BOTNETS

GRAD SEC

NOV 21 2017

TODAY'S PAPERS

Understanding the Mirai Botnet

Manos Antonakakis* Tim April‡ Michael Bailey† Matthew Bernhard* Elie Bursztein◊
Jaime Cochran▷ Zakir Durumeric* J. Alex Halderman* Luca Invernizzi◊
Michalis Kallitsis§ Deepak Kumar† Chaz Lever* Zane Ma†* Joshua Mason†
Damian Menscher◊ Chad Seaman‡ Nick Sullivan▷ Kurt Thomas◊ Yi Zhou†

‡*Akamai Technologies* ▷*Cloudflare* ***Georgia Institute of Technology* ***Google*
§*Merit Network* †*University of Illinois Urbana-Champaign* ◊*University of Michigan*

Abstract

The Mirai botnet, composed primarily of embedded and IoT devices, took the Internet by storm in late 2016 when it overwhelmed several high-profile targets with massive distributed denial of service (DDoS) attacks. In this paper, we provide a seven-month retrospective analysis of Mirai's growth to a peak of 600k infections and a history of its DDoS victims. By combining a variety of measurement perspectives, we analyze how the botnet emerged, what classes of devices were affected, and how Mirai variants evolved and competed for vulnerable hosts. Our measurements serve as a lens into the fragile ecosystem of IoT devices. We argue that Mirai may represent a sea change in the evolutionary development of botnets—the simplicity through which devices were infected and its precipitous growth, demonstrate that novice malicious techniques can compromise enough low-end devices to threaten even some of the best-defended targets. To address this risk, we recommend technical and non-technical interventions, as well as propose future research directions.

1 Introduction

Starting in September 2016, a spree of massive distributed denial-of-service (DDoS) attacks temporarily crippled Krebs on Security [45], OVH [43], and Dyn [35]. The initial attack on Krebs exceeded 600 Gbps in volume [46]—among the largest on record. Remarkably, this overwhelming traffic was sourced from hundreds of thousands of some of the Internet's least powerful hosts—Internet of Things (IoT) devices—under the control of a new botnet named Mirai.

While other IoT botnets such as BASHLITE [85] and Carna [38] preceded Mirai, the latter was the first to emerge as a high-profile DDoS threat. What explains Mirai's sudden rise and massive scale? A combination of factors—efficient spreading based on Internet wide scanning, rampant use of insecure default passwords in IoT products, and the insight that keeping the botnet's behavior simple would allow it to infect many heterogeneous devices—all played a role. Indeed, Mirai has spawned many variants that follow the same infection strategy, leading to speculation that "IoT botnets are the new normal of DDoS attacks" [54].

In this paper, we investigate the precipitous rise of Mirai and the fragile IoT ecosystem it has subverted. We present longitudinal measurements of the botnet's growth, composition, evolution, and DDoS activities from August 1, 2016 to February 28, 2017. We draw from a diverse set of vantage points including network telescope probes, Internet wide banner scans, IoT honeypots, C2 milkers, DNS traces, and logs provided by attack victims. These unique datasets enable us to conduct the first comprehensive analysis of Mirai and post technical and non-technical defenses that may stymie future attacks.

We track the outbreak of Mirai and find the botnet infected nearly 65,000 IoT devices in its first 20 hours before reaching a steady state population of 200,000–300,000 infections. These bots fell into a narrow band of geographic regions and autonomous systems, with Brazil, Columbia, and Vietnam disproportionately accounting for 41.5% of infections. We confirm that Mirai targeted a variety of IoT and embedded devices ranging from DVRs, IP cameras, routers, and printers, but find Mirai's ultimate device composition was strongly influenced by the market shares and design decisions of a handful of consumer electronics manufacturers.

By statically analyzing over 1,000 malware samples, we document the evolution of Mirai into dozens of variants propagated by multiple, competing botnet operators. These variants attempted to improve Mirai's detection avoidance techniques, add new IoT device targets, and introduce additional DNS resilience. We find that Mirai harnessed its evolving capabilities to launch over 15,000 attacks against not only high-profile targets (e.g., Krebs

*Denotes primary, lead, or "first" author

Your Botnet is My Botnet: Analysis of a Botnet Takeover

Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski,
Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna

University of California, Santa Barbara
{bstone,marco,sullivan,rgilbert,msz,kemm,chris,vigna}@cs.ucsb.edu

ABSTRACT

Botnets, networks of malware-infected machines that are controlled by an adversary, are the root cause of a large number of security problems on the Internet. A particularly sophisticated and insidious type of bot is Torpig, a malware program that is designed to harvest sensitive information (such as bank account and credit card data) from its victims. In this paper, we report on our efforts to take control of the Torpig botnet and study its operations for a period of ten days. During this time, we observed more than 180 thousand infections and recorded almost 70 GB of data that the bots collected. While botnets have been "hijacked" and studied previously, the Torpig botnet exhibits certain properties that make the analysis of the data particularly interesting. First, it is possible (with reasonable accuracy) to identify unique bot infections and relate that number to the more than 1.2 million IP addresses that contacted our command and control server. Second, the Torpig botnet is large, targets a variety of applications, and gathers a rich and diverse set of data from the infected victims. This data provides a new understanding of the type and amount of personal information that is stolen by botnets.

1. INTRODUCTION

Malicious code (or malware) has become one of the most pressing security problems on the Internet. In particular, this is true for bots [5], a type of malware that is written with the intent of taking over a large number of hosts on the Internet. Once infected with a bot, the victim host will join a botnet, which is a network of compromised machines that are under the control of a malicious entity, typically referred to as the botmaster. Botnets are the primary means for cyber-criminals to carry out their nefarious tasks, such as sending spam mails [36], launching denial-of-service attacks [29], or stealing personal data such as mail accounts or bank credentials [15, 39]. This reflects the shift from an environment in which malware was developed for fun, to the current situation, where malware is spread for financial profit.

