CS 241 Honors
Minilectures 2 - Revenge of the Sys

University of Illinois Urbana–Champaign

March 7, 2018

Logistics

- Next week is Peer Mentoring - Try to have something to share with your peers!

CS 241 Honors
Linking and Loading

Kevin Hong

University of Illinois Urbana–Champaign

March 7, 2018

Linking and Loading

Motivation

```c
#include <pthread.h> //Dynamic Linked Library

void* hello_thread(void *payload){ //Pthread
    write(1, "Hello world!", 12); //Sys Call
    return NULL;
}

int main(){ //Process Start
    pthread_create(NULL, hello_thread,
        NULL, NULL);
    pthread_exit() //Some kind of scheduling
}
```

What are we even talking about?

There are two types of libraries, those compiled with your programs and those that are linked dynamically at runtime. There are many benefits to use programs that get compiled with your program, but some drawbacks.

Cost-Benefit Analysis: Compile-Time Libraries

Benefits

- You have the source code/debug checking
- All code is in your code segment
- You can modify the library

Cost-Benefit Analysis: Compile-Time Libraries

Benefits

- You have the source code/debug checking
- All code is in your code segment
- You can modify the library

Drawbacks

- Updating is often tedious
- Your executable is bigger
- Your library cannot be reused by other applications

Solution: Dynamic Libraries

- What if we have one library that a bunch of programs can use (make it read only) and have it dynamically link the function calls in the program?

Solution: Dynamic Libraries

- What if we have one library that a bunch of programs can use (make it read only) and have it dynamically link the function calls in the program?
- Updating your executable's library is a piece of cake

Solution: Dynamic Libraries

- What if we have one library that a bunch of programs can use (make it read only) and have it dynamically link the function calls in the program?
- Updating your executable's library is a piece of cake
- Reduce the size of your executable

Solution: Dynamic Libraries

- What if we have one library that a bunch of programs can use (make it read only) and have it dynamically link the function calls in the program?
- Updating your executable's library is a piece of cake
- Reduce the size of your executable
- That library can be used by other applications.

Dynamic Lookup table?

- Cool! But where are the library functions?

Dynamic Lookup table?

- Cool! But where are the library functions?
- Any problem in computer science and be solved with another layer of abstraction.

Dynamic Lookup table?

- Cool! But where are the library functions?
- Any problem in computer science and be solved with another layer of abstraction.
- Have the functions in your code point to pointer where the functions are going to be.

Dynamic Lookup table?

- Cool! But where are the library functions?
- Any problem in computer science and be solved with another layer of abstraction.
- Have the functions in your code point to pointer where the functions are going to be.
- Have your code jump to the pointer and the pointer jump to the actual function

Where is the table stored

"In Unix-like systems that use ELF for executable images and dynamic libraries, such as Solaris, 64-bit versions of HP-UX, Linux, FreeBSD, NetBSD, OpenBSD, and DragonFly BSD, the path of the dynamic linker that should be used is embedded at link time into the .interp section of the executable's PT_INTERP segment. In those systems, dynamically loaded shared libraries can be identified by the filename suffix .so (shared object)." - Wikipedia

So what does that mean?

- Exec generates lookup-table (PT_INTERP segment)
- Calls to a library will cause a jump to the lookup table
- Lookup table entry will redirect the program to the library function
- After execution, returns back to your program

So about that library

- During exec, the function checks what libraries you need.

So about that library

- During exec, the function checks what libraries you need.
- If that library is already loaded into memory, increase the reference count and link the functions.

