Social Network-Based Botnet Command-and-Control: Emerging Threats and Countermeasures

Erhan J. Kartaltepe[1], Jose Andre Morales[1],
Shouhuai Xu[2], Ravi Sandhu[1]

[1]Institute for Cyber Security
University of Texas at San Antonio

[2]Department of Computer Science
University of Texas at San Antonio

ACNS'10, Beijing, China

Threats of Botnets

- Can launch many attacks
- Including against crypto --- putting trustworthiness of cryptographic services/utilities in question:
  - Compromising cryptographic keys (without being detected after a long time)
  - Compromising cryptographic functions (oracle accesses)

Part I: Twitter-based Bots & Beyond

Current Generation

Possible Future Generation

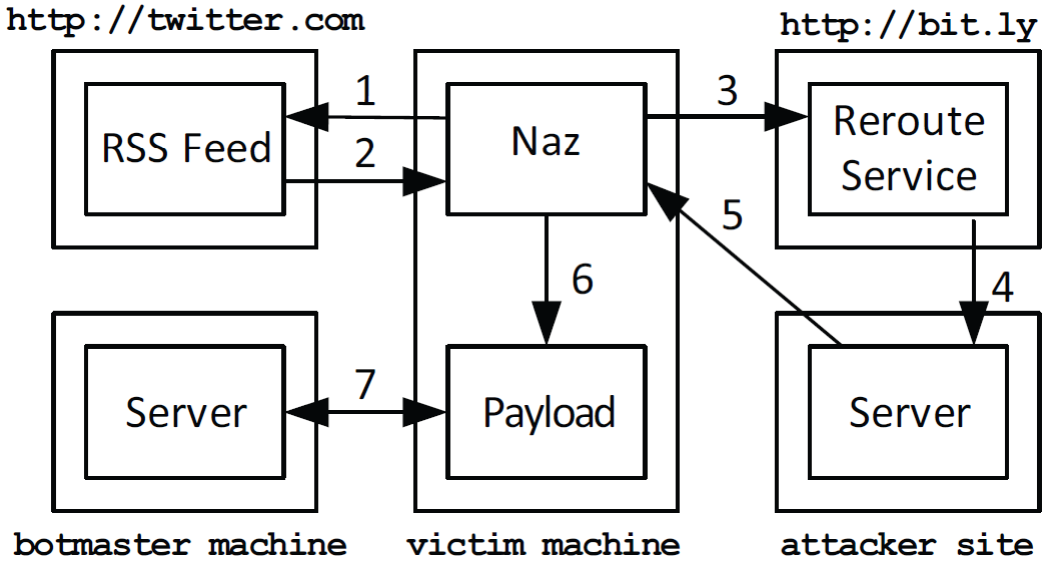
Part II: Defense & limitations

NazBot (TwitterBot) Refresher

- In 2009, Jose Nazario from Arbor Networks accidentally found a bot that used Twitter as its command-and-control (we're nicknaming it NazBot).
- A user ("upd4t3") updated its Twitter account to control NazBot; the bot read the updates via an RSS feed.
- The bot decoded the messages, which were Base64-encoded URLs, and downloaded their malicious payload.
- The payload (gbpm.exe and gbpm.dll) were password and info stealers that did the actual malicious work.