Given the importance of the problem, significant research effort has been invested to gain a better understanding of the botnet phenomenon.

One approach to study botnets is to perform *passive analysis* of secondary effects that are caused by the activity of compromised machines. For example, researchers have collected spam mails that were likely sent by bots [47]. Through this, they were able to make indirect observations about the sizes and activities of different spam botnets. Similar measurements focused on DNS queries [34, 35] or DNS blacklist queries [37] performed by bot-infected machines. Other researchers analyzed network traffic (netflow data) at the tier-1 ISP level for cases that are characteristic for certain botnets (such as scanning or long-lived IRC connections) [24]. While the analysis of secondary effects provides interesting insights into particular botnet-related behaviors, one can typically only monitor a small portion of the Internet. Moreover, the detection is limited to those botnets that actually exhibit the activity targeted by the analysis.

A more active approach to study botnets is via *infiltration*. That is, using an actual malware sample or a client simulating a bot, researchers join a botnet to perform analysis from the inside. To achieve this, honeypots, honey clients, or spam traps are used to obtain a copy of a malware sample. The sample is then executed in a controlled environment, which makes it possible to observe the traffic that is exchanged between the bot and its command and control (C&C) server(s). In particular, one can record the commands that the bot receives and monitor its malicious activity. For some botnets that rely on a central IRC-based C&C server, joining a botnet can also reveal the IP addresses of other clients (bots) that are concurrently logged into the IRC channel [4, 11, 35]. While this technique worked well for some time, attackers have unfortunately adapted, and most current botnets use stripped-down IRC or HTTP servers as their centralized command and control channels. With such C&C infrastructures, it is no longer possible to make reliable statements about other bots by joining as a client.

Interestingly, due to the open, decentralized nature of peer-to-peer (P2P) protocols, it is possible to infiltrate P2P botnets such as Storm. To this end, researchers have developed crawlers that actively search the P2P network for client nodes that exhibit bot-like characteristics. Such crawls are the basis for studying the number of infected machines [18, 21] and the ways in which criminals orchestrate spam campaigns [23]. Of course, the presented techniques only work in P2P networks that can be actively crawled. Thus, they are not applicable to a majority of current botnets, which rely mostly on a centralized IRC or HTTP C&C infrastructure.

To overcome the limitations of passive measurements and infiltration—in particular in the case of centralized IRC and HTTP botnets—one can attempt to *hijack* the entire botnet, typically by taking control of the C&C channel. One way to achieve this is to directly seize the physical machines that host the C&C infrastructure [8]. Of course, this is only an option for law enforcement agencies. Alternatively, one can tamper with the domain name ser-

CCS'09, November 9–13, 2009, Chicago, Illinois, USA.

BOTNETS

- Collection of compromised machines (bots) under unified control of an attacker (botmaster)

- Method of compromise decoupled from method of control
  - Launch a worm/virus, etc.: remember, payload is orthogonal!

- Upon infection, a new bot "phones home" to *rendezvous* with botnet "command-and-control" (C&C)

- Botmaster uses C&C to push out commands and updates

BOTNETS

- Collection of compromised machines (bots) under unified control of an attacker (botmaster)

- Method of compromise decoupled from method of control
  - Launch a worm/virus, etc.: remember, payload is orthogonal!
- Upon infection, a new bot "phones home" to *rendezvous* with botnet "command-and-control" (C&C)
- Botmaster uses C&C to push out commands and updates

BOTNETS

- Collection of compromised machines (bots) under unified control of an attacker (botmaster)

- Method of compromise decoupled from method of control
  - Launch a worm/virus, etc.: remember, payload is orthogonal!
- Upon infection, a new bot "phones home" to *rendezvous* with botnet "command-and-control" (C&C)

C&C

- Botmaster uses C&C to push out commands and updates

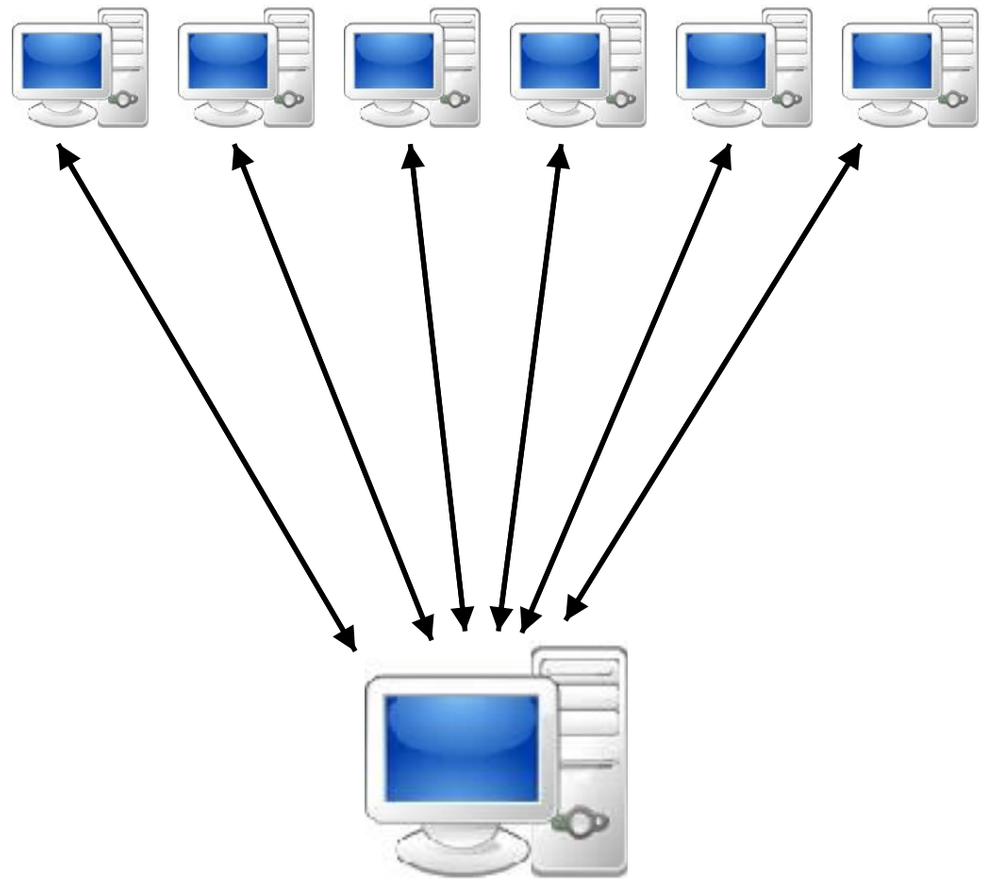
BOTNETS

- Collection of compromised machines (bots) under unified control of an attacker (botmaster)