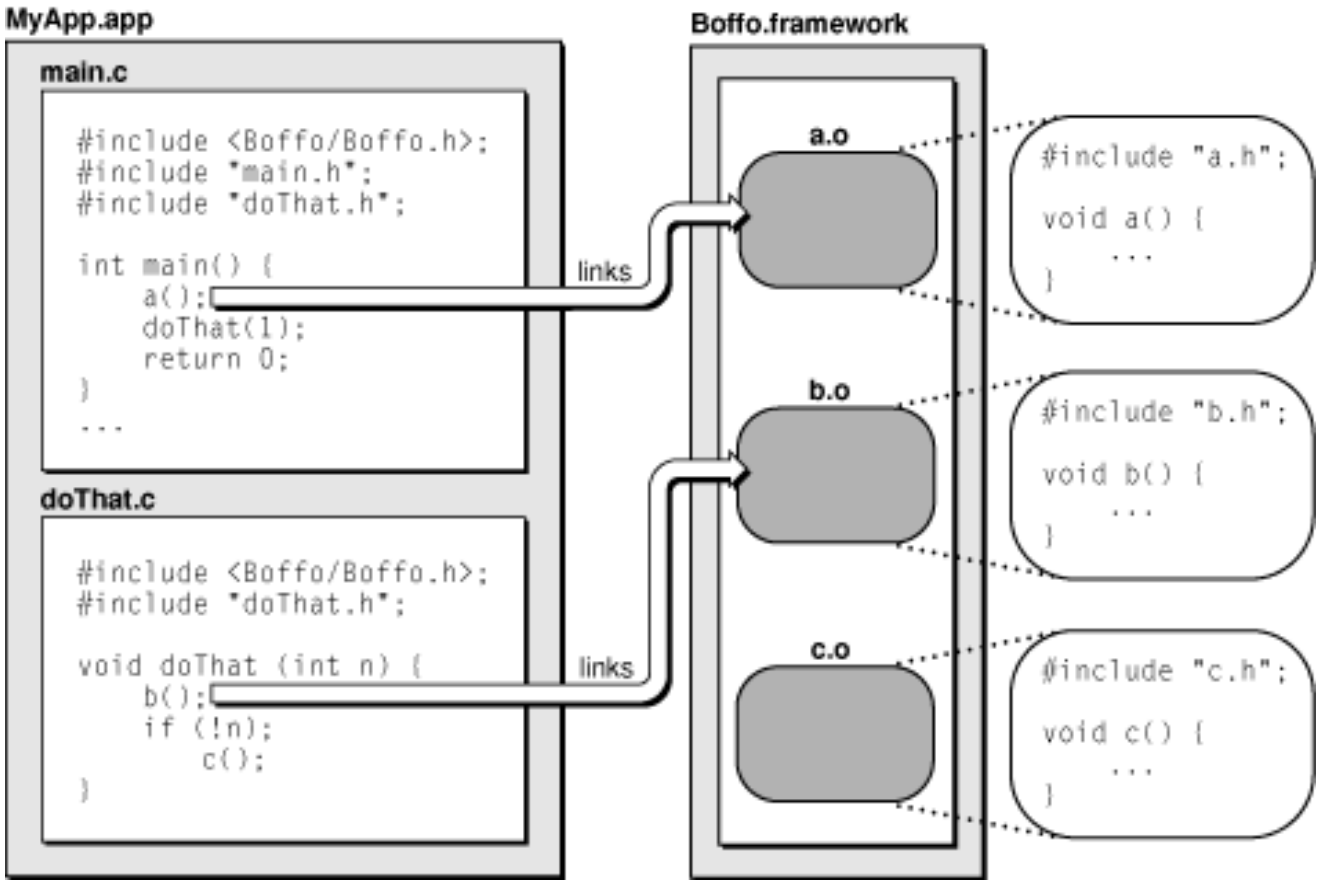
So about that library

- During exec, the function checks what libraries you need.
- If that library is already loaded into memory, increase the reference count and link the functions.
- Else, mmap the library into memory. Set the executable bit (memory can either be executable or writeable) and then link the function. Cache this library's location.

So about that library

- During exec, the function checks what libraries you need.
- If that library is already loaded into memory, increase the reference count and link the functions.
- Else, mmap the library into memory. Set the executable bit (memory can either be executable or writeable) and then link the function. Cache this library's location.
- When a process is done, reduce the reference count and return the page back to the system if need be.

So about that library



Drawbacks: Hacking

- Perfect! Let's dynamically link ALL THE THINGS!

