Insights and Experiences from Monitoring Multiple P2P Botnets

Leon Böck, Shankar Karuppayah, Valentin Sundermann, Max Mühlhäuser, Dave Levin

TECHNISCHE UNIVERSITÄT DARMSTADT

UNIVERSITY OF MARYLAND

UNIVERSITI SAINS MALAYSIA

Monitoring multiple botnets

- Blind men and the elephant
- How (dis-)similar are botnets?
  - Affected devices
  - Behavior over time
  - Affected regions
  - Dynamics

- P2Pwned – Rossow et. al[1]
- Long Term Tracking and Characterization of P2P Botnet [2]
- What has changed with new IoT botnets?

[1] Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian J. Dietrich, Herbert Bos:
SoK: P2PWNED - Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. IEEE Symposium on Security and Privacy 2013: 97-111

[2] Jia Yan, Lingyun Ying, Yi Yang, Purui Su, Dengguo Feng:
Long Term Tracking and Characterization of P2P Botnet. TrustCom 2014: 244-251

Challenges and Motivation

Human Effort

- Reverse Engineering
- Implementing Crawlers
- Data analysis

Resource constraints and requirements

- Size and number of botnets
- Anti-monitoring mechanisms
- (Local) network limitations

Challenges and Motivation

Human Effort

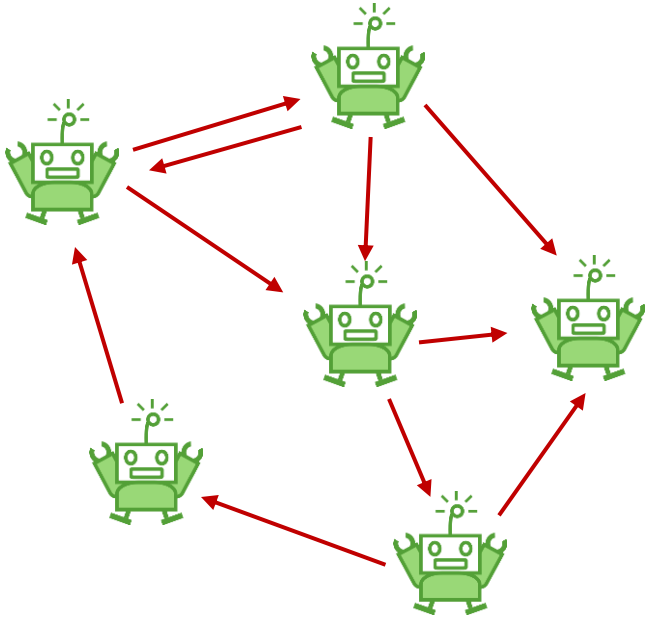
- Reverse Engineering
- Implementing Crawlers
- Data analysis

Resource requirements

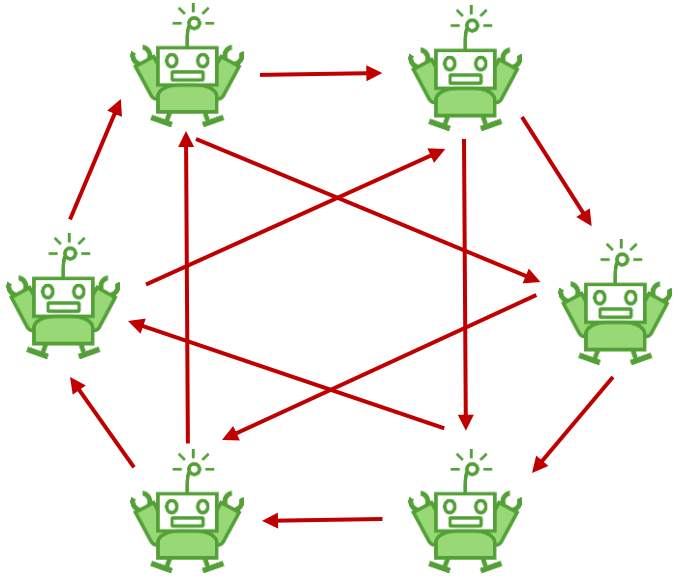
- Size and number of botnets
- Anti-monitoring mechanisms
- (Local) network limitations