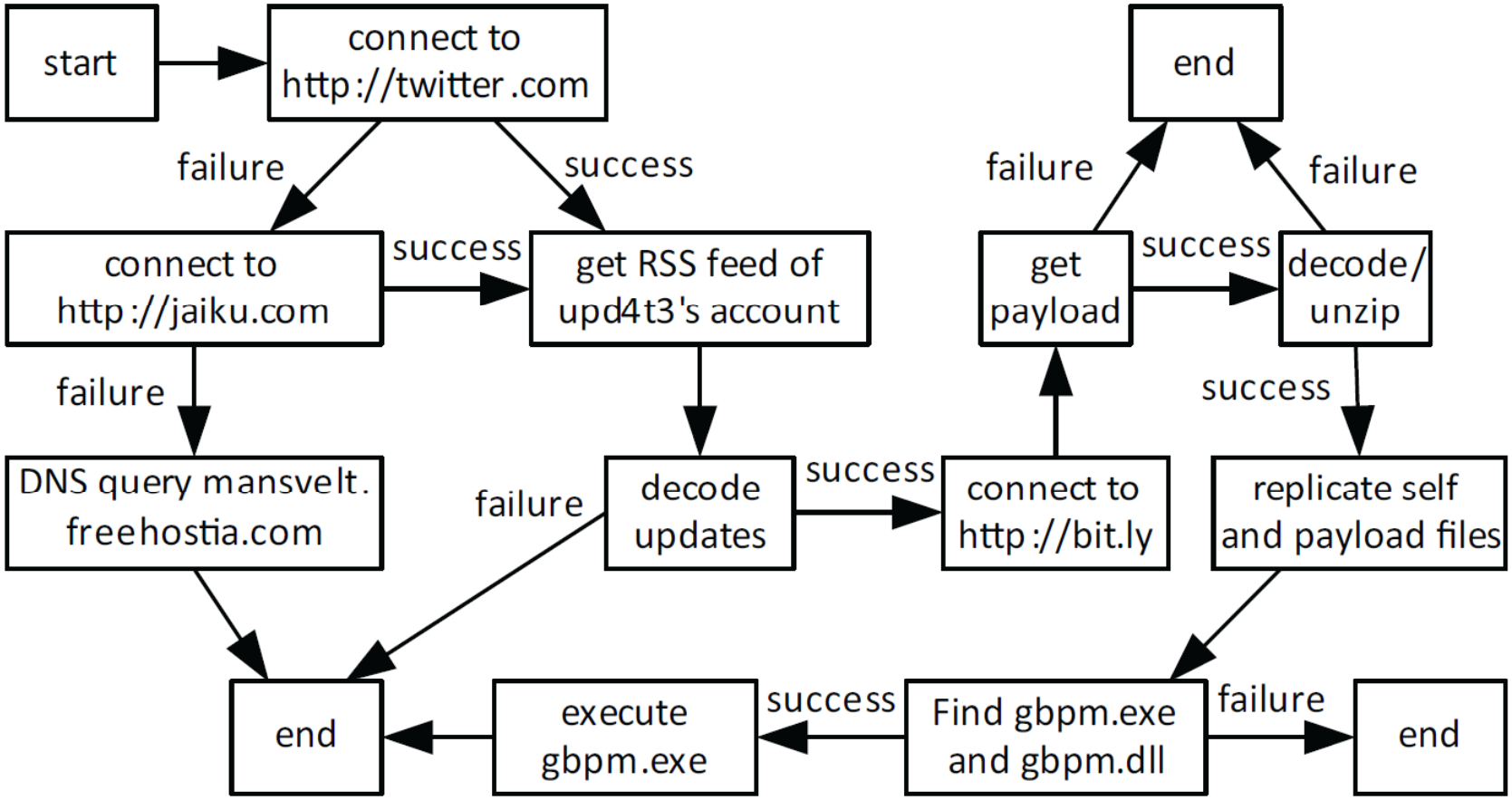
NazBot: Abusing twitter.com

NazBot Flow (controlled experiment)



1. Makes a HTTP GET request to (our fake) upd4t3's Twitter RSS feed.
2. Returns the RSS feed, containing Base64-encoded text.
3. Decodes text as bit.ly URLS (we set up); makes a request to each.
4. Redirects to a malicious zip file on our server.
5. Downloads malicious payload (the real ones).
6. Unzips the payload, copies itself, executes the contents.
7. Gather and transmits victim's information to botmaster.

NazBot Control Flow ("anatomy")



Based on monitored network activities and CWSandbox output.