Drawbacks: Hacking

- Perfect! Let's dynamically link ALL THE THINGS!
- Not quite. You have to trust the library you are linking to. What if the library has been hacked?

Drawbacks: Hacking

- Perfect! Let's dynamically link ALL THE THINGS!
- Not quite. You have to trust the library you are linking to. What if the library has been hacked?
- Some viruses redirect library calls (hint: LD_PRELOAD) thus, executing commands that the user is not aware of.

Drawbacks: Hacking with Bugs

- What if there is a bug in the DLL? Added point of failure.

Drawbacks: Hacking with Bugs

- What if there is a bug in the DLL? Added point of failure.
- Example: libc had a buffer overflow bug (any application that uses libc was affected).

Drawbacks: Hacking with Bugs

- What if there is a bug in the DLL? Added point of failure.
- Example: libc had a buffer overflow bug (any application that uses libc was affected).
- But the pros outweigh the cons so DLLs are here to stay.

Drawbacks: Hacking with Bugs

- Demo time!

Edit history

- Bhuvan Venkatesh
- Aneesh Durg

CS 241 Honors
Life without GLIBC

Aneesh Durg

University of Illinois Urbana–Champaign

March 7, 2018

glibc is just another library?

- yes

glibc is just another library?

- yes
- ...and no
  - glibc is a shared library

glibc is just another library?

- yes
- ...and no
  - glibc is a shared library
  - but glibc does some pretty special things

Need proof?

- Let's compile sample.c and check linking output
- Let's also check gdb using:
  - set backtrace past-main on
  - set backtrace past-entry on

What are the advantages of using glibc?

- Well maintained

What are the advantages of using glibc?

- Well maintained
- Well optimized

What are the advantages of using glibc?

- Well maintained
- Well optimized
- Provides clean, easy-to-use interface (we'll see more about this later)

What are the advantages of using glibc?

- Well maintained
- Well optimized
- Provides clean, easy-to-use interface (we'll see more about this later)
- Unless you know what you're doing, DO NOT get rid of glibc!

What are the disadvantages of using glibc?

- Bloated?

What are the disadvantages of using glibc?

- Bloated?
  - A few people have developed alternatives

What are the disadvantages of using glibc?

- Bloated?
  - A few people have developed alternatives
- Use-case specific optimization (ignore the general case - think about your malloc)

Obesity the new epidemic?

- What's wrong with a library being bloated?

Obesity the new epidemic?

- What's wrong with a library being bloated?
  - compile time
  - limited resource environments
  - Remember - just loading an executable requires memory

Obesity the new epidemic?

- What's wrong with a library being bloated?
  - compile time
  - limited resource environments
  - Remember - just loading an executable requires memory
- So how bad are we actually talking? (in terms of bytes)

Tools of the trade

Here's some tools we'll use today:

- `wc` - Allows us to check size
- `objdump` - Allows us to dump a compiled program
- `gcc` - We're going to need some special flags
- `x86 asm` - Not afraid to get your hands dirty with assembly right?

Let's get started

- Step 1 - `-nostdlib`

Let's get started

- Step 1 - `-nostdlib`
- Step 2 - Need to define `start`

Let's get started

- Step 1 - `-nostdlib`
- Step 2 - Need to define `start`
- Step 3 - What about exit? Need some syscalls!

```
printf "#include<sys/syscall.h>\nNUMBER SYS_name" |
gcc -E -
```

Let's get started

- Step 1 - `-nostdlib`
- Step 2 - Need to define `start`
- Step 3 - What about exit? Need some syscalls!
  ```
  printf "#include<sys/syscall.h>\nNUMBER SYS_name" |
  gcc -E -
  ```
- Step 4 - Can we do better? Maybe gcc will help us?

Let's get started

- Step 1 - `-nostdlib`
- Step 2 - Need to define `start`
- Step 3 - What about exit? Need some syscalls!
  ```
  printf "#include<sys/syscall.h>\nNUMBER SYS_name" |
  gcc -E -
  ```
- Step 4 - Can we do better? Maybe gcc will help us?
- Step 5 - Okay, but can we do EVEN better?