- Method of compromise decoupled from method of control
  - Launch a worm/virus, etc.: remember, payload is orthogonal!

- Upon infection, a new bot "phones home" to *rendezvous* with botnet "command-and-control" (C&C)

- Botmaster uses C&C to push out commands and updates



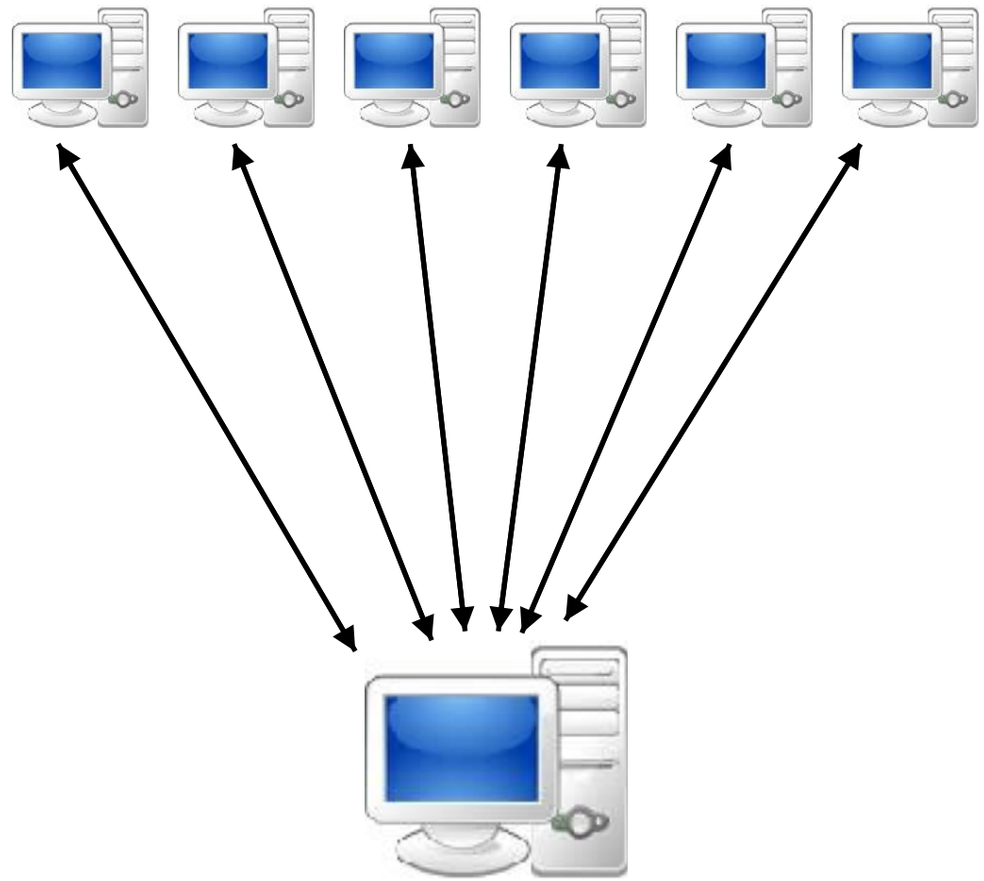
BOTNETS

- Collection of compromised machines (bots) under unified control of an attacker (botmaster)

- Method of compromise decoupled from method of control
  - Launch a worm/virus, etc.: remember, payload is orthogonal!

- Upon infection, a new bot "phones home" to *rendezvous* with botnet "command-and-control" (C&C)

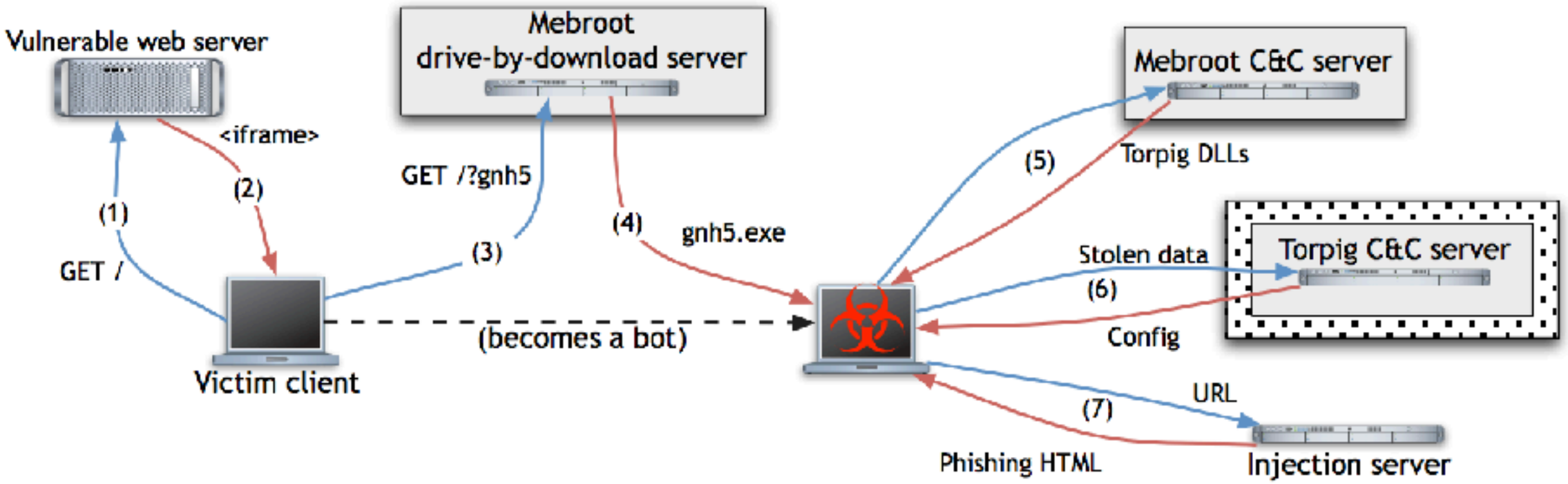
- Botmaster uses C&C to push out commands and updates