P2P Botnets

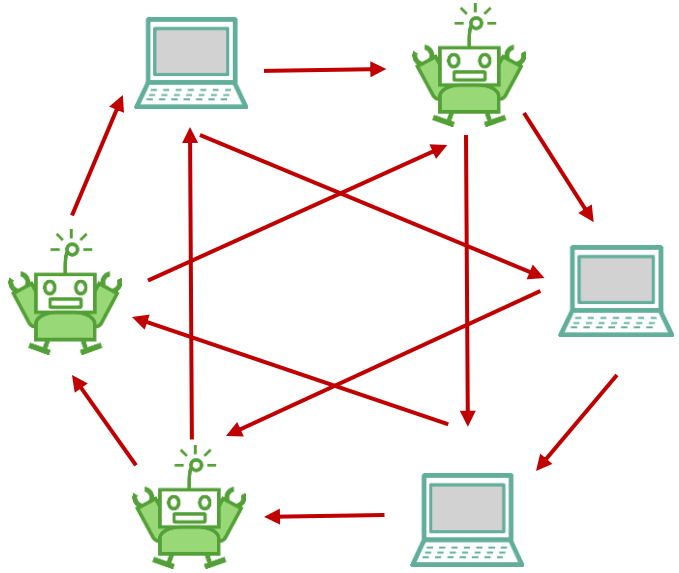
Unstructured



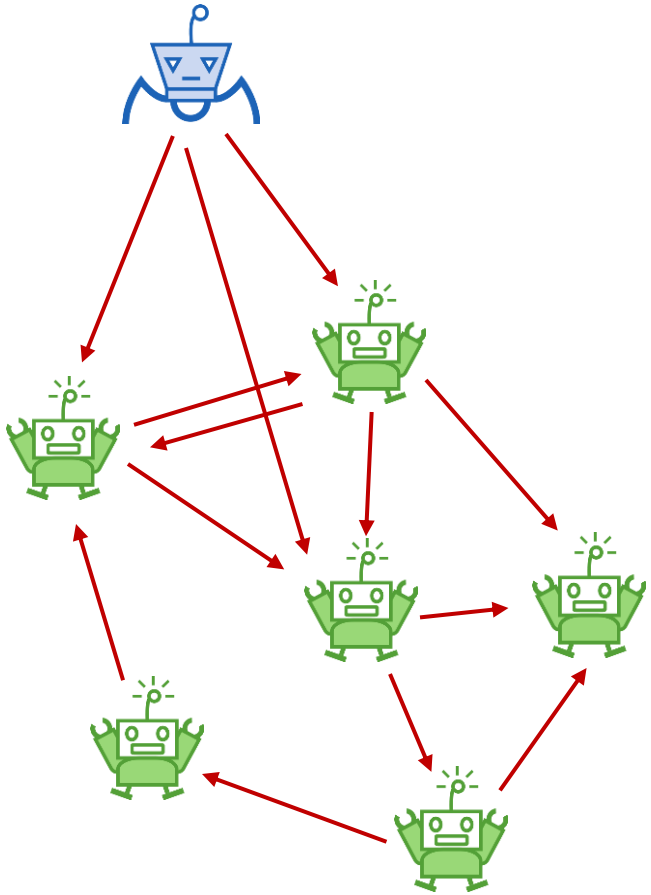
Structured

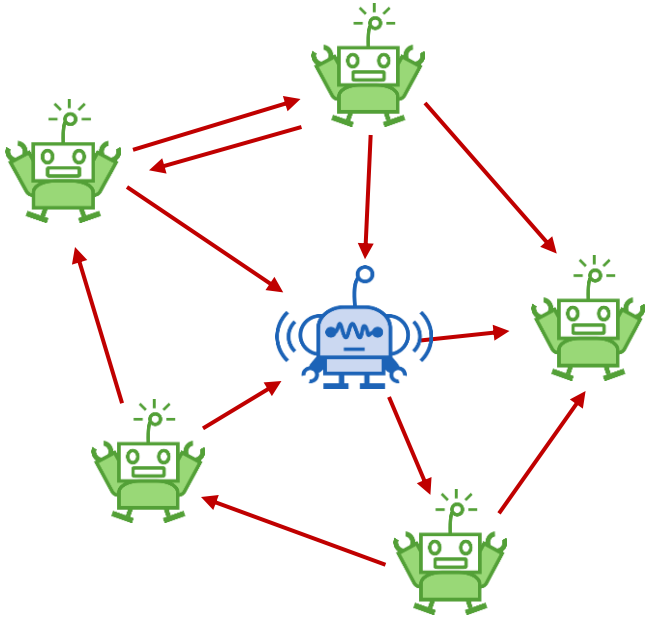


