Tracking and Characterizing Botnets Using Automatically Generated Domains

Hack In The Box
October 17th, 2013

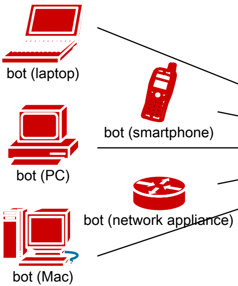
Stefano Zanero
Politecnico di Milano, Italy
@raistolo, stefano.zanero@polimi.it

Federico Maggi, Politecnico di Milano, Italy
Lorenzo Cavallaro, Royal Holloway, University of London, UK
Stefano Schiavoni, Politecnico di Milano, Italy and Google, UK

Introduction

Botnet: Definition

Network of **malware-infected devices** under the control of an external entity.



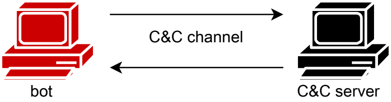
Compromised devices are employed for **malicious purposes**:

information harvesting: login credentials, credit card numbers,

distributed computations: spamming, DDOS attacks.

Command&Control Channel

It is the channel employed for bot-botmaster communications.



It is **logically bidirectional**:

botmaster $\rightarrow$ bot: commands to execute, attacks to launch,

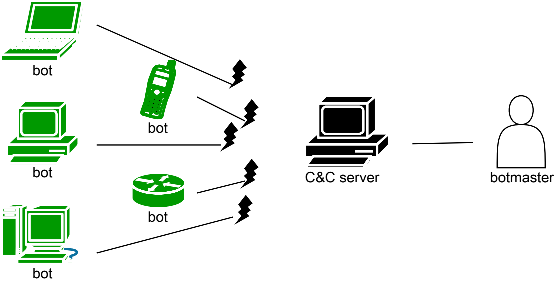
bot $\rightarrow$ botmaster: harvested information, feedbacks.

Single Point of Failure

If bots cannot communicate with their master, they are **innocuous** and **do no produce profit**.

The C&C channel is **single point of failure** of the whole botnet.

Security **defenders strive to disable C&C channels** as means to disable botnets without sanitizing the infected machines.



C&C Channels Security

Botnet architects need to buid ***sinkholing-proof*** C&C infrastructures.

No perfect solution exists, but sinkholing can be made **hard** or **antieconomic**.

Employing **P2P architectures** helps, but these are difficult to manage and provide little guarantees.

Client-server C&C infrastructures can be effective if a **strong rallying mechanism** is employed.

Rallying Mechanisms

Rallying Mechanism: Definition

The process with which a bot looks up for a **rendezvous point** with its master, before starting the actual communication.

The rendezvous point can be:

- an IP address,
- a domain address.

Many mechanisms exist, with different **security properties**.

Hardcoded IP: Functioning

The bot **knows the address** of its botmaster.

Actually, the bot can have **a list of addresses**.

Moreover, it can be instructed to **learn new rendezvous addresses** when necessary, with a **migration-by-delegation**.

Hardcoded IP: Problems

The rendezvous IP is **written in the malware code**: it can be leaked through **reverse engineering**.

If we sinkhole that address:

- the bots **cannot reach their master**,
- the bots are left **without a backup plan**.

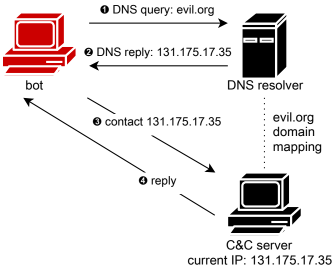
A precise defensive action would **disable the whole botnet**.

Hardcoded Domain: Functioning

The bot resolves a domain `evil.org` and discovers the IP address of the C&C server.

The resulting architecture is **extremely more flexible**.

There is no more vulnerability to IP sinkholing.



Hardcoded Domain: Problems

But actually, **we just moved the single point of failure**: Now it is the domain `evil.org`.