C&C

Topology can be star (like this), hierarchical, peer-to-peer…

TORPIG


Vulnerable web server
<iframe>
(1)
GET /
(2)
Victim client
Mebroot
drive-by-download server
GET /?gnh5
(3)
(4)
gnh5.exe
(becomes a bot)
Mebroot C&C server
(5)
Torpig DLLs
Stolen data
(6)
Config
Torpig C&C server
URL
(7)
Phishing HTML
Injection server

DOMAIN FLUXING

How do these bots know where to go?

Issue DNS lookups for a known hostname

Provides a level of indirection:

Bots know the name ahead of time, but the botmaster can move the C&C node to different IP addresses, as needed

Problem:

Network operators will simply firewall a known-malicious domain name

Domain fluxing:

Generate random domain names. Move on by the time you're found

```
suffix =  ["anj", "ebf", "arm", "pra", "aym", "unj",
    "ulj", "uag", "esp", "kot", "onv", "edc"]

def generate_daily_domain():
    t = GetLocalTime()
    p = 8
    return generate_domain(t, p)

def scramble_date(t, p):
    return (((t.month ^ t.day) + t.day) * p) +
        t.day + t.year

def generate_domain(t, p):
    if t.year < 2007:
        t.year = 2007
    s = scramble_date(t, p)
    c1 = (((t.year >> 2) & 0x3fc0) + s) % 25 + 'a'
    c2 = (t.month + s) % 10 + 'a'
    c3 = ((t.year & 0xff) + s) % 25 + 'a'
    if t.day * 2 < '0' ||  t.day * 2 > '9':
        c4 = (t.day * 2) % 25 + 'a'
    else:
        c4 = t.day % 10 + '1'
    return c1 + 'h' + c2 + c3 + 'x' + c4 +
        suffix[t.month - 1]
```

Listing 1: Torpig daily domain generation algorithm.

YOUR BOTNET IS MY BOTNET

Domain fluxing:

Generate random domain names.

Move on by the time you're found

(This) Botnet takeover:

Anticipate the domain names;

register those not yet purchased

YOUR BOTNET IS MY BOTNET

Domain fluxing:

Generate random domain names.
Move on by the time you're found

(This) Botnet takeover:

Anticipate the domain names;
register those not yet purchased

ETHICAL CONCERN: DO NO HARM

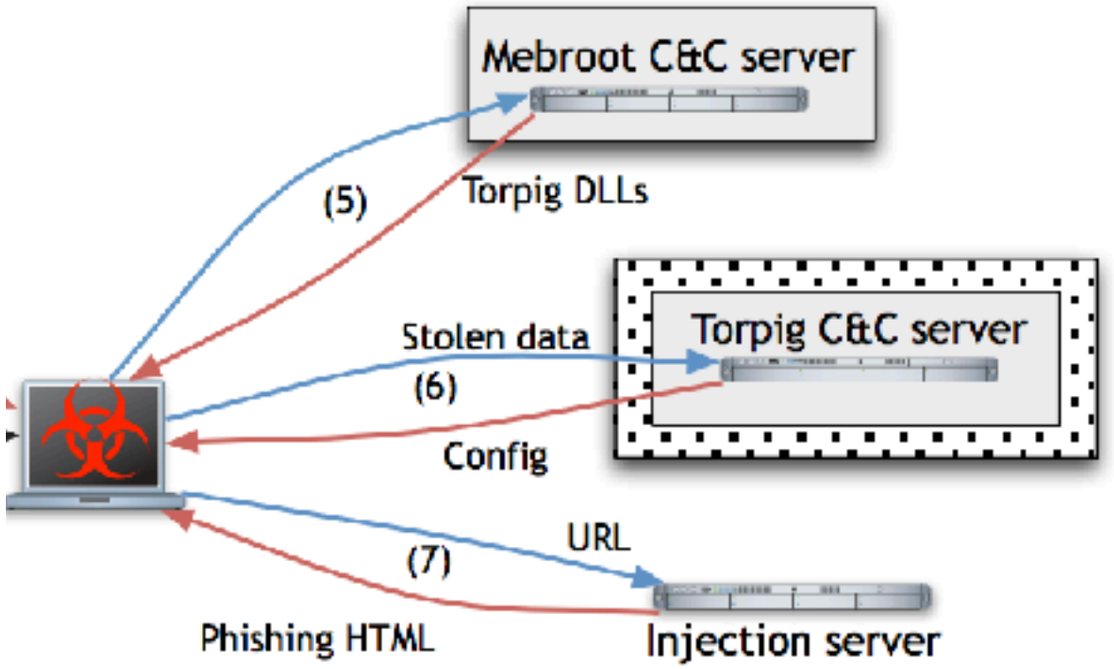
PRINCIPLE 1. *The sinkholed botnet should be operated so that any harm and/or damage to victims and targets of attacks would be minimized.*

Keep in touch with bots,
but never send a new config file

PRINCIPLE 2. *The sinkholed botnet should collect enough information to enable notification and remediation of affected parties.*

Worked with ISPs and law
enforcement to take them down

WHAT DID THEY LEARN?



