

Learning on User Behavior for Novel Worm Detection

Steve Martin, Anil Sewani, Blaine Nelson, and Karl Chen
{steve0, anil, nelsonb, quarl}@cs.berkeley.edu

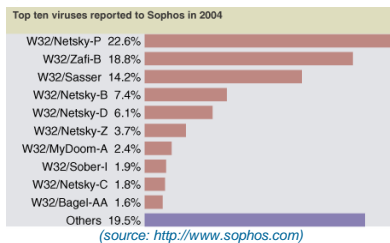
University of California at Berkeley
Spring 2005

Talk Outline

- Introduction
- Recent Research
- Our General Approach
- A System for Detecting Worm Infections
- Results
- Conclusions and Future Plans
- Questions/Feedback?

The Problem: Email Worms

- Email worms cause billions of dollars of damage yearly.
 - Nearly all of the most virulent worms of 2004 spread by email:



Current Solutions

- Signature-based methods are effective against known worms only.
 - **25 new Windows viruses a day** released during 2004!
- Human element slows reaction times.
 - Signature generation can take hours to days.
 - Signature acquisition and application can take hours to never.
- Signature methods are mired in an arms race.
 - MyDoom.m and Netsky.b got through EECS mail scanners

Recent Research Directions

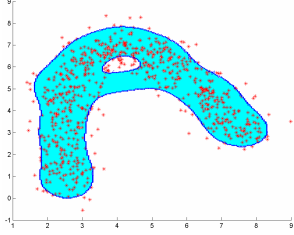
- **Objective:** lower/eliminate rate of novel worm propagation to contain epidemics.
- Idea 1: Throttle new connection rate or rate of emails to different recipients allowed.
 - *Connection Throttling* (HP), implemented in WinXP SP2
- Idea 2: Adaptive rule-based traffic filtering.
 - *Bro*, *Automated Worm Fingerprinting*, etc.
- Idea 3: Unsupervised statistical learning.
 - *Email Mining Toolkit* (Columbia), etc.

Using Statistical Learning

- Unsupervised learning on network behavior.
 - Leverage behavioral invariant among infected machines.
 - By definition, a worm seeks to propagate itself over a network.
- Abnormal behavior could indicate infection.
 - Combine with signature methods.
- Quarantine suspect machines for supervisor clearance.
 - Prevents propagation of infection.

Novelty Detection Overview

- Main idea: define what is 'normal' and classify data based on this definition:



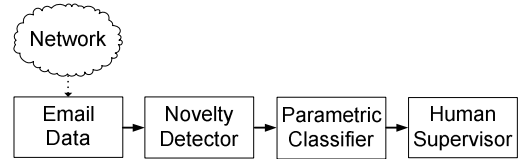
Statistical Learning Caveats

- Previous work: novelty detection by itself is not enough.
 - Many false negatives = worm attack will succeed.
 - Many false positives = irritated network administrators.
- Common solution: make the novelty detector model very sensitive.
 - Tradeoff: Introduces additional false positives.
 - Can render a detection system useless.

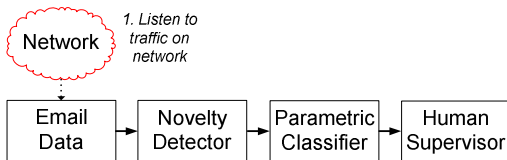
Our Approach

- Use two-layer approach to filter novelty detector results.
 - Novelty detector minimizes false negatives.
 - Secondary classifier filters out false positives.
- Leverage human reactions and existing methods to improve secondary classifier.
 - Use supervisor feedback to partially label data corpus
 - Correct and retrain as signatures become available
- Filter novelty detection results with *per-user* classifier trained on *semi-supervised* data.