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.
  - You want to learn more about the tools you use.

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.
  - You want to learn more about the tools you use.
- What are some alternative methods?

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.
  - You want to learn more about the tools you use.
- What are some alternative methods?
  - Use a different language

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.
  - You want to learn more about the tools you use.
- What are some alternative methods?
  - Use a different language
  - Write a shared library that wraps glibc

Sanity Check - literally

- 90% of the time, this is #extra
- When might this be useful?
  - You're making a new stdlib to give C your own spin and features.
  - You have very limited resources.
  - You want to learn more about the tools you use.
- What are some alternative methods?
  - Use a different language
  - Write a shared library that wraps glibc
  - Compile glibc with only the features that you want - hooray for open source!

Sources

- http://weeb.ddns.net/0/programming/c_without_standard_library_linux.txt

Distributed Denial-of-Service (DDoS) Attacks

Shreyas Patil

University of Illinois Urbana–Champaign

March 7, 2018

Context

On Feb 28, 2018, GitHub was targeted by the largest DDoS attack ever.

Context

On Feb 28, 2018, GitHub was targeted by the largest DDoS attack ever.

- Peak traffic of 1.35 Tbps, 126.9 million packets/second.

Context

On Feb 28, 2018, GitHub was targeted by the largest DDoS attack ever.

- Peak traffic of 1.35 Tbps, 126.9 million packets/second.
- Used memcached servers as an attack vector.
- No botnet involved.

Context

On Feb 28, 2018, GitHub was targeted by the largest DDoS attack ever.

- Peak traffic of 1.35 Tbps, 126.9 million packets/second.
- Used memcached servers as an attack vector.
- No botnet involved.

Previous known record was the attack on Dyn in October 2016.

- Peak traffic of around 1 Tbps.
- Used a botnet of IoT devices infected by Mirai.
- Took down siginifcant portions of the internet
  Amazon.com, GitHub, PayPal, Reddit, Slack

Background

- User Datagram Protocol (UDP)
- DoS and DDoS
- Amplification
- memcached
- Border Gateway Protocol (BGP)

User Datagram Protocol (UDP)

- Transport Layer (Layer 4) protocol on top of IP.
- Not TCP (Transmission Control Protocol).

User Datagram Protocol (UDP)

- Transport Layer (Layer 4) protocol on top of IP.
- Not TCP (Transmission Control Protocol).
- Connectionless.
- Send and forget.
- Like sending a letter in the mail.
  - No guarantee of delivery, order, or correctness.
  - Used for many real-time applications, where we only care about the freshest information.

DoS and DDoS

Denial of Service (DoS): attacks in which the target is sent massive amounts of network traffic in an attempt to make the target inaccessible to legitimate users or otherwise malfunction.

- Easy to carry out: just send a lot of traffic.
- Easy to defend against: just block the source of traffic.
- Small; large targets can absorb the traffic easily.

DoS and DDoS

Denial of Service (DoS): attacks in which the target is sent massive amounts of network traffic in an attempt to make the target inaccessible to legitimate users or otherwise malfunction.

- Easy to carry out: just send a lot of traffic.
- Easy to defend against: just block the source of traffic.
- Small; large targets can absorb the traffic easily.

Distributed Denial of Service DDoS): DoS attacks in which traffic is sent to the target from a large number of sources.

- Hard to carry out (usually): need control over many hosts.
- Hard to defend against: can't simply block a few hosts.
- Can be massive in scale; viable against large targets.

Amplification

Amplification: when a server responds to an n byte request with a response of size (much) larger than n.

- Used in DDoS attacks to greatly increase attack size.
- Common amplifiers are NTP, DNS, SNMP.
  - NTP: *monlist*, 550x amplification
  - DNS: 100x amplification or more

Amplification

Amplification: when a server responds to an n byte request with a response of size (much) larger than n.