70GB over 10 days

Data Type	Data Items (#)
Mailbox account	54,090
Email	1,258,862
Form data	11,966,532
HTTP account	411,039
FTP account	12,307
POP account	415,206
SMTP account	100,472
Windows password	1,235,122

Table 1: Data items sent to our C&C server by Torpig bots.

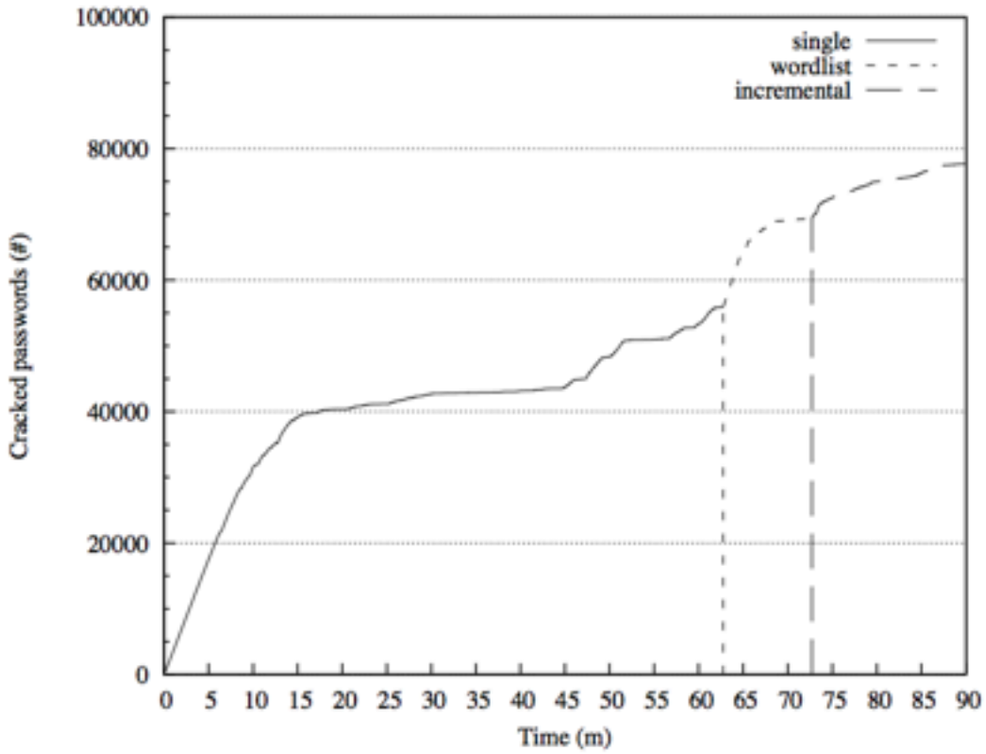


Figure 13: Number of passwords cracked in 90 minutes by the John the Ripper password cracker tool. Vertical lines indicate when John switches cracking mode. The first vertical line represents the switching from simple transformation techniques ("single" mode) to wordlist cracking, the second from wordlist to brute-force ("incremental").

BOTNET SIZE: HOW TO COUNT?

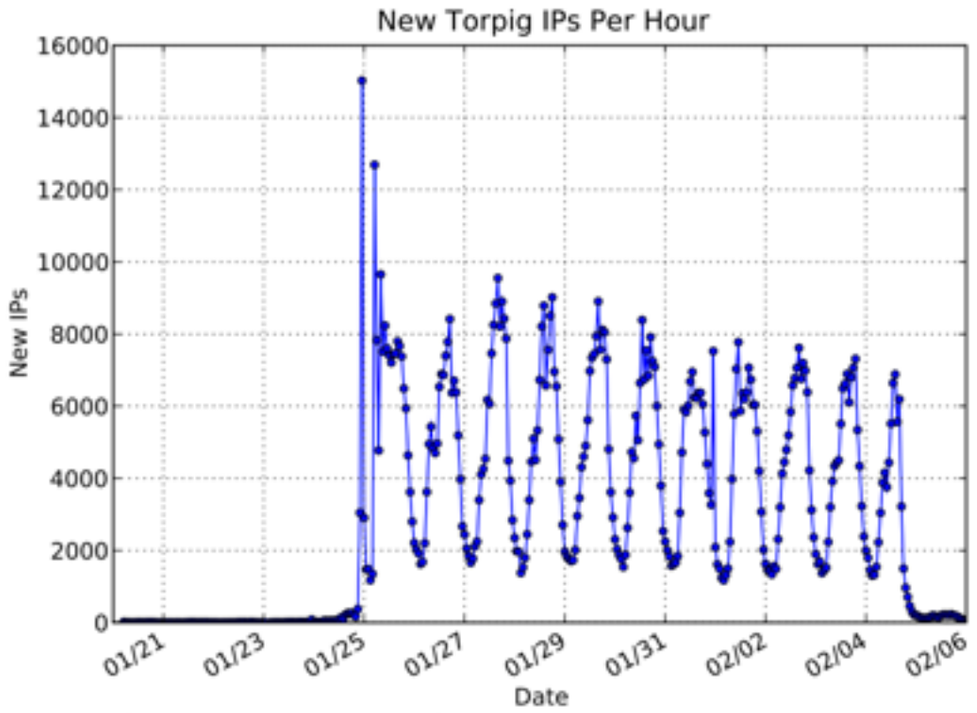


Figure 5: New unique IP addresses per hour.

