Auto-learning of SMTP TCP Transport-Layer Features for Spam and Abusive Message Detection

Georgios Kakavelakis, Robert Beverly, Joel Young

Center for Measurement and Analysis of Network Data
Naval Postgraduate School, Dept. Computer Science
`{gkakavel,rbeverly,jdyoung}@cmand.org`

December 8, 2011

USENIX LISA 2011

Outline

1. Motivation
2. Detecting Bot-Generated Spam
3. SpamFlow Architecture
4. SpamFlow Results
5. Conclusions

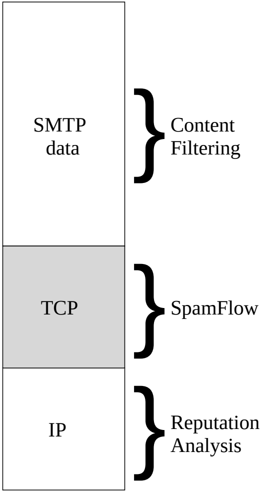
Background

- 2011Q3 MAAWG email metrics: 89% of email is abusive.
- Huge volumes of spam, spammers quickly adapt to defenses.
- Whether user, provider, or vendor, spam is *still* a problem!

Our Prior SpamFlow Work Asked:

- What is the *transport* (TCP/IP packet stream) character of spam?
- Are there *differences* between spam and ham flows?
- How to exploit differences in a way which spammers cannot easily evade?

Understanding SpamFlow



- Not looking at IP header (reputation)
- Not looking at data (conent)
- SpamFlow: TCP stream, incl timing
- FINs, RSTs, Duplicates, OOO pkts, 3WHS timing, packet jitter, receive window, maximum idle time, etc. (20 features in total)

SpamFlow, previous work

“*Exploiting Transport-Level Characteristics of Spam*” [BS08]:

- Utilize statistical machine learning methods
- Offline analysis
- Demonstrate $>$ 90% accuracy, precision, recall (w/o content or reputation!)
- Correctly identify $\simeq$ 78% of false negatives from content filtering alone

Obstacles to Deployment

But ... Obstacles to Deployment:

- Lots of “plumbing,” i.e. exposing transport-features to higher layers
- Must be real-time
- Must be on-line
- Training a supervised learner

USENIX LISA 2011 Contributions:

- Tackle these deployment issues, did the “hard” work
- Built an opensource SpamFlow plugin for SpamAssassin
- (And show performance numbers – it really works!)

Outline

1 Motivation

2 Detecting Bot-Generated Spam

3 SpamFlow Architecture

4 SpamFlow Results

5 Conclusions

Transport-Level Characteristics of Spam

Why does SpamFlow work?

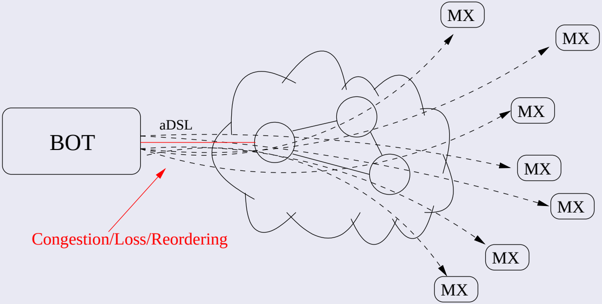
Two Observations on Spam

1. Low Penetration:
   - due to existing filters, user ambivalence
   - → huge volumes of spam
2. Sending Method:
   - Botnets, dialup, etc.
   - → Low asymmetric bandwidth, widely distributed

Transport-Level Characteristics of Spam

Combining Observations: Low Penetration + Sending Methods

Volume + Methods + Economics → link/host resource contention

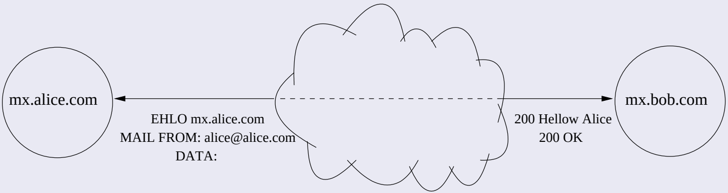


Contention:

Contention manifests as TCP/IP loss, retransmission, reordering, jitter, flow control, etc. Particularly with the large buffers in consumer cable/DSL modems.