- Used in DDoS attacks to greatly increase attack size.
- Common amplifiers are NTP, DNS, SNMP.
  - NTP: *monlist*, 550x amplification
  - DNS: 100x amplification or more

  (note: these are all UDP services. Why?)

Amplification

Amplification: when a server responds to an n byte request with a response of size (much) larger than n.

- Used in DDoS attacks to greatly increase attack size.
- Common amplifiers are NTP, DNS, SNMP.
  - NTP: *monlist*, 550x amplification
  - DNS: 100x amplification or more

  (note: these are all UDP services. Why?)
- UDP has no way of verifying source IP.
- Send requests to amplifiers with the target's IP as source; amplifier responds to target with large request.

Memcached

Open-source "distributed memory object caching system".

- Effectively a large, distributed hash table.
- Used to minimize accesses to external data sources (like APIs or large databases).

Memcached

Open-source "distributed memory object caching system".

- Effectively a large, distributed hash table.
- Used to minimize accesses to external data sources (like APIs or large databases).
- Not supposed to be exposed to the public Internet.

Memcached

Open-source "distributed memory object caching system".

- Effectively a large, distributed hash table.
- Used to minimize accesses to external data sources (like APIs or large databases).
- Not supposed to be exposed to the public Internet.
- Uses UDP.

Memcached

Open-source "distributed memory object caching system".

- Effectively a large, distributed hash table.
- Used to minimize accesses to external data sources (like APIs or large databases).
- Not supposed to be exposed to the public Internet.
- Uses UDP. (See where this is going?)

Memcached

Open-source "distributed memory object caching system".

- Effectively a large, distributed hash table.
- Used to minimize accesses to external data sources (like APIs or large databases).
- Not supposed to be exposed to the public Internet.
- Uses UDP. (See where this is going?)
- Over 50,000x amplification. 200 B request $\rightarrow$ 100 MB response

Border Gateway Protocol (BGP)

If DNS is the address book of the Internet, BGP is street signs.

Border Gateway Protocol (BGP)

If DNS is the address book of the Internet, BGP is street signs.

- The internet is a network of Autonomous Systems (AS).
  AS38: University of Illinois.
  AS36459: GitHub

Border Gateway Protocol (BGP)

If DNS is the address book of the Internet, BGP is street signs.

- The internet is a network of Autonomous Systems (AS).
  AS38: University of Illinois.
  AS36459: GitHub
- Each AS's core routers advertise routes.
  "72.36.64.0/18 is here!"
  "8.42.112.0/20 is that way."

Border Gateway Protocol (BGP)

If DNS is the address book of the Internet, BGP is street signs.

- The internet is a network of Autonomous Systems (AS).
  AS38: University of Illinois.
  AS36459: GitHub
- Each AS's core routers advertise routes.
  "72.36.64.0/18 is here!"
  "8.42.112.0/20 is that way."
- ASs route traffic to/from each other.

February 2018 GitHub Attack

- Used publicly exposed memcached servers as amplifiers.
  - There are estimated to be around 100,000 such servers.
- Hit GitHub with 1.3Tbps of traffic at its peak.

February 2018 GitHub Attack

- Used publicly exposed memcached servers as amplifiers.
  - There are estimated to be around 100,000 such servers.
- Hit GitHub with 1.3Tbps of traffic at its peak.
- GitHub survived! This was an example of excellent mitigation.
- There was apparently a *larger* attack the next day.

GitHub Attack Mitigation

- Automated systems detected the attack within 10 minutes.

GitHub Attack Mitigation

- Automated systems detected the attack within 10 minutes.
- GitHub routers withdrew their BGP advertisements.

GitHub Attack Mitigation

- Automated systems detected the attack within 10 minutes.
- GitHub routers withdrew their BGP advertisements.
- Akamai Prolexic began to announce GitHub's IP blocks.
  - Akamai Prolexic is a DDoS protection vendor.