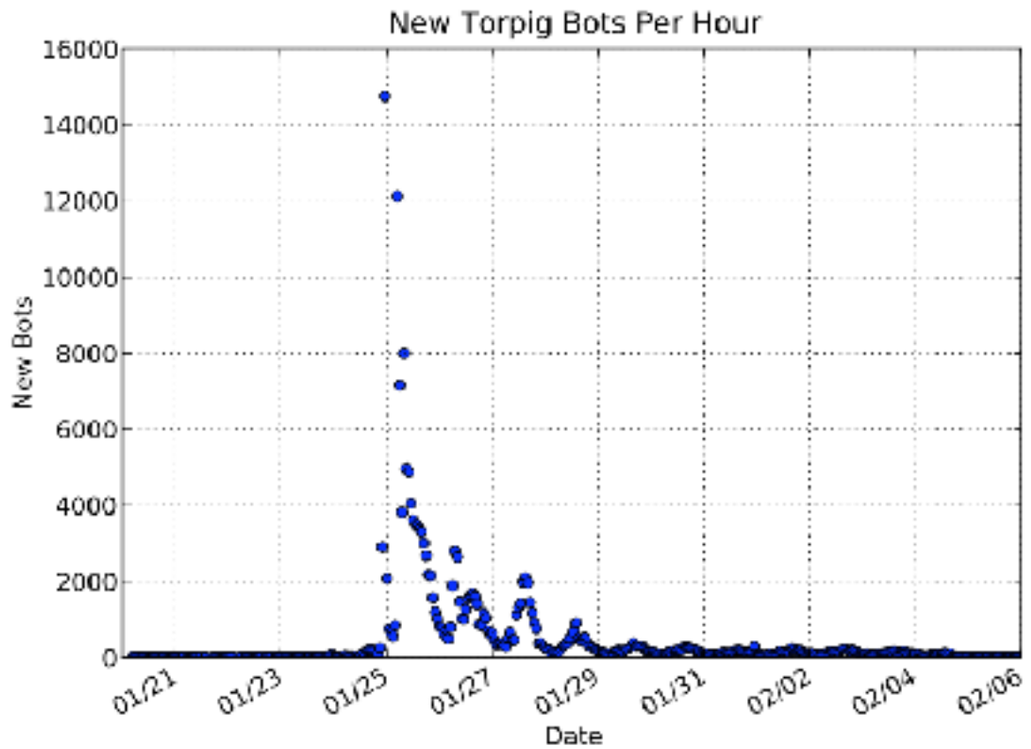


Figure 6: New bots per hour.

BOTNET SIZE: HOW TO COUNT?

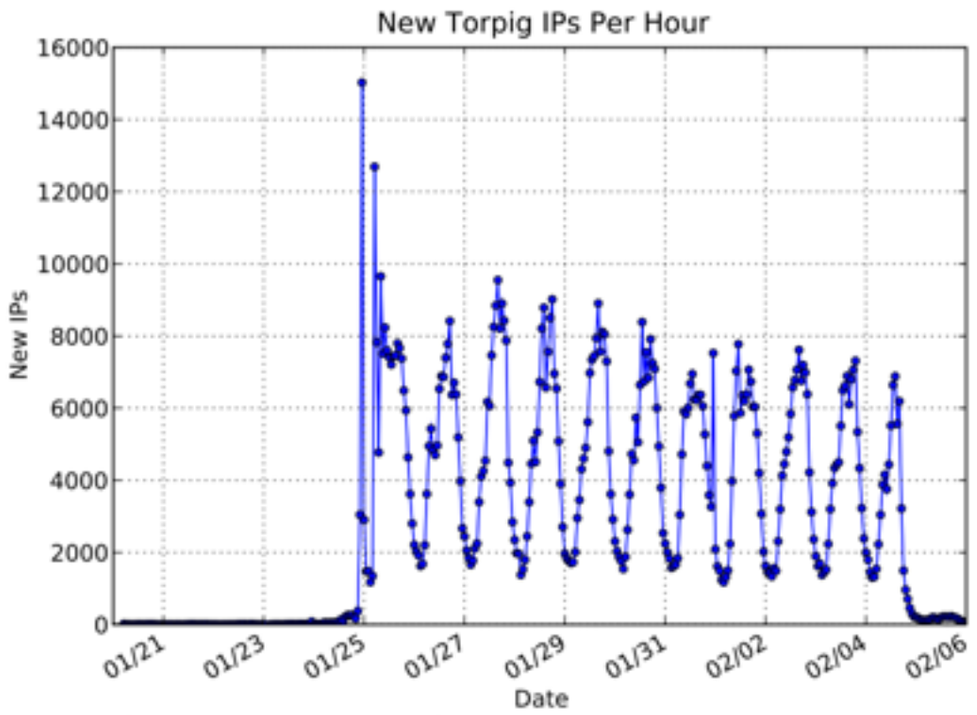


Figure 5: New unique IP addresses per hour.

