slide 8. Let's speak a bit more about centralized C&C centre. When you have a central server to control your bots, the common way is to hardcode its domain name in the bot's malware code. And if you want to move your control server to another location, you can change the DNS record to reflect this new address. This architecture is easy to construct and very efficient in distributing botmaster's commands; however, it has a single point of failure - the C&C server. One way to protect to protect your server from law enforcement is to host it with a hosting provider in countries that do not cooperate with law-enforcement quickly. This is both for the control server and the DNS server that resolves this name

As a botmaster, if law enforcement learns the IP address of the C&C server, then after she time they might be able to take it down. So you would be interested in hiding the location of the command and control centre. Once way to make IP address detection harder is by using Fast-Flux server network.

Slide 9. Fast-flux networks means that one domain name is assigned to a large number of IP addresses and these address are swapped in and out with hight frequency. Each of these IP address can belong to existing bots in he network. These bots then forward the request to the real command and control center.

Slide 10. For example consider the following picture. Assume a botmaster controls a botnet. He dedicates some of his bots to serve as proxies.

- Assume a normal bot tries to reach the C&C server for a new command.
- He will first try to resolve its domain name by contacting a DNS server controlled by the attackers

- Because the mapping to IP address constantly changes, the bot will get an IP address of some other bot from the proxy layer
- Our bot then will send the request to the resolved IP address, as if it was the command and control centre.
- The request is forwarded to the command and control center.
- Because the IP addresses constantly change, it is harder for the law-enforcement to track down the real IP address of the C&C
- It also makes it harder to block the C&C by blocking IP addresses.

Slide 11.

- Even with Fast-Flux networks there is a problem with single domain name.
- This domain name can be blocked, it's not easy to do, but in it's possible
- This will not allow the law-enforcement to take down the botnet, but it will allow to stop the botnet for some time.
- The attacker can register several domain names, but if this is a few domain names they can again be learned and blocked. This can be done for example by reverse engineering the bot malware
- One possible solution is to dynamically generate domain names
- In order to add unpredictability to the domain generation algorithm, the botmaster and bots can use some seed values that are only available to to the botmaster and the bots.
- You can still reverse the algorithm, but you cannot learn the domain names in advance (for example to register them before the attacker does)
- The effort required to defend against this kind of attack increases massively, as the botmaster can choose to simply pick one of the possible domains, register it and make sure it provides new instructions, or an update, during the validity of this domain name.
- For example the seed value can include a timestamp and a twitter messages.

Slide 12.

- A special case of centralised botnets is so-called locomotive botnets. The idea behind locomotion is to rely on a centralised C&C model and to synchronously switch the centralised component regularly. Figure 4 illustrates the concept of locomotion. By changing domain names or associated servers frequently, the botmaster exploits the

<u>Slide 13.</u>

- With regard to robustness, peer-to-peer botnets have the major advantage that no central server can be attacked to mitigate them directly. On the other hand, relying on the selfpropagation of commands through the botnet means a low reaction time.
- In this case, optional centralised servers are used additionally for commanding, thus making the botnet hybrid.
- A possible implementation could be that The central C&C server distributes peer lists to bots – at certain intervals
- If the C&C cannot be reached, P2P and DGA domains are used for communication