NazBotnet C&C Strengths

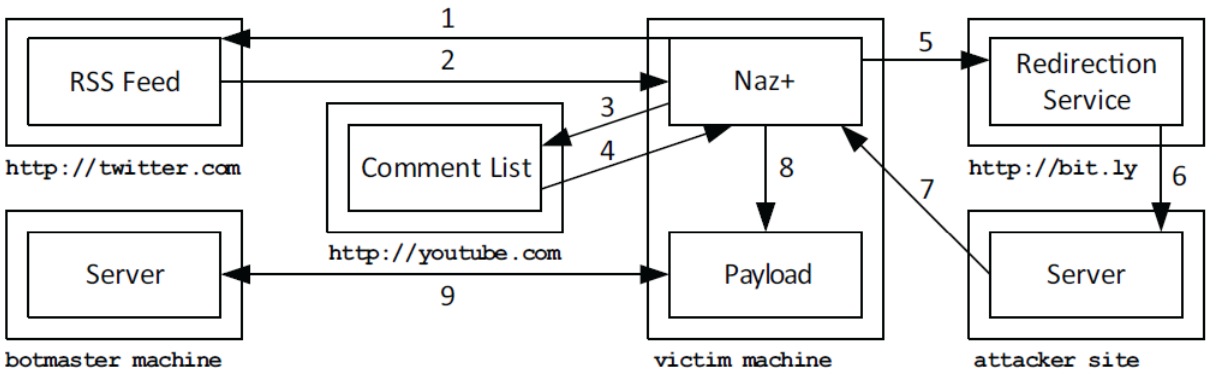
- The Naz Botnet C&C has a number of strengths.
  - Abusing trusted popular websites as a C&C server: Sites such as Twitter, FaceBook, are legitimate and heavily used.
  - Exploiting popular port for C&C communication: Using port 80 with legitimate HTTP requests and responses are not suspicious.
  - Abusing application features for C&C: Common features such as an RSS feeds to auto-update bots are indiscernible from normal traffic.
- The above demonstrates that botmasters have begun to exploit the "hiding in plain sight" approach to conduct stealthy botnet C&C.

NazBotnet C&C Weaknesses

- The Naz Botnet C&C also demonstrated weaknesses (not meant to help the bad guys)
  - The bots only read from RSS feeds:, not Atom, email, etc..
  - The bots read commands from one account on two sites: easy to dismantle the botnets, once detected.
  - The bots used Base64-encoded ASCII: Trivial to recognize/decode.
  - The bots did not use other standard ports: Port 80 is stealthy, but others such as SSL port 443.
- As we discuss, these weaknesses can be avoided in future social network-based, help the defenders look ahead.

Imagine NazBot+

- Imagine a next-gen botnet C&C (call it Naz+).
  - The bots can read commands via any social network-based automatic channel.
  - The bots read commands from any account on any website.
  - The bots can employ other encoding, encryption, and steganography.
  - The bots can read from multiple ports, including SSL port 443.
- Such a bot might have a more complicated flow.