SMTP and TCP

Transmission Control Protocol:



- Simple Mail Transport Protocol (SMTP) uses TCP for transport
- Sequence of SMTP commands between Mail Transport Agents (MTAs)
- Mail contents are packetized

How do Spam Connections Behave?

How do Spam Connections Behave?

...or, a quick look at `netstat`

```
RcvQ   SndQ   Local    Foreign Addr                State
0      0      srv:25   92.47.129.89:49014          SYN_RECV
0      0      srv:25   ppp83-237-106-114.:29081    SYN_RECV
0      0      srv:25   88.200.227.123:25068        SYN_RECV
0      0      srv:25   92.47.129.89:49014          SYN_RECV
0      0      srv:25   ppp83-237-106-114.:29084    SYN_RECV
0      0      srv:25   88.200.227.123:25068        SYN_RECV
0      0      srv:25   88.200.227.123:25069        SYN_RECV
0      0      srv:25   88.200.227.123:25070        SYN_RECV
0      0      srv:25   88.200.227.123:25074        SYN_RECV
0      0      srv:25   84.255.150.15:4232          SYN_RECV
0      25     srv:25   222.123.147.41:50282        LAST_ACK
0      28     srv:25   adsl-pool-222.123.:1720     LAST_ACK
0      31     srv:25   222.123.147.41:50152        LAST_ACK
0      15     srv:25   222.123.147.41:50889        LAST_ACK
0      9      srv:25   88.245.3.19:venus           LAST_ACK
0      25     srv:25   78.184.155.70:1854          FIN_WAIT1
0      23     srv:25   190-48-30-225.spe:50920     FIN_WAIT1
0      23     srv:25   dsl.dynamic812132:48154     FIN_WAIT1
0      23     srv:25   ip-85-160-91-16.e:48093     FIN_WAIT1
0      23     srv:25   88.234.141.158:48389        FIN_WAIT1
0      23     srv:25   p5B0FBB5D.dip.t-d:11965     FIN_WAIT1
...
```

How do Spam Connections Behave?

...or, a quick look at `netstat`

```
RcvQ   SndQ   Local    Foreign Addr                State
0      0      srv:25   92.47.129.89:49014          SYN_RECV
0      0      srv:25   ppp83-237-106-114.:29081    SYN_RECV
0      0      srv:25   88.200.2
0      0      srv:25   92.47.12
0      0      srv:25   ppp83-23
0      0      srv:25   88.200.2
0      0      srv:25   88.200.2
0      0      srv:25   88.200.2
0      0      srv:25   88.200.2
0      0      srv:25   84.255.1
0      25     srv:25   222.123.
0      28     srv:25   adsl-poo
0      31     srv:25   222.123.
0      15     srv:25   222.123.
0      9      srv:25   88.245.3
0      25     srv:25   78.184.1
0      23     srv:25   190-48-3
0      23     srv:25   dsl.dyna
0      23     srv:25   ip-85-16
0      23     srv:25   88.234.14
0      23     srv:25   p5B0FBB5D.dip.t-d:11965     FIN_WAIT1
...
```

TCP Stuck in States

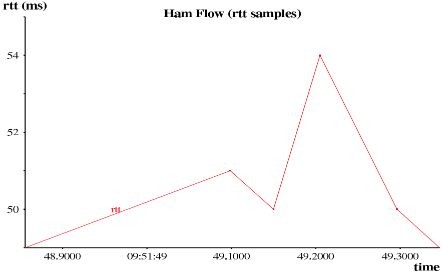
- Stays in these states for minutes
- Half-open connections
- Remote MTAs that “disappear” mid-connection
- Remote MTAs that send `FIN` and disappear

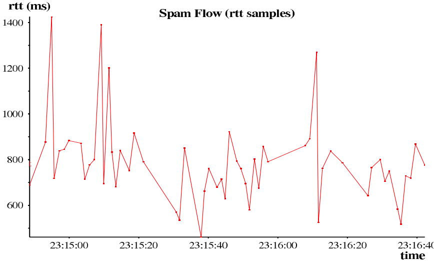
What about RTT?