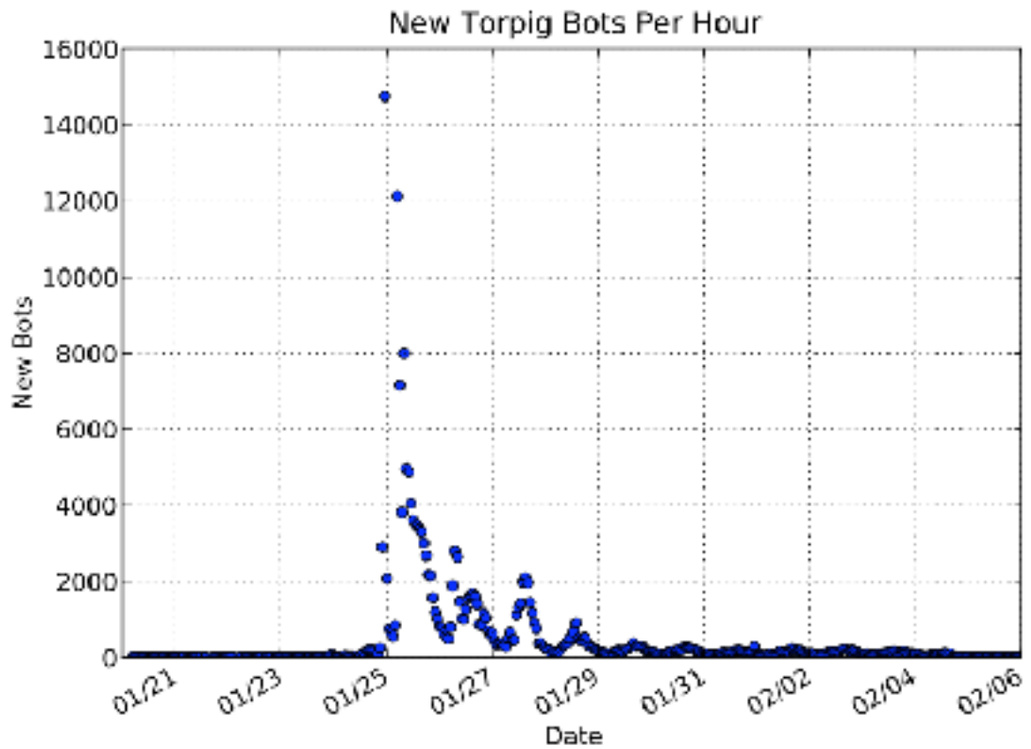


Figure 6: New bots per hour.

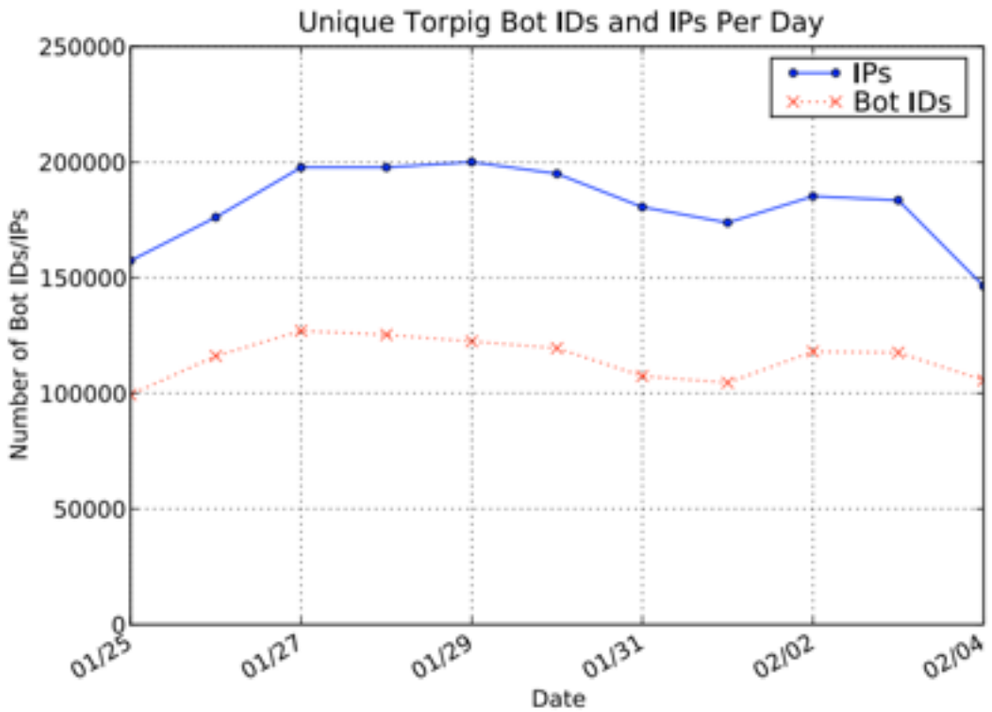


Figure 10: Unique Bot IDs and IP addresses per day.

BOTNET SIZE: HOW TO COUNT?

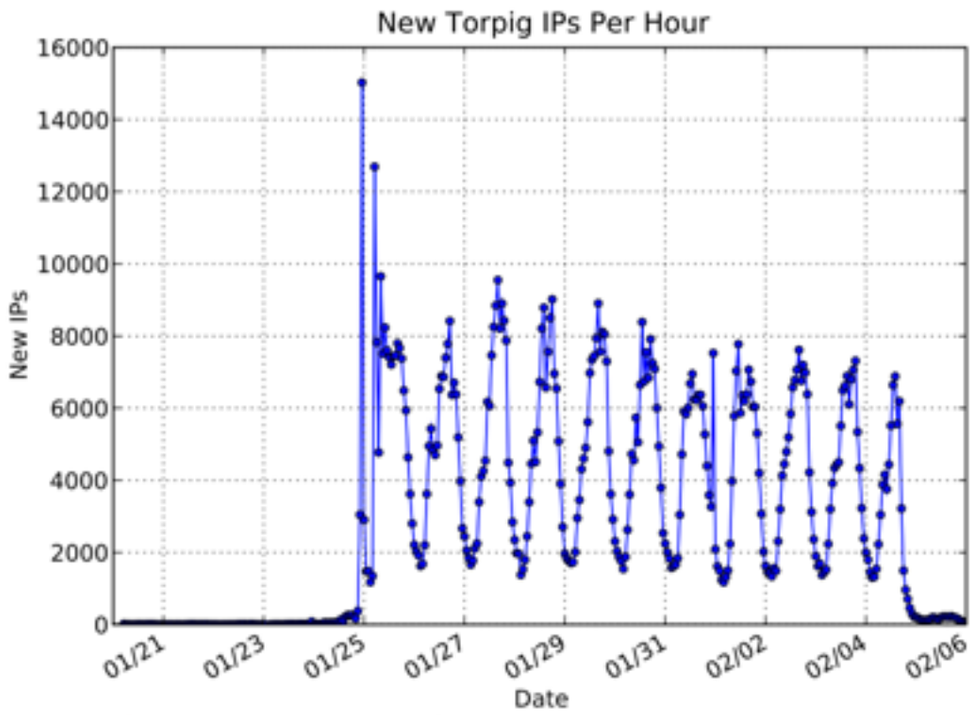


Figure 5: New unique IP addresses per hour.