Parasitic

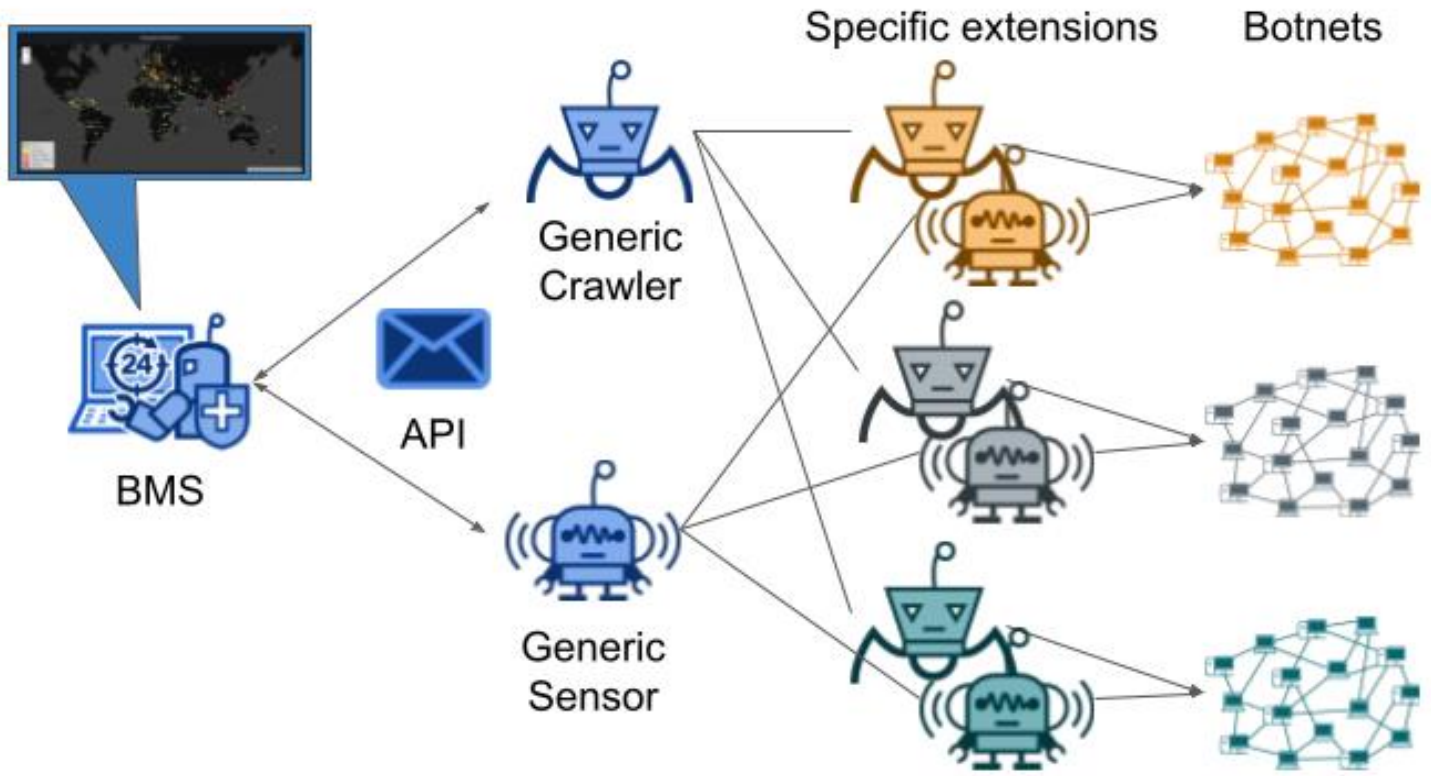


Crawlers and Sensors



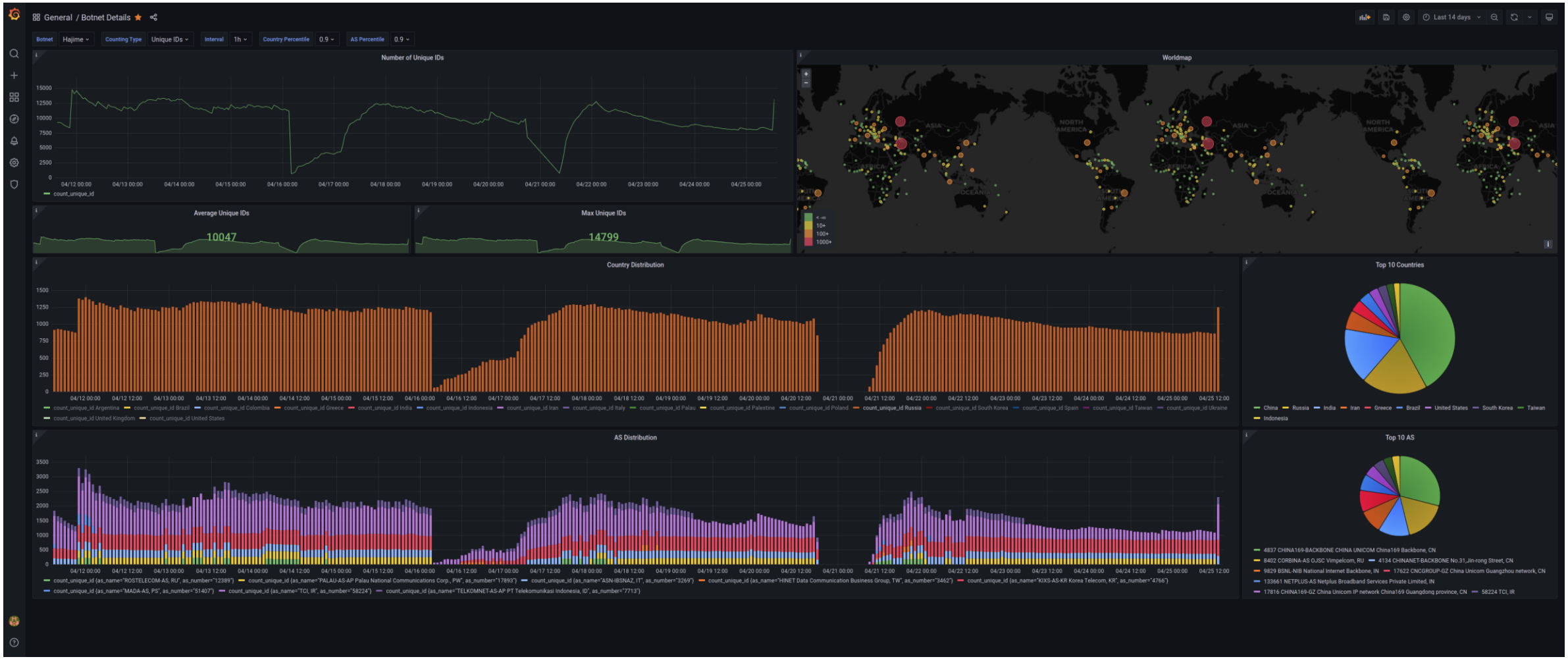