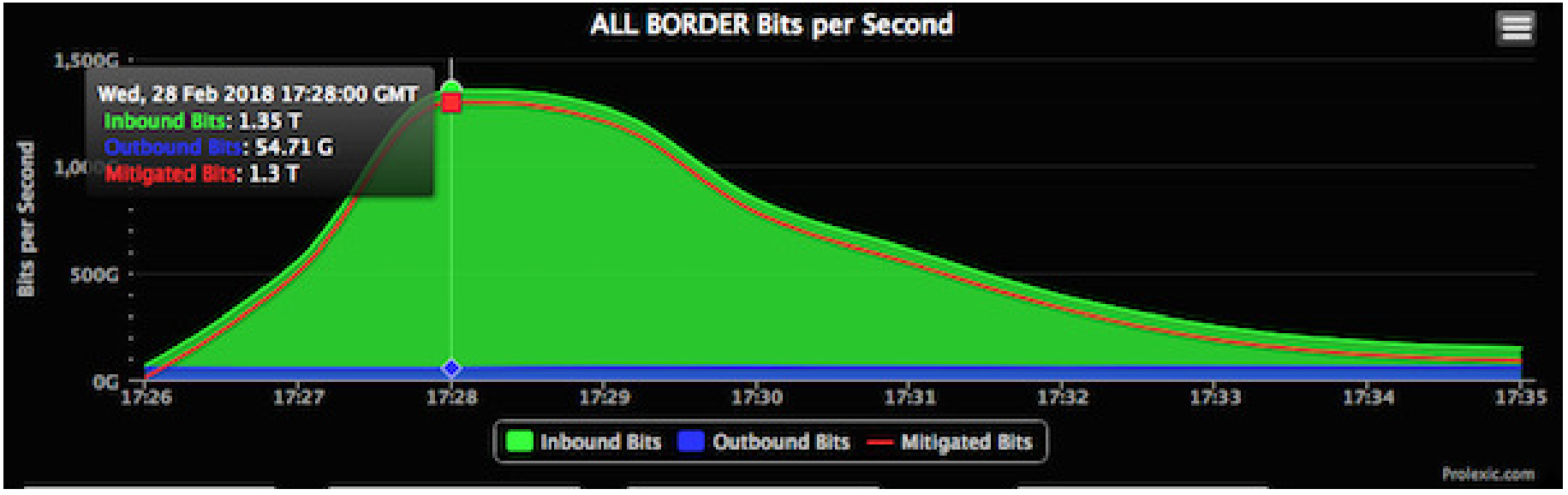
GitHub Attack Mitigation

- Automated systems detected the attack within 10 minutes.
- GitHub routers withdrew their BGP advertisements.
- Akamai Prolexic began to announce GitHub's IP blocks.
  - Akamai Prolexic is a DDoS protection vendor.
- All traffic to GitHub went through Prolexic's scrubbing centers.
  - Datacenters capable of massive capacity filter out all DDoS traffic.
  - Filtered out UDP traffic on port 11211 (memcached default).
  - Legitimate traffic is proxied back to GitHub.

GitHub Attack Mitigation

- Automated systems detected the attack within 10 minutes.
- GitHub routers withdrew their BGP advertisements.
- Akamai Prolexic began to announce GitHub's IP blocks.
  - Akamai Prolexic is a DDoS protection vendor.
- All traffic to GitHub went through Prolexic's scrubbing centers.
  - Datacenters capable of massive capacity filter out all DDoS traffic.
  - Filtered out UDP traffic on port 11211 (memcached default).
  - Legitimate traffic is proxied back to GitHub.
- GitHub traffic went through Prolexic for about 6 hours.

GitHub Attack Mitigation



Lessons

- Don't expose memcached servers to the Internet!!!

Lessons

- Don't expose memcached servers to the Internet!!!
- Be like GitHub.

Lessons

- Don't expose memcached servers to the Internet!!!
- Be like GitHub.

Learn more:

- GitHub Incident Report
- Akamai Incident Report
- Poke around with BGP: `bgp.he.net`
- Other attacks: `krebsonsecurity.com/tag/ddos/`