...building more intuition

```
Received:  from vms044pub.verizon.net
From:  "Dr.  Beverly, MD" <b@ex.com>
Subject:  thoughts
Dear Robert,
I hope you have had a great week!
```

```
Received:  from unknown (59.9.86.75)
From:  Erich Shoemaker <ried@ex.com>
Subject:  Repl1ca for you
A T4g Heuer w4tch is a luxury statement
on its own.
In Prest1ge Repl1cas, any T4g Heuer...
```





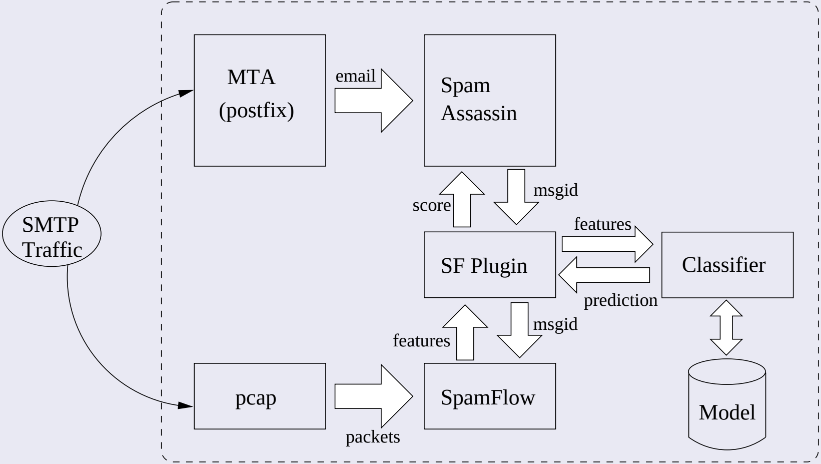
Outline

1. Motivation
2. Detecting Bot-Generated Spam
3. SpamFlow Architecture
4. SpamFlow Results
5. Conclusions

SpamAssassin Plugin

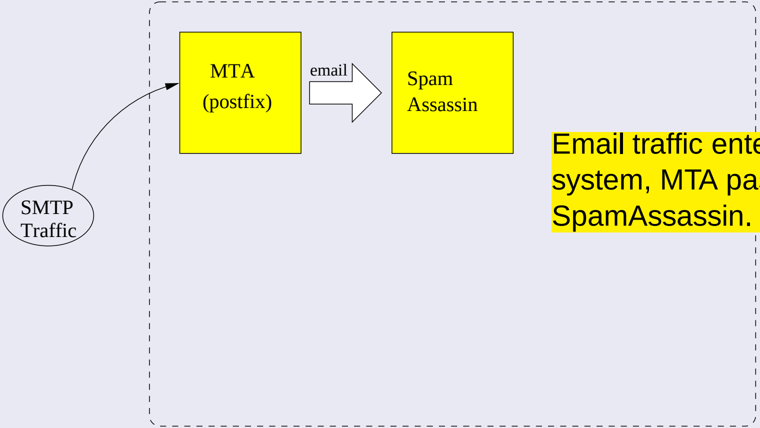
So... we built it.

Moving from research to production:



SpamAssassin Plugin

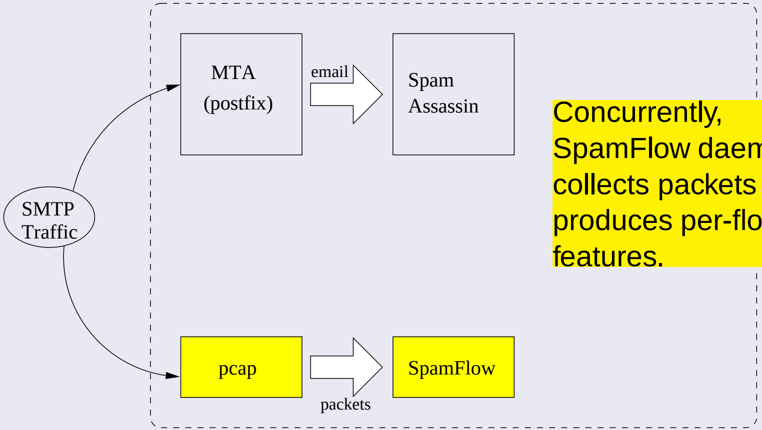
Architecture:



Email traffic enters the system, MTA passes to SpamAssassin.

SpamAssassin Plugin

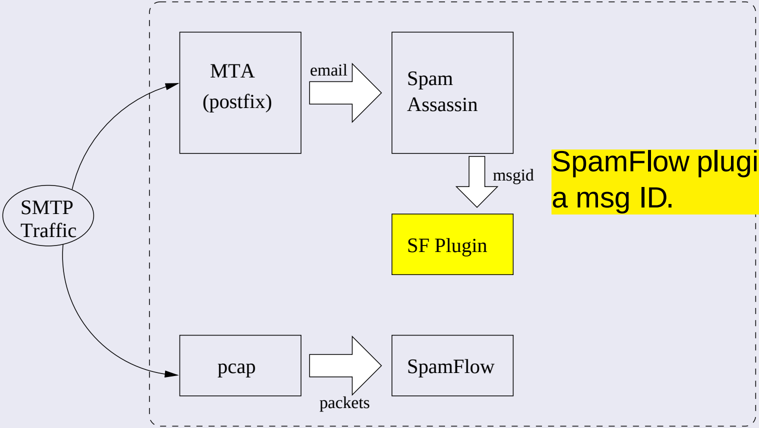
Architecture:



Concurrently, SpamFlow daemon collects packets and produces per-flow features.

SpamAssassin Plugin

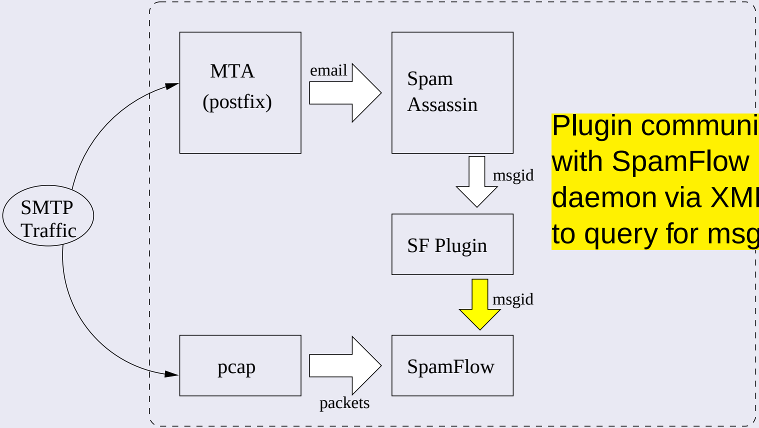
Architecture:



SpamFlow plugin takes a msg ID.

SpamAssassin Plugin

Architecture:



Plugin communicates with SpamFlow daemon via XML-RPC to query for msg ID.

Mapping Traffic Flows to Email

Querying SpamFlow by Message ID:

- SF Plugin queries SpamFlow for traffic features corresponding to an email message
- How to determine which network traffic flow (and its packets) belongs to a given email message?

Mapping Traffic Flows to Email:

- ***Message-ID***: RFC2822, §3.6.4: “Though optional, every message SHOULD have a `Message-ID:` field. The `Message-ID:` field contains a single unique message identifier.”
- ***IP:Port Tuple***: Modify the MTA to record in the email header the ephemeral port of the remote MTA.

Mapping Traffic Flows to Email

Message-ID:

- Not guaranteed to be present
- Requires SpamFlow to perform Deep Packet Inspection
- Increases SpamFlow complexity to reassemble transport stream

IP:Port Tuple:

- Reliable, fast, simple
- Requires trivial change to MTA
- No DPI

SpamFlow:

We use **IP:Port** as the message identifier. Message-ID support planned in next version.