Botnet Monitoring System

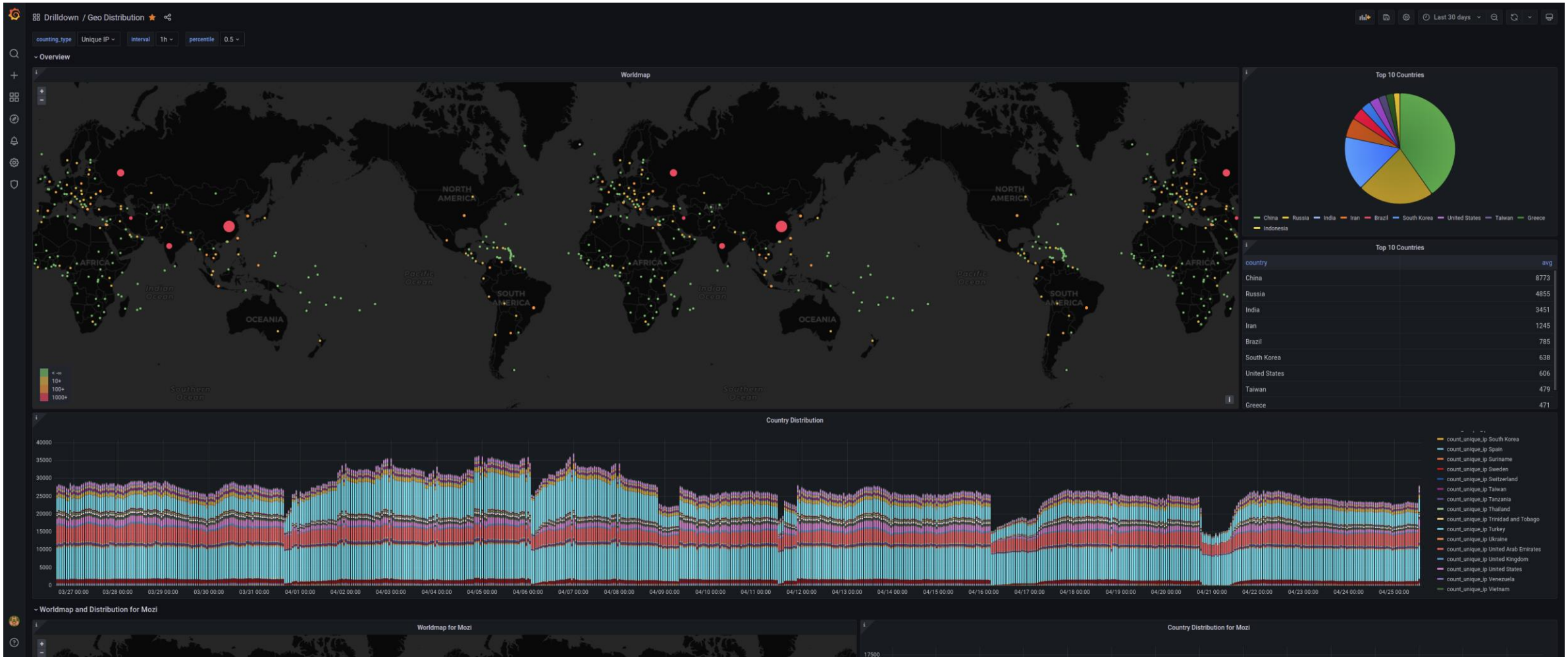


Live Preview – Overview

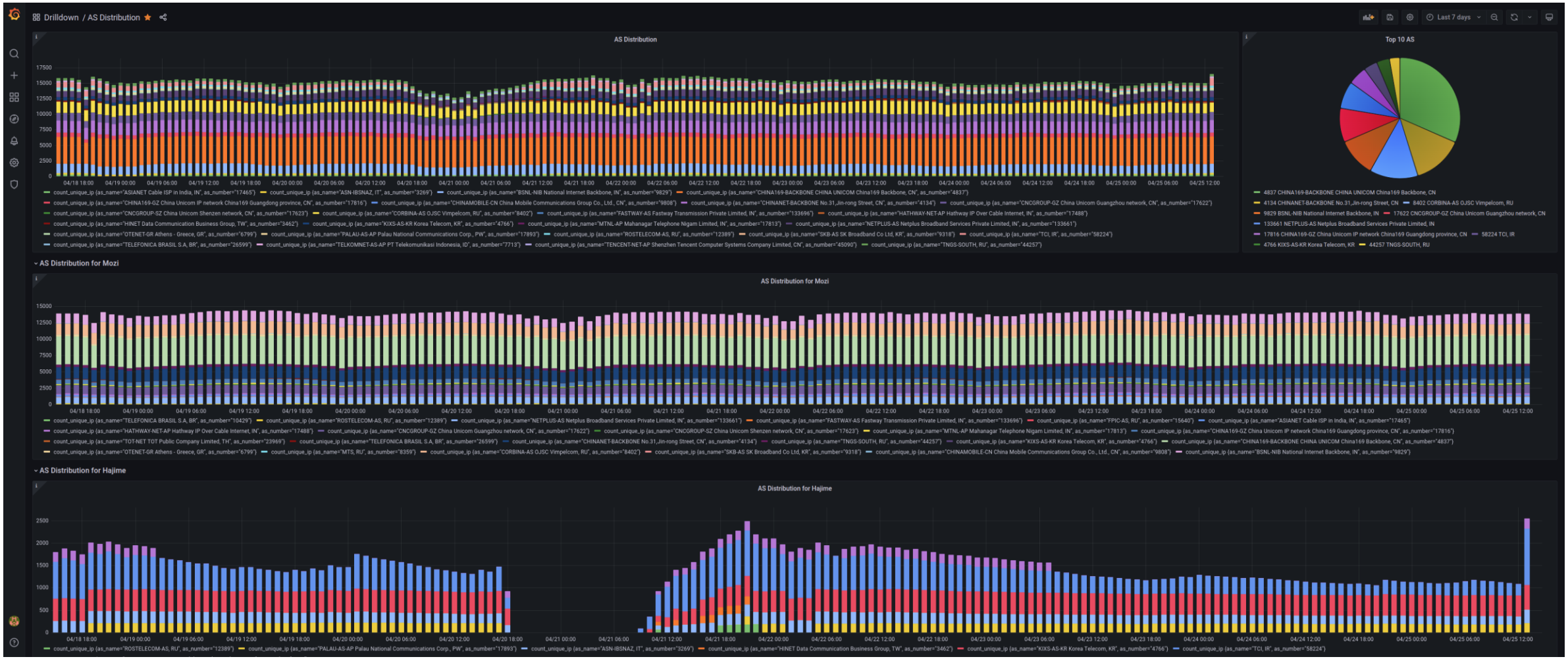
Live Preview – Single Botnet