Part I: Twitter-based Bots & Beyond

Part II: Defense & limitations

- Server-side Defense
- Client-side Defense
- Integrated Server-Client Defense

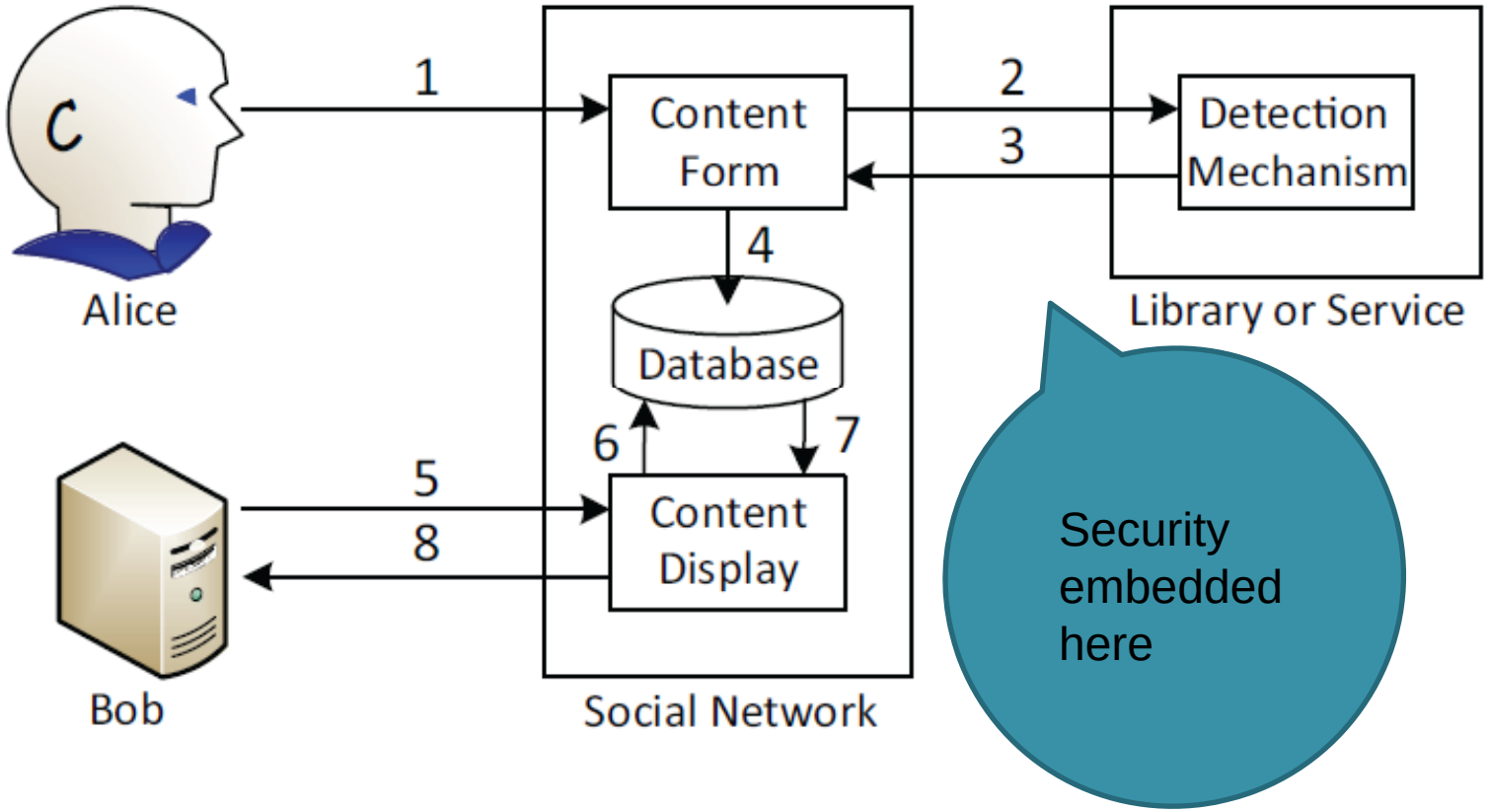
Server-side Defense

- As demonstrated, sites such as Twitter are currently abused to conduct botnet C&C.
- Thus, these servers must defend against both current and future botnets that would abuse them for botnet C&C.
- Observations
  - All social network messages are text, and botmasters must encode their commands textually.
  - Moreover, just like legitimate messages may include web links, so might C&C messages (e.g., links for downloading payload).
- A server-side defense should distinguish between encoded and plain text and to follow links to their destination.

Server-side Defense: Advantages

- Account agnostic: Looks for text attributes that are shared with encoded text rather than individual behavioral patterns.
- Language agnostic: Looks at text for attributes that are shared with encoded text rather than individual words.
- Easy to deploy: Uses light-weight machine learning algorithms and thus deployed as software-as-a-service.
- Web aware: Follows links to determine if the destination is trusted, using SSL authentication (if possible) as a trust infrastructure.

Server-side Defense: Architecture



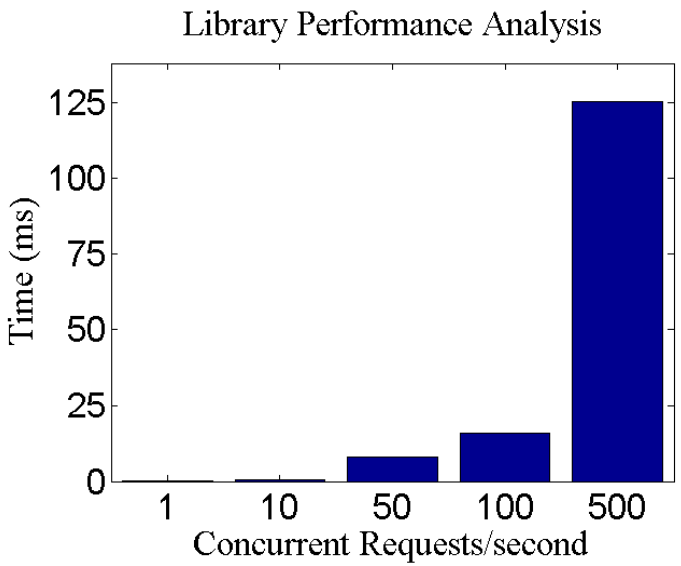
2. Social network's content updater sends the text content to the server-end system.
3. Detection mechanism determines if the text is suspicious.