Mapping Traffic Flows to Email

Postfix:

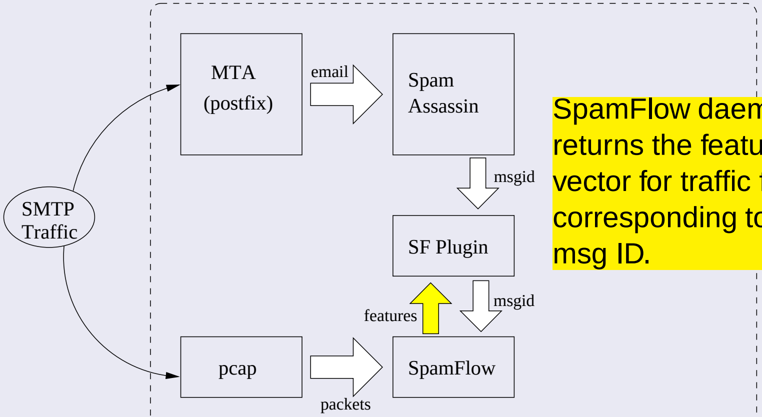
```
--- src/smtpd/smtpd.c.orig
+++ src/smtpd/smtpd.c
@@ -2807,9 +2807,9 @@
 if (!proxy || state->xforward.flags == 0) {
   out_fprintf(out_stream, REC_TYPE_NORM,
-    "Received: from %s (%s [%s])",
+    "Received: from %s (%s [%s:%s])",
     state->helo_name ? state->helo_name : state->name,
-    state->name, state->rfc_addr);
+    state->name, state->rfc_addr, state->port);
```

Qmail:

```
--- received.c.orig
+++ received.c
@@ -44,2 +44,3 @@
+char *remoteport;
 char *remotehost;
@@ -63,2 +64,5 @@
   safeput(qqt,remoteip);
+  remoteport = getenv("TCPREMOTEPORT");
+  qmail_puts(qqt,":");
+  safeput(qqt,remoteport);
   qmail_puts(qqt,")\n  by ");
```

SpamAssassin Plugin

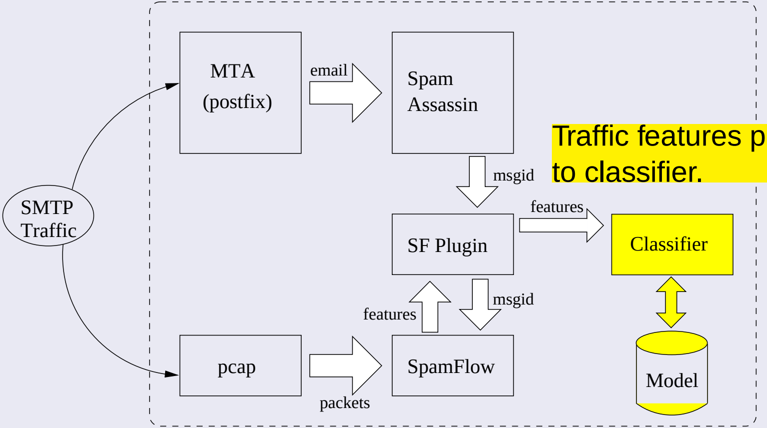
Architecture:



SpamFlow daemon returns the feature vector for traffic flow corresponding to email msg ID.