Live Preview – Geolocations

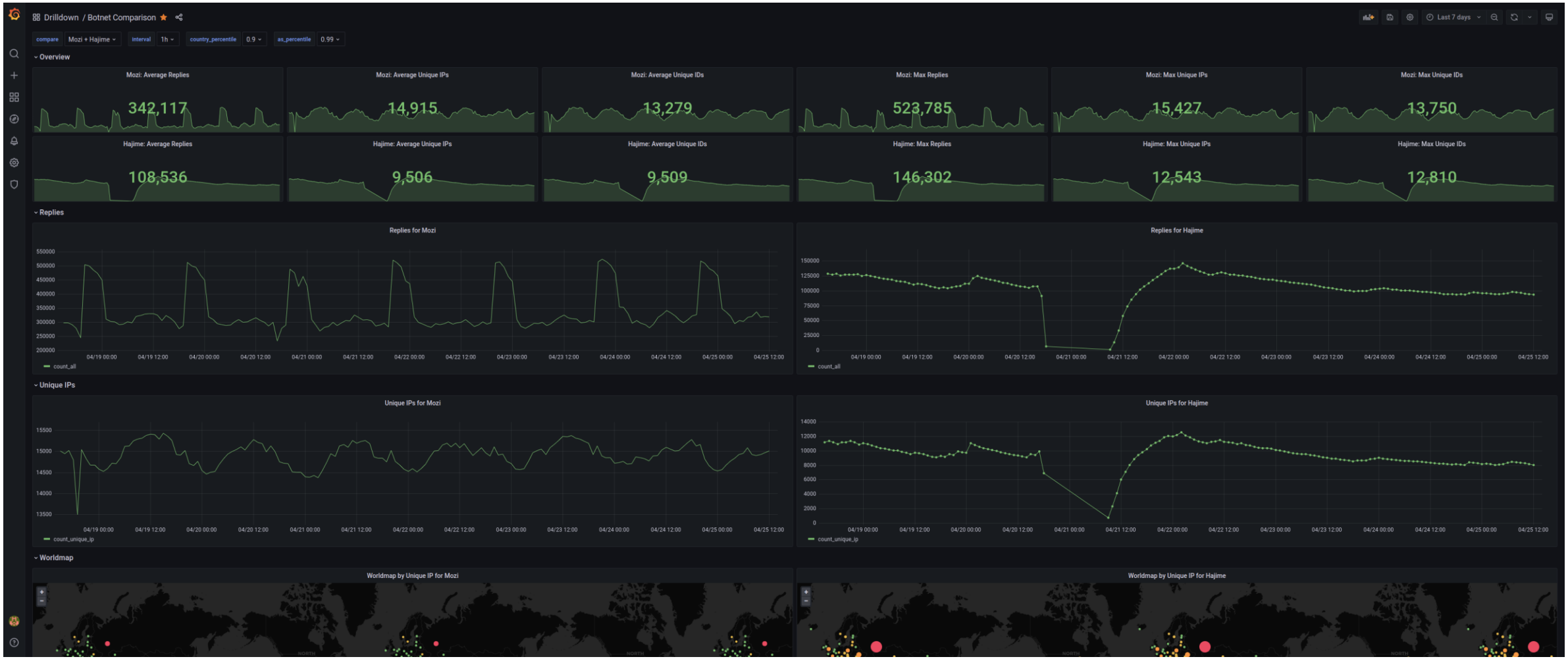


Live Preview – ASes

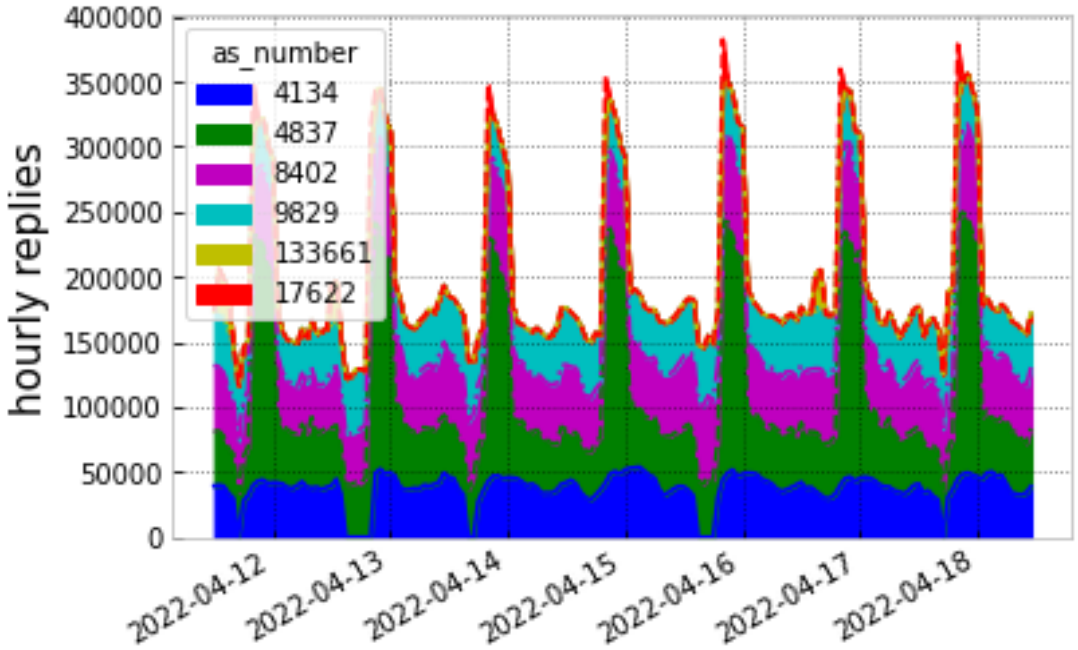


Live Preview – Counting Comparison

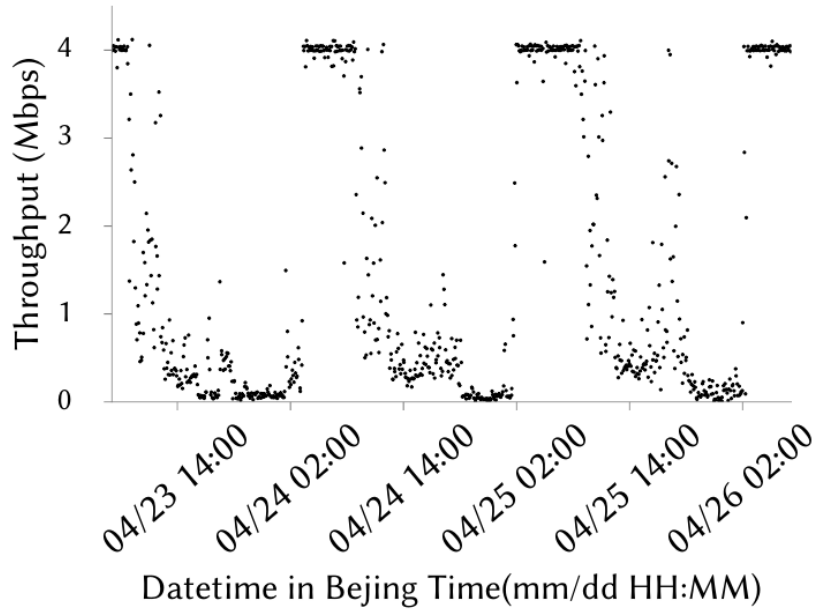
Live Preview – Botnet Comparison



Anomalies in Mozi – Diurnals or Throttling?

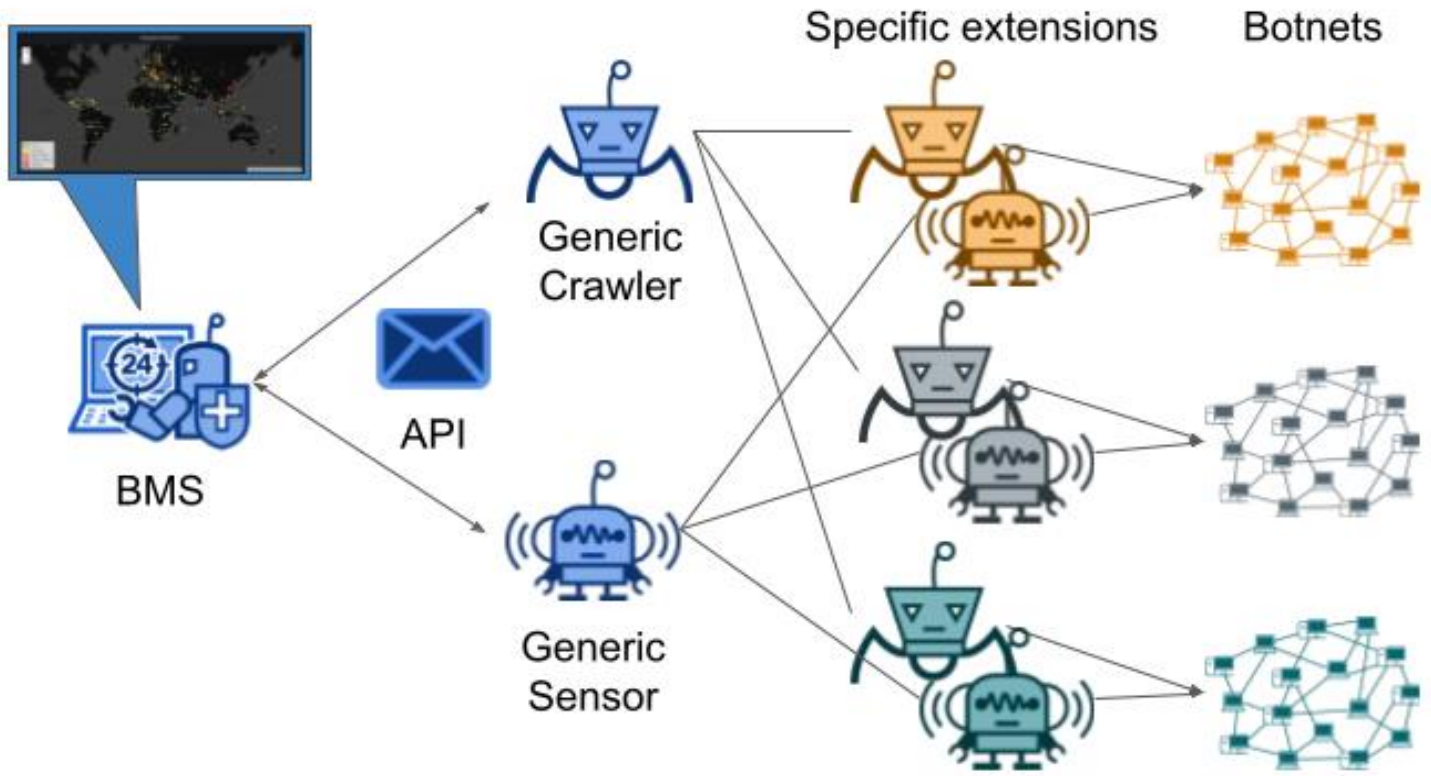


Zhu et. al *



* Pengxiong Zhu, Keyu Man, Zhongjie Wang, Zhiyun Qian, Roya Ensafi, J. Alex Halderman, Hai-Xin Duan:
Characterizing Transnational Internet Performance and the Great Bottleneck of China. Proc. ACM Meas. Anal. Comput. Syst. 4(1): 13:1-13:23 (2020)

Botnet Monitoring System



Backend – Database Format

Replies:

- Any kind of interaction with a bot at the specified time
- Crawlers, Sensors, Honeypots, etc.

timestamp	ip	port	id	json
2022-04-...	1.2.3.4	1337	null	null
2022-04-...	4.3.2.1	42	383838..	V389

Edges:

- P2P-specific
- Enables graph analysis
- Crawlers

timestamp	Src_ip	Src_port	Dst_ip	Dst_port	Src_id	Dst_id
2022-04-..	1.1.1.1	1000	2.2.2.2	2000	null	null
2022-04-..	2.2.2.2	2000	3.3.3.3	3000	null	null

Backend – Communication

Simple solution:

- Distributed crawlers with centralized DB

Drawbacks:

- No validation / authentication
- "one way" communication
- No coordination possible
  - Circumvent countermeasures
  - Dynamic resource allocation

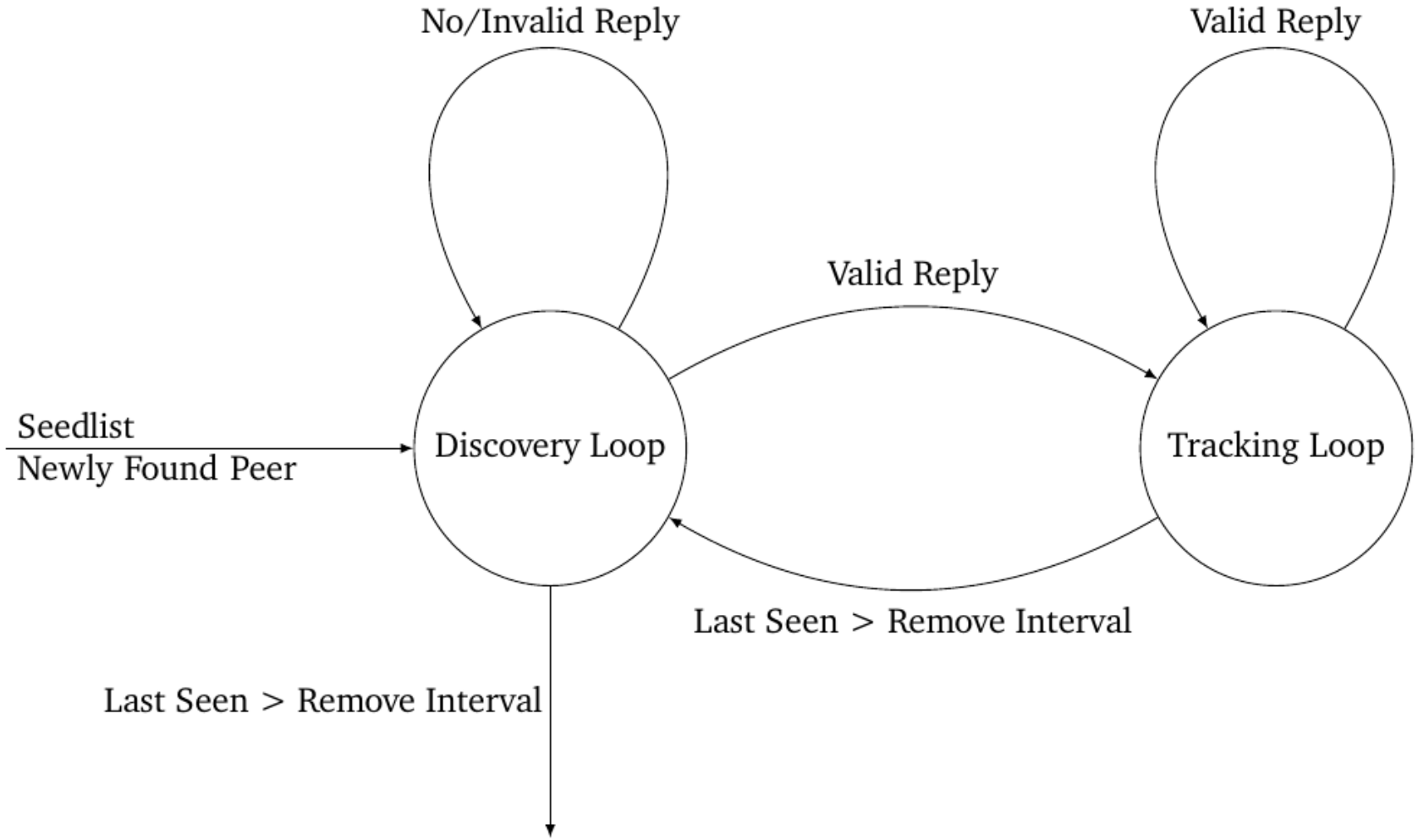
=> Using a custom API

BaseCrawler

Features:

- Backend and API connection
- Queuing and parallelized crawling
  - 20k+ simultaneous connections
- Removal of unresponsive peers
- TCP and UDP
  - Implements custom retry mechanism for UDP
- 2-Queue system to minimize slow down by unresponsive peers

BaseCrawler – Double Queue



BaseCrawler – Example HnS

Most simple case:

2 Functions

- SendPeerRequest
- ReadReply

BaseCrawler calls methods and processes the output

```go
func (ci *HnSProtocol) SendPeerRequest(conn *net.UDPConn, addr *net.UDPAddr, logger *logrus.Logger) {
	msg := []byte("~")

	logger.Debug("Sending to ", addr)
	_, _ = conn.WriteTo(msg, addr) //@Todo verify that readfrom after writeto does not lose packets...
}

func (ci *HnSProtocol) ReadReply(msg []byte, msgLen int, srcAddr *net.UDPAddr, logger *logrus.Logger) (bcrawler.CrawlResult, []string, []bmsclient.DatedEdge, []

	newPeers := []string{}
	edges := []bmsclient.DatedEdge{}
	datedBotReplies := []bmsclient.DatedBotReply{}

	if msg[0] == 94 {
		datedBotReplies = []bmsclient.DatedBotReply{{
			Timestamp: time.Now(),
			BotID:  "",
			IP:     srcAddr.IP,
			Port:   uint16(srcAddr.Port),
		}}

		var ip uint32
		var port uint16
		portRead := bytes.NewReader(msg[2:4])
		ipRead := bytes.NewReader(msg[4:8])
		binary.Read(ipRead, binary.BigEndian, &ip)
		binary.Read(portRead, binary.BigEndian, &port)

		hostStr := net.JoinHostPort(int2ip(ip).String(), strconv.Itoa(int(port)))
		logger.Debug("Received reply: ", srcAddr, " ", hostStr)
		edge := bmsclient.DatedEdge{
			Timestamp: time.Now(),
			SrcBotID:  "",
			SrcIP:     srcAddr.IP,
			SrcPort:   uint16(srcAddr.Port),
			DstBotID:  "",
			DstIP:     net.ParseIP(int2ip(ip).String()),
			DstPort:   uint16(port),
		}
		edges = append(edges, edge)

		newPeer := hostStr
		newPeers = append(newPeers, newPeer)

		return bcrawler.BOT_REPLY, newPeers, edges, datedBotReplies
	} else {
		return bcrawler.NO_REPLY, newPeers, edges, datedBotReplies
	}
}
```

Ongoing and Future Work

- Automated measurement of Churn and lifetimes
  - How long do bots remain active?
- Sensor / Crawler detection
  - Analyze if we can detect ourselves
  - Filter out activities of other researchers
- Coordination
  - Dynamic load allocation
  - Circumventing anti-monitoring mechanisms

Availability

- We are happy to collaborate directly
- Access to source code upon request
- Selected dashboards will be made public soon
- Contact us at: botnets@tk.tu-darmstadt.de

Summary

- Measuring multiple botnets provies unique insights
  - Geographic differences
  - Diurnal patterns
  - IP / ID counts may not be comparable
- BMS enables resource efficient monitoring
- Availability: botnets@tk.tu-darmstadt.de