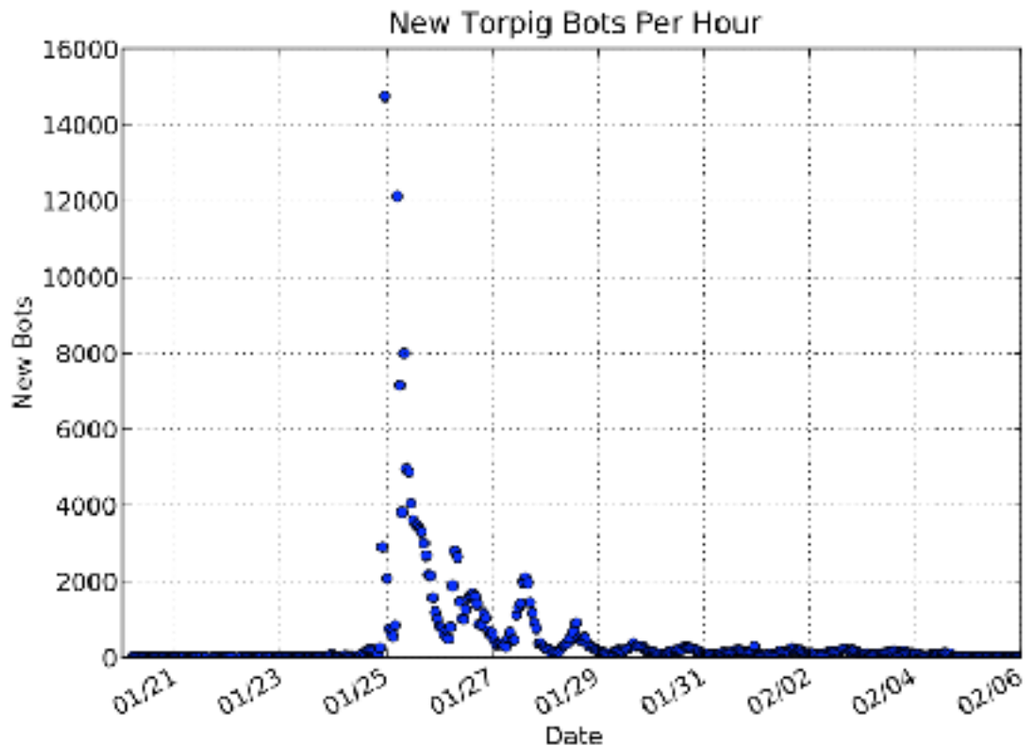


Figure 6: New bots per hour.

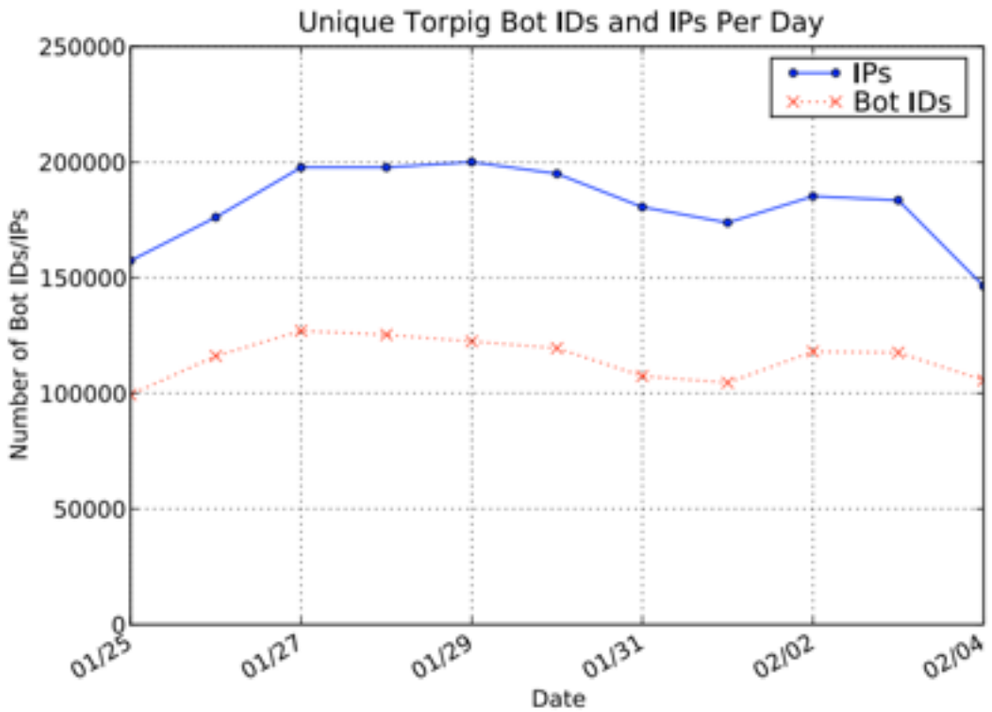


Figure 10: Unique Bot IDs and IP addresses per day.

Country	IP Addresses (Raw #)	Bot IDs	DHCP Churn Factor
US	158,209	54,627	2.90
IT	383,077	46,508	8.24
DE	325,816	24,413	13.35
PL	44,117	6,365	6.93
ES	31,745	5,733	5.54
GR	45,809	5,402	8.48
CH	30,706	4,826	6.36
UK	21,465	4,792	4.48
BG	11,240	3,037	3.70
NL	4,073	2,331	1.75
Other	180,070	24,766	7.27
Totals:	1,247,642	182,800	6.83

Table 2: Top 10 infected hosts by country.

IP ADDRESSES ARE POOR IDENTIFIERS

NAT boxes:

Small set of public IP addresses (typically one),

Large set of private IP addresses (many)

Carrier-grade NATs (CGNATS):

NATs at a regional/national level

A single host can have a different IP address for each connection

"The trouble with Tor"

Tor exit nodes also NAT

Destinations cannot (based on IP addr)distinguish between the exit node's traffic and Tor clients' traffic

Cloudflare shows Tor users captchas to differentiate