SpamAssassin Plugin

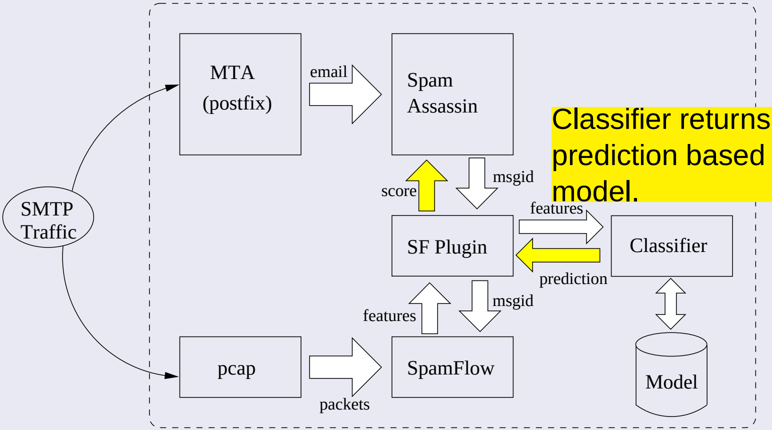
Architecture:



Traffic features passed to classifier.

SpamAssassin Plugin

Architecture:



Classifier returns a prediction based on model.

Example Email

Example Tagged Email:

```
From Josephine@rsi.com Tue Feb 01 23:21:58 2011
Return-Path:  <Josephine@rsi.com>
X-Spam-Checker-Version:  SpamAssassin 3.3.1 (2010-03-16) on ralph.rbeverly.net
X-Spam-Level:  **
X-Spam-Status:  No, score=2.9 required=5.0 tests=BAYES_40,HTML_MESSAGE,SPAMFLOW,
UNPARSEABLE_RELAY autolearn=no version=3.3.1
X-Spam-Spamflow-Tag:  3792891725:37689,12,10,0,0,0,0,1,1,0,53248,34.464852,0.162818,
120.441156,148.297699,51.891697,5840,48,1,64
X-Spam-SpamFlow-Predict:  1
Received:  (qmail 30920 invoked from network); 1 Feb 2011 23:21:57 -0000
Received:  from cm-static-18-226.telekabel.ba (77.239.18.226:37689)
Received:  from vdhvjcvivjvbwyhxnscvfwq (192.168.1.185) by bluebellgroup.com (77.239.18.226)
with Microsoft SMTP
Message-ID: <4D489025.504060@etisbew.com>
Date:  Wed, 2 Feb 2011 00:20:48 +0100
From:  Essie <Essie@hermes.com>
User-Agent:  Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.12)
```

Auto-Learning

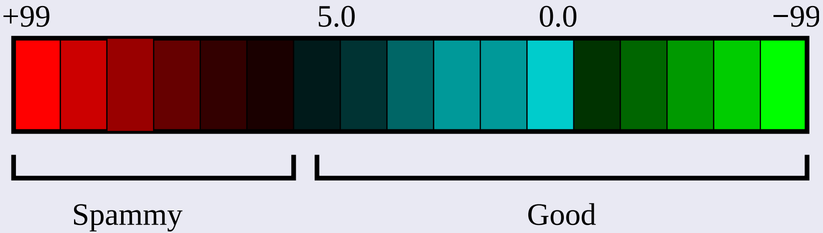
Training:

- Central problem in any supervised learner – how to train?
- Attacks and traffic features evolve
- Every installation environment is different, we observe very different traffic characteristics
- Can't distribute “canned” or ”stock” trained traffic – how to customize per site?

SpamAssassin Scoring

SpamAssassin Scoring:

- Many rules, e.g.
  - In header, subject contains a gappy version of 'cialis': `SUBJECT_DRUG_GAP_C : 2.108 0.989`
  - In body, HTML font color similar to background : `HTML_FONT_LOW_CONTRAST : 0.713 0.001`
- Each rule hit contributes to final continuous message *score*



Auto-Learning

Some messages are clearly spam (hit many rules), or clearly ham (very low score). Two random examples:

Non-Spammy Message (-1.5):

```
X-Spam-Status: No, score=-1.5 required=5.0
tests=BAYES_00,RP_MATCHES_RCVD,
        UNPARSEABLE_RELAY autolearn=ham version=3.3.2
```