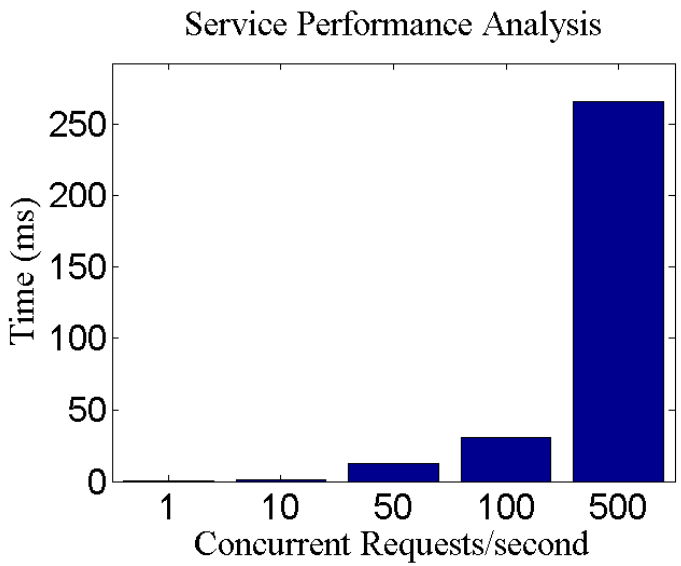
Server-side Testing and Results

- Prototype used Weka's decision tree algorithm to classify Base64/Hex-encoded and natural language text.
- 4000 messages from 200 Twitter accounts built a pool of "non-suspicious" text.
- Our bot commands were 400 encrypted, encoded, random commands.

	Base64		Hexadecimal		Alt. Base64		Alt. Hexadecimal	
	Actual Positive	Actual Negative	Actual Positive	Actual Negative	Actual Positive	Actual Negative	Actual Positive	Actual Negative
Tested Positive	100%	0%	100%	0%	100%	1.25%	96.75%	12.5%
Tested Negative	0%	100%	0%	100%	0%	98.75%	3.25%	87.5%

Server-side Performance





- Twitter's usage analysis is displayed below. Verifying one message daily (and first three for new accounts) check 173.7 mps (increasing 12.1 monthly). Only active and explosive users → 25.6 mps (increasing 2.1 monthly).

15000000 users	Percentage	Update Rate	Messages Per Day	Messages Per Second
Passive users	85.3%	1/day	12795000	148.1
Active users	14.2%	16/day	34080000	394.5
Explosive users	0.5%	1000/day	75000000	868.1
Total users	100%	—	121875000	1410.7

Client-side Defense

- Attributes we look at:
  - Self-Concealing: Attempts to avoid detection with the use of stealth mechanisms (lack of a GUI or HCI).
  - Dubious Network Traffic: Engages in network communication with another machine in a covert or devious way (exclusive social network request, encoded text processing).
  - Unreliable Provenance: Lacks a reliable origin (self-reference replication, dynamic code injection, or unverifiable digital signature of code).
- We classify a process P as being suspicious of being a social network-based bot C&C process if it is either self-concealing or has an unreliable provenance (or both), and engages in dubious network traffic.

Detection Model

- For any process P, we state the following:
- Self Concealing: $(\neg P_{gui}) \wedge (\neg P_{hci}) \rightarrow P_{sc}$
- Dubious Network Traffic: $P_{snr} \wedge (P_{etp} \vee P_{sfd}) \rightarrow P_{dnt}$
- Unreliable Provenance: $(P_{srr} \vee P_{dci}) \wedge (\neg P_{vds}) \rightarrow P_{up}$

- Putting it all together, we classify a process as suspicious of being a Social-Network Based Bot if:

$$(P_{\quad} \vee P_{\quad}) \wedge P_{\quad} \rightarrow P_{\quad}$$

Client-side Detection Results

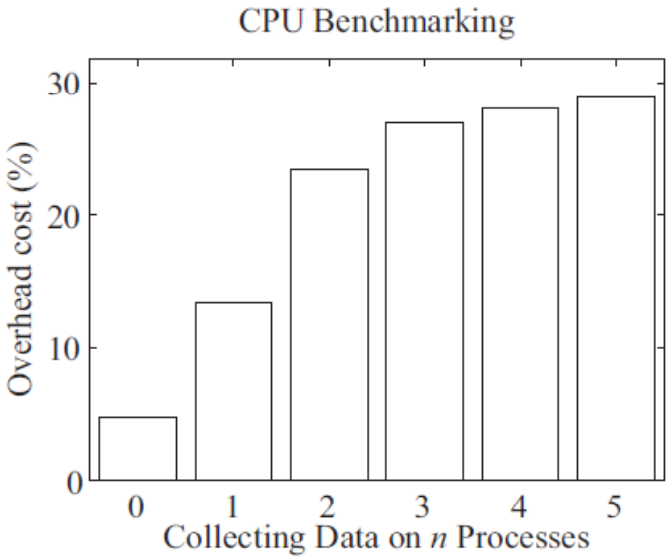
- We used a test set of benign applications, non-social network-based bots, and Naz/Naz+.

Application	Self-Concealing		Unreliable Provenance			Dubious Network Traffic			Result
	Graphical User Interface	Human Computer Interaction	Self-Reference Replication	Dynamic Code Injection	Verifiable Digital Signature	Social Network Request	Encoded Text Processing	Suspicious File Download	Social Network-Based Bot?
AOL Explorer	Y	Y	N	N	Y	N	N	N	N
Avant	Y	Y	N	N	Y	N	N	N	N
BlogBridge	Y	Y	N	N	Y	N	N	N	N
FeedReader	Y	Y	N	N	Y	N	N	N	N
Firefox	Y	Y	N	N	Y	N	N	N	N
Flock	Y	Y	N	N	Y	N	N	N	N
Internet Explorer	Y	Y	N	N	Y	N	N	N	N
Google Chrome	Y	Y	N	N	Y	N	N	N	N
K-Meleon	Y	Y	N	N	Y	N	N	N	N
Maxthon	Y	Y	N	N	Y	N	N	N	N
Mercury	Y	Y	N	N	Y	N	N	N	N
Opera	Y	Y	N	N	Y	N	N	N	N
RSS Bandit	Y	Y	N	N	Y	N	N	N	N
RSS Owl	Y	Y	N	N	Y	N	N	N	N
SeaMonkey	Y	Y	N	N	Y	N	N	N	N
Snarfer	Y	Y	N	N	Y	N	N	N	N
Tweetdeck	Y	Y	N	N	N	Y	N	N	N
Twhirl	Y	Y	N	N	N	Y	N	N	N
Bobax	N	N	Y	Y	N	N	Y	Y	N
Ozdok	N	N	Y	Y	N	N	Y	Y	N
Virut	Y	N	Y	Y	N	N	N	Y	N
Waledac	N	N	Y	Y	N	N	Y	Y	N
Naz	N	N	Y	Y	N	Y	Y	Y	Y
Naz+	N	N	N	N	N	Y	Y	Y	Y

- A limitation is the lack of other social network-based botnet C&Cs analysis, due to their lack of discovery.

Client-side Performance

- Using CPU Mark's PerformanceTest 7.0
- Running the data collector added a 4.8% overhead to the overall system.
- Track one to five processes added between 13.3% and 28.9% overhead.



Integrated Defense

- We can certainly integrate the client-side and the server-side countermeasures
- We have the prototype systems based on this paper (and others) that we plan to put into real-life experiments at some point

Limitations (of Integrated Defense)

- Even when our classifier is utilized by a social network provider and a machine has our client solution installed, using both still has some limitations due to steganography.
  - A bot that reads steganographic commands and can evade our client-side sensors.
  - A bot that reads steganographic commands and masquerades as a benign process.
  - A bot that reads steganographic commands and runs scripts.

Related Work

- Well-recognized approaches (many references):
  - Network-centric
  - Host-centric
- Yet-to-understand approaches:
  - Application-centric (hinted/reiterated by the fact that the twitter.com bots were detected by "digging around" although the concept was mentioned several times years ago)

Conclusions and Future Work

- Our future work includes:
  - implementing the client-side countermeasures as real-time detection systems
  - improving the server-side classifier to detect steganography
  - handling multiple stepping stones in payload redirection
  - porting the client-side countermeasures to other computer and mobile platforms.

Thank You

Questions and comments?