Nevertheless, **sinkholing a domain is much harder** than sinkholing an IP address [Jiang et al. 2012].

General Issues

The aforementioned schemes fail because:

1. **the rendezvous coordinates can be leaked** by the malware binary through reverse engineering;
2. a rendezvous point change needs an **explicit agreement**.

The mechanism of **domain generation algorithms (DGAs)** targets and solves these issues.

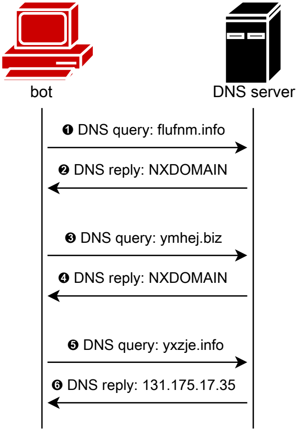
Domain Generation Algorithms

Domain Generation Algorithms: Functioning

Every day the bots generate a **long list of pseudo-random domains**, with an unpredictable seed (e.g., Twitter TT).

The botmaster **registers one of them**.

When the bots find it, **they find the rendezvous point**.



Domain Generation Algorithms: Properties

Malware code is **agnostic**: reverse engineering it is useless.

There is an **asymmetry in the costs and efforts**:

botmaster: needs to register **one domain** to talk to his bots,

defender: needs to register all the **domain pool**, to avoid it.

Migrations of C&C servers **do not need explicit agreement**.

Domain Generation Algorithms: Defense

The DGA mechanism **does not allow proactive defense strategies** and does not have obvious vulnerabilities.

It is necessary to study defensive solutions that allow to **identify and block** DGA-related domains (**AGDs**) timely.

The natural observation point is the **DNS infrastructure**.

State of the Art and Motivation

Domain Reputation Systems

Domain reputation systems exist able to **tell malicious and benign domains apart**.

Some exist that do so by **mining DNS network traffic**, e.g., **Exposure** [Bilge et al. 2011], **Kopis** [Antonakakis et al. 2011], **Notos** [Antonakakis et al. 2010]

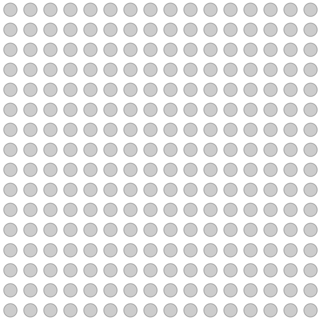
They leverage the fact that malicious domains tend to **exhibit different patterns** with respect to benign domains:

- Behavior over time
- TTL values
- Domain-IP mappings
- ...

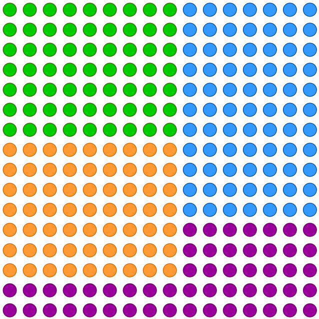
Domain Reputation Systems: Drawbacks I

They **fail in correlating** distinct yet related domains.

256 malicious domains



4 distinct threats



Domain Reputation Systems: Drawbacks II

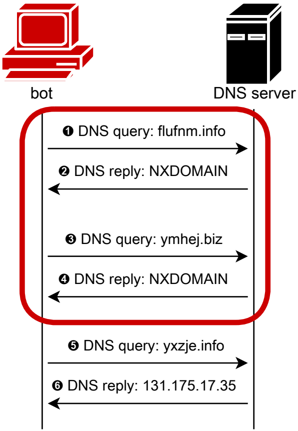
They even fail in providing information about the **specific malicious activity** related to each domain.

- Command&Control of botnets?
- Phishing?
- Drive-by download?

DGA Detection Systems

Detection systems exist that **specifically identify active DGAs** and related domains [Yadav et al. 2010, Yadav and Reddy 2012, Antonakakis et al. 2012].

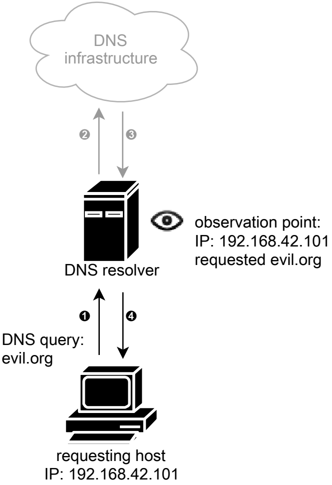
They are driven by the hypothesis that malware-infected machines operating a DGA **generate huge amounts of NXDOMAIN** DNS replies.



DGA Detection Systems: Drawbacks

Nevertheless, they require access to network data that:

- violates users' **privacy**,
- leads to non-repeatable experiments.



Objectives and Challenges

Objectives

Given the limitations of the state-of-the-art systems, we propose **Phoenix**, which:

1. **identifies active DGAs** and the related domains with realistic hypoteses,
2. **correlates the activities of different domains** related to the same DGAs.
3. produces **novel knowledge** and **intelligence insights**.

Challenges

Studying DGAs translates into **analyzing DNS traffic**.

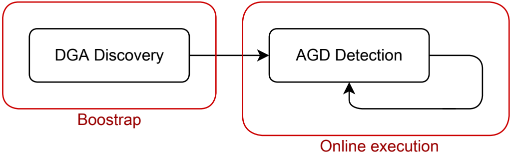
- Where to collect the traffic?
- How to process such **high-volume** and **high-volatility** data?

No ground-truth information is available about DGAs, if not months after they have been employed.

System Description

Overview

Phoenix works in two phases:



DGA Discovery: Discovers **DGAs active in the wild** and characterizes the generation processes.

AGD Detection: Detects **previously-unseen AGDs** and assigns them to a specific DGA.

During its execution, it produces **novel intelligence knowledge**.

DGA Discovery

AGD Filtering: Rationale

AGDs are the result of **randomized computations**.
They look like **"high-entropy" strings**:

`vljiic.org`	`vitgyyizzz.biz`	`79ec8...f57ef.co.cc`
`f0938...772fb.co.cc`	`nlgie.org`	`gkeqr.org`
`jyzirvf.info`	`aawrqv.biz`	`xtknjczaafo.biz`
`hughfgh142.tk`	`yxipat.cn`	`yxzje.info`
`fyivbrl3b0dyf.cn`	`rboed.info`	`ukujhjg11.tk`

We automatize the process of **recognizing the randomness** of domain names.

We do so by computing **linguistic-based features**.

AGD Filtering: Features I

R: percentage of symbols of the domain name d composing **meaningful words**.

For instance:

$d = \texttt{facebook.com}$

$$R(d) = \frac{|\texttt{face}| + |\texttt{book}|}{|\texttt{facebook}|} = 1$$

likely HGD

$d = \texttt{pub03str.info}$

$$R(d) = \frac{|\texttt{pub}|}{|\texttt{pub03str}|} = 0.375.$$

likely AGD

AGD Filtering: Features II

S_n: **popularity** of the n-grams of domain d.

For instance:

$d =$ `facebook.com`

fa	ac	ce	eb	bo	oo	ok
109	343	438	29	118	114	45

mean: $S_2 = 170.8$

likely HGD

$d =$ `aawrqv.com`

aa	aw	wr	rq	qv
4	45	17	0	0

mean: $S_2 = 13.2$

likely AGD

AGD Filtering: Construction

Every domain d is assigned a vector of linguistic features

$$f(d) = [R(d), S_1(d), S_2(d), S_3(d)]^T$$

We compute the values of f for the **100,000 most popular domains** according to Alexa, and we use them as **reference**.

Automatically Generated Domain (AGD)

A domain d' is *automatically generated* when $f(d')$ significantly diverges from the reference.

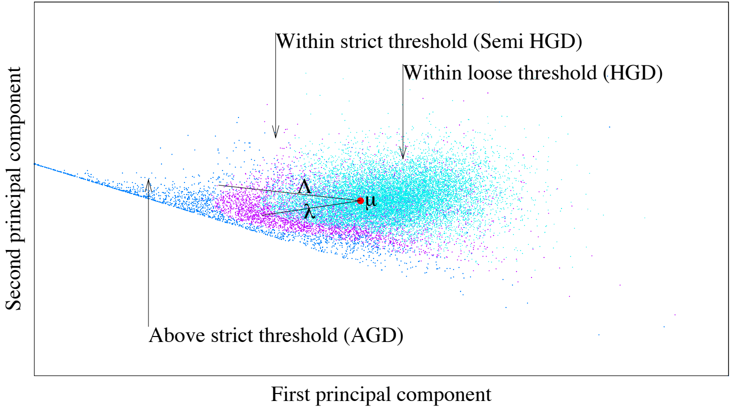
AGD Filtering: Distance and Thresholds Identification I

We define the distance from the reference through the **Mahalanobis distance**.

We set two divergence thresholds $\lambda < \Lambda$, a **strict** and a **loose** one.

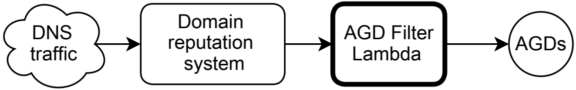
We set the thresholds by **deciding *a priori* the amount of error** we wish to allow.

AGD Filtering: Distance and Thresholds Identification II



Identifying AGDs Between Malicious Domains

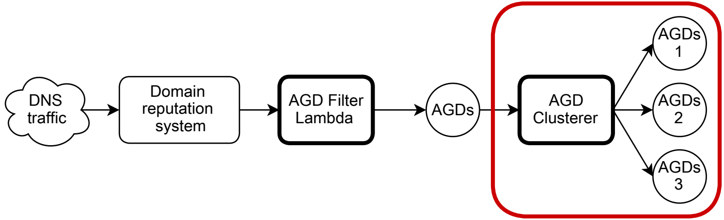
Starting from a *flat* list of malicious domains (e.g., Exposure), we identify those **malicious and automatically generated** (with strict threshold).



These domains are the result of different **generation mechanisms**, and thus have been employed by **different botnets**.

AGD Clustering

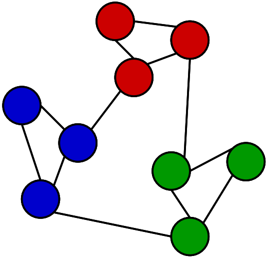
It is possibile to leverage historical DNS network traffic to **cluster together domains employed by the same botnet**.



AGD Clustering: Approach

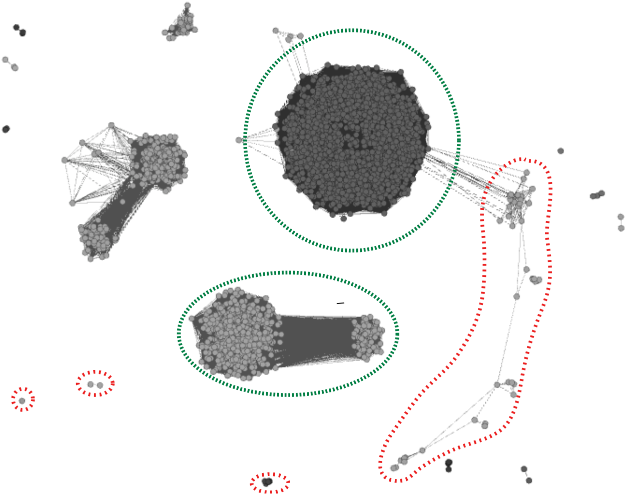
We build a **graph** such that

- every AGD is a node,
- an edge exists if two nodes resolved to the same IP,
- the stronger the peculiarity of the shared IP, the stronger the weight of the edge.



The resulting graph is a **social network**.
We wish to isolate the communities.

AGD Clustering: Example



AGD Fingerprinting

The communities correspond to **families of domains**. Each family corresponds to a generation algorithm.

sbhecmv.tk	sedewe.cn	caftvmvf.org	zsx.net
dughuhg39.tk	lomonosovv.cn	gkeqr.org	vkh.net
dughuhg27.tk	jatokfi.cn	xtknjczaafo.biz	ypr.net
hughfgh142.tk	yxipat.cn	yxzje.info	vqt.org
ukujhjg11.tk	fyivbrl3b0dyf.cn	rboed.info	uon.org

We extract **characterizing fingerprints** from each family:

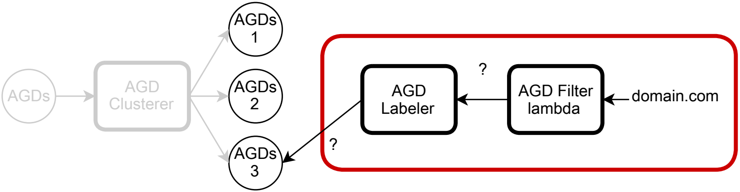
- TLD employed,
- linguistic features (e.g., length, character set),
- C&C IP addresses associated to the botnet.

AGD Detection

Classification of Previously-unseen Domains I

We leverage the fingerprints to **classify previously-unseen domain**, so to extend the blacklist we employed during the bootstrap.

Classification of Previously-unseen Domains II



Given a previously-unseen domain, we answer the questions:

1. does it look like it was **automatically generated** (with loose threshold)?
2. can we associate it with one of the **known domain families**?

If yes, then we found a **new malicious AGD**.

System Evaluation

Approach to Validation

Validating Phoenix is far from trivial, as it **produces novel knowledge**.

For instance, no information is available about the membership of a given malicious domain to one family of AGDs

In **lack** of an established **ground truth**, we:

- run **quantitative tests** to validate each module,
- provide a **qualitative validation** of the whole approach.

DGA Discovery

AGD Filter Evaluation: Dataset

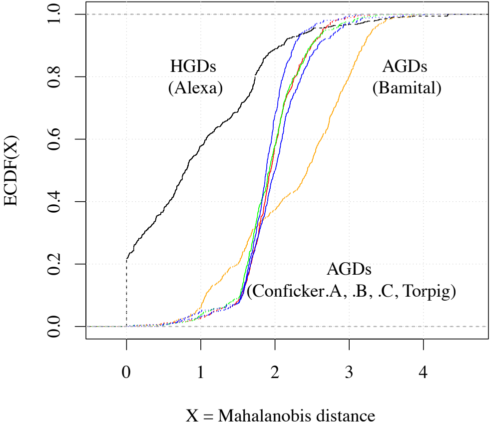
We employ AGDs of **known botnets of the past** to verify the accuracy of the filter.

Specifically, we use the AGDs of:

- Conficker.A (7,500),
- Conficker.B (7,750),
- Conficker.C (1,101,500),
- Torpig (420),
- Bamital (36,346).

AGD Filter Evaluation: Distance ECDF

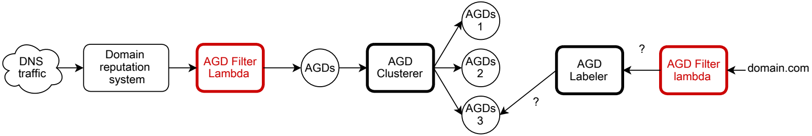
First, we show that the distance from the reference we employed **discriminates well** between HGDs and AGDs.



AGD Filtering Evaluation: Recall

Then, we **validate the recall** of the filter, with both the thresholds.

	$d_{Mah} > \Lambda$	$d_{Mah} > \lambda$
	Pre-clustering selection	Recall
Conficker.A	46.5%	**93.4%**
Conficker.B	47.2%	**93.7%**
Conficker.C	52.9 %	**94.8%**
Torpig	34.2%	**93.0%**
Bamital	62.3%	**81.4%**



AGD Clustering Evaluation

We show that the clustering based on DNS features **partitions well** the AGDs according to **DGA-dependent features** (e.g., TLD, domain length).

We verify the correspondence between the families we isolate and some active botnets: **Conficker**, **Bamital**, **SpyEye**, **Palevo**.

Moreover, we **verify the sensitivity** of the clustering from the **configuration thresholds**, and we evaluate them automatically.

AGD Detection

Detection of Previously-unseen Domains

We feed Phoenix with a **previously-unseen DNS traffic dump**.

We show that it identifies AGDs and associates each of them to a specific family.

Previously-unseen domains		
hy613.cn	5ybdiv.cn	73it.cn
69wan.cn	hy093.cn	08hhwl.cn
hy673.cn	onkx.cn	xmsyt.cn
watdj.cn	dhjy6.cn	algxy.cn

Previously-unseen domains		
dky.com	ejm.com	eko.com
efu.com	elq.com	bqs.com
bec.com	dpl.com	eqy.com
dur.com	bnq.com	ccz.com

Cluster A

pjrn3.cn	3dcyp.cn	x0v7r.cn
0bc3p.cn	hdnx0.cn	9q0kv.cn
5vm53.cn	7ydzr.cn	fyj25.cn
qwr7.cn	xq4ac.cn	ygb55.cn

Cluster B

uon.org	jhg.org	eks.org
mzo.net	zuh.com	bwn.org
zuw.org	ldt.org	lxx.net
ntz.com	cbv.org	iqd.com

Intelligence and Insights

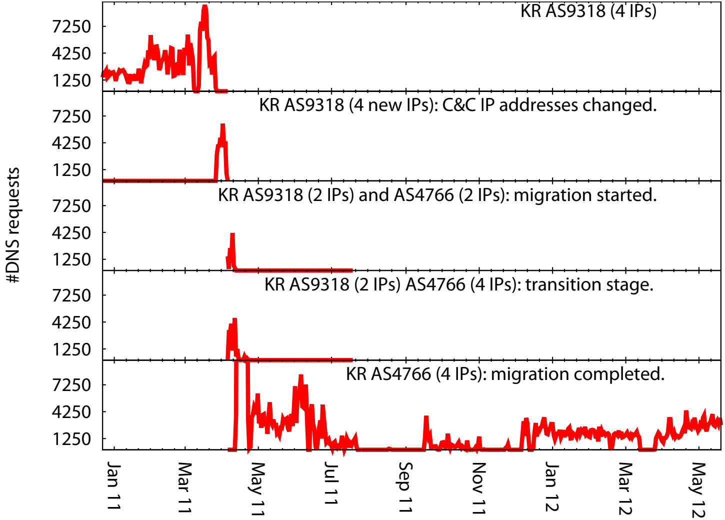
Intelligence and Insights

We produced **novel blacklists of AGDs**.

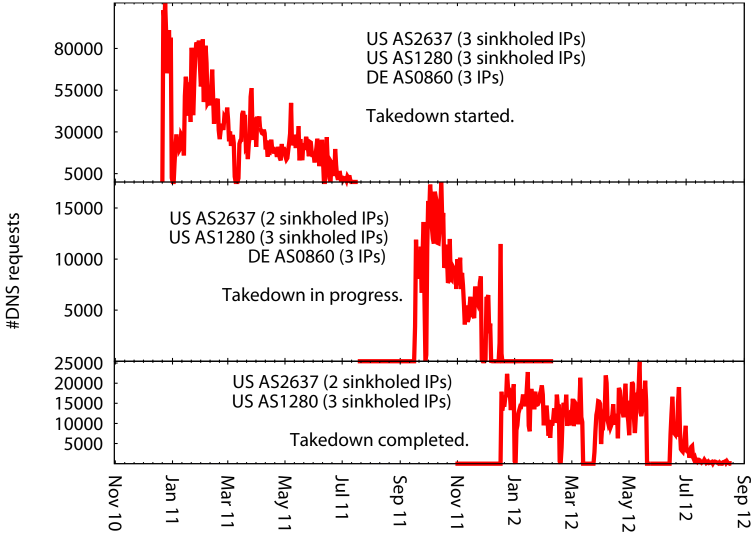
We discovered **C&C servers** employed by each botnet

We processed data in a way which allows us to **follow the evolution of each botnet** over time.

Botnet Evolution Tracking: C&C Migration



Botnet Evolution Tracking: C&C Takedown



Conclusions

Limitations

The AGD Filter of Phoenix assumes to be always dealing with **domains targeting an English-speaking population**.

- Chinese domains? Swedish domains?
- Non-ASCII domains?
  - π`.com`
  - $\clubsuit \rightarrow \heartsuit \rightarrow \spadesuit \rightarrow \diamondsuit \rightarrow$`.com`

Phoenix **may not provide warnings earlier** than similar systems employing NXDOMAIN replies:

- it is fed with data that **take longer to be collected**,
- nevertheless, this makes our system **easier to deploy** and more **privacy-preserving**.

Conclusions

Phoenix gives the following contributions:

1. it **identifies groups of AGDs** between malicious domains and characterizes the generation processes under **more realistic hypoteses** with respect to similar approaches;
2. it **identifies previously-unseen malicious domains** and associates them to the activity of a specific botnet;
3. it produces novel knowledge, which allows—for instance—to **track the evolution of a botnet** over time.

Future Work

Reduce the bias of the AGD Filter from the English language:

- try to **capture the language target** of each domain,
- evaluate its "randomness" according to that language.

Implement an **incremental** version of the clustering algorithm.

Publish our findings and allow users to navigate the data (almost there... :-)

Thank you for your attention. **Questions?**

Let's keep talking on Twitter (@raistolo) or on email (stefano.zanero@polimi.it)

Acknowledgments

The work has been partially funded by the EPSRC-funded project "Mining the Network Behaviour of Bots", under research agreement EP/K033344/1, and by the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257007 "SysSec"

References I

Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster.
Building a dynamic reputation system for dns.
In *Proceedings of the 19th USENIX conference on Security*, pages 18–18. USENIX Association, 2010.

Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon.
Detecting malware domains at the upper DNS hierarchy.
In *Proceedings of the 20th USENIX Security Symposium, USENIX Security*, volume 11, pages 27–27, 2011.

References II

Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon.
From throw-away traffic to bots: detecting the rise of DGA-based malware.
In *USENIX Security '12*. USENIX Association, August 2012.

Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi.
Exposure: Finding malicious domains using passive DNS analysis.
In *Proceedings of NDSS*, 2011.

Jian Jiang, Jinjin Liang, Kang Li, Jun Li, Haixin Duan, and Jianping Wu.
Ghost domain names: Revoked yet still resolvable.
2012.

References III

Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna.
Your botnet is my botnet: Analysis of a botnet takeover.
In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 635–647. ACM, 2009.

Sandeep Yadav and AL Narasimha Reddy.
Winning with DNS failures: Strategies for faster botnet detection.
Security and Privacy in Communication Networks, pages 446–459, 2012.

References IV

Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan.
Detecting algorithmically generated malicious domain names.
In *Proceedings of the 10th annual conference on Internet measurement*, pages 48–61. ACM, 2010.

Sandeep Yadav, Ashwath Kumar Krishna Reddy, AL Narasimha Reddy, and Supranamaya Ranjan.
Detecting algorithmically generated domain-flux attacks with DNS traffic analysis.
2012.