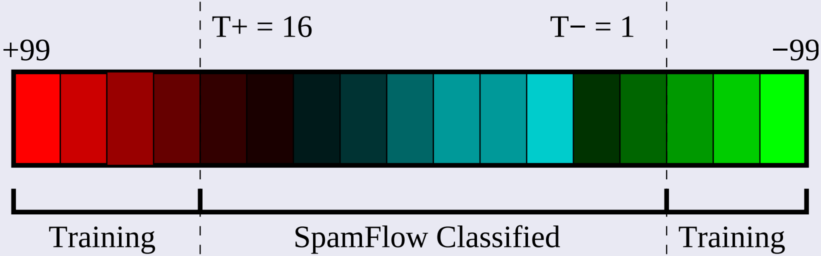
Very Spammy Message (30.8):

```
From: Wellsfargo Internet Banking Alerts!!! <services@wellsfargo.com>
Subject: You Have 1 New Security Message Alerts!!!
X-Spam-Status: Yes, score=30.8 required=5.0
tests=BAYES_50,DATE_IN_PAST_96_XX,
        DOS_OE_TO_MX_IMAGE,FORGED_MUA_OUTLOOK,FORGED_OUTLOOK_HTML,FROM_MISSP_DKIM,
        FROM_MISSP_MSFT,FROM_MISSP_NO_TO,FROM_MISSP_USER,FSL_HELO_NON_FQDN_1,
        HELO_NO_DOMAIN,HTML_MESSAGE,MIME_HTML_ONLY,MISSING_HEADERS,NSL_RCVD_FROM_USER,
        RCVD_IN_BRBL_LASTEXT,RCVD_IN_XBL,RDNS_NONE,SHORT_HELO_AND_INLINE_IMAGE,
        TO_NO_BRKTS_DIRECT,TO_NO_BRKTS_MSFT,UNPARSEABLE_RELAY,
        XMAILER_MIMEOLE_OL_1ECD5 autolearn=no version=3.3.2
```

Auto-Learning

Auto-Learning:

- If other modalities (e.g. keywords, rule tests) indicate strong possibility of spam (high score) or ham (low score), use that as an *training example*
- Incrementally build the model
- Requires *no* human labeling or work!



Outline

1. Motivation
2. Detecting Bot-Generated Spam
3. SpamFlow Architecture
4. SpamFlow Results
5. Conclusions

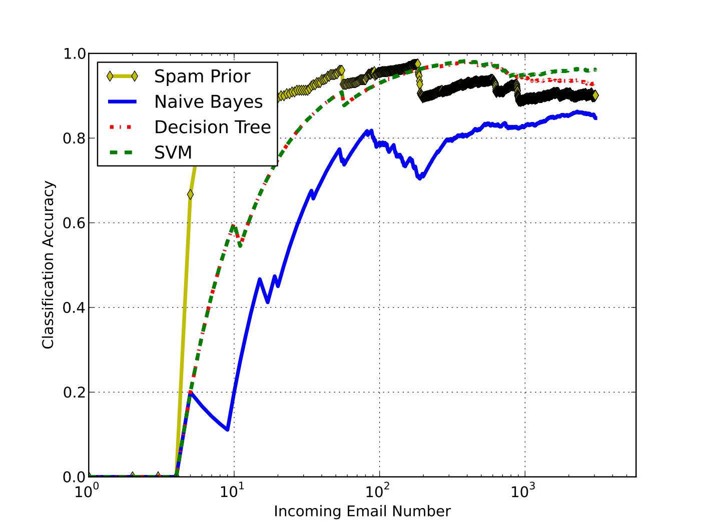
Production Experiments

January-March, 2011:

- Auto-learning thresholds based on spam distribution (normal, $\mu = 16.3, \delta = 7.7$)
- $\tau^{+} = 16$ and $\tau^{-} = 1$
- Yields training of 2,685/5,510 (48.7%) spam and 267/416 (64.2%) ham messages
- Experiments using Naive Bayes, C4.5 decision trees, SVM

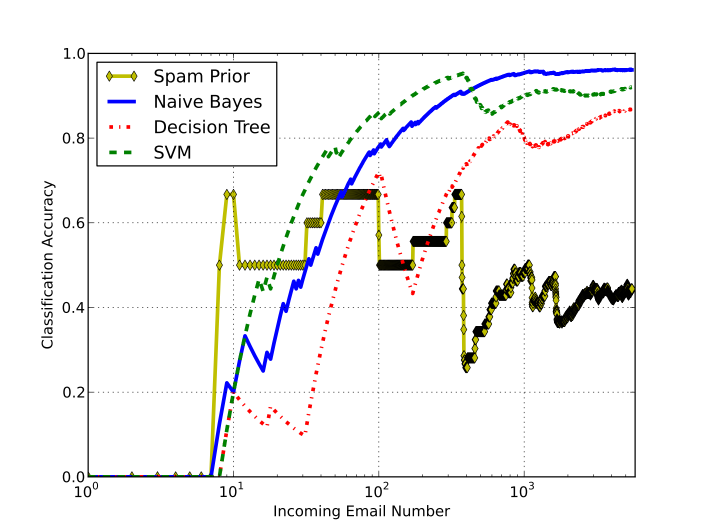
Auto-Learning Performance

Auto-Learning Accuracy ($\tau^+ = 16, \tau^- = 1$):



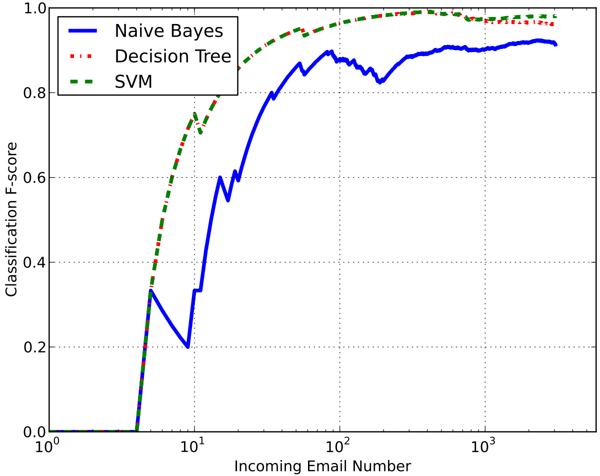
Auto-Learning Performance

Auto-Learning Accuracy ($\tau^+ = 30, \tau^- = 1$):



Auto-Learning Performance

Auto-Learning F-Score ($\tau^+ = 16, \tau^- = 1$):



Auto-Learning Performance

SpamFlow Weight in Composite Score

- Currently a (configurable) fixed weight vote by SpamFlow that contributes to final score
- We experimented with two weights
- Working on optimizing and providing continuous weight depending on SpamFlow confidence

Real-World Benefit

	tp	*fp*	*tn*	*fn*	F-Score
SpamAssassin	5288	3	137	87	0.991
SpamFlow	5224	65	75	151	0.980
SA+SpamFlow(1)	5299	3	137	76	0.992
SA+SpamFlow(2)	5335	19	121	40	0.995

Outline

1 Motivation

2 Detecting Bot-Generated Spam

3 SpamFlow Architecture

4 SpamFlow Results

5 Conclusions

Current Research

Application to Other Domains:

- Attacks (automated) against web servers
- Can't rely on reputation/ports (as compared to SMTP)
- Scam-hosting infrastructure, Botnet CDNs (e.g. Canadian pharma, proxying, relaying, etc.)

Utilizing Transport Features:

- Adversarial TCP/IP stack to cause suspected bot to perform *more* work, contributing to the feedback loop such that transport features are exacerbated
- LISA 2011 poster with details, come see us!

SpamFlow Availability

SpamFlow Availability:

- Final testing phases
- Running in production at several installations
- autoconf’d, packaged, etc.
- January, 2012 release
- OpenSource license
- Tested with Postfix/Qmail and SpamAssassin
- Please contact us, or sign-up on mailing list for release updates

`http://www.cmand.org/spamflow/`

Summary

Thanks!

- Attacking spam at a **different layer**
- Created SpamFlow SpamAssassin plugin + architecture:
  - *On-line* and *real-time* transport-layer classification of live email messages on a production MTA.
  - Auto-learning of transport features to build model across different operating environments without human training.

Questions?

`http://www.cmand.org/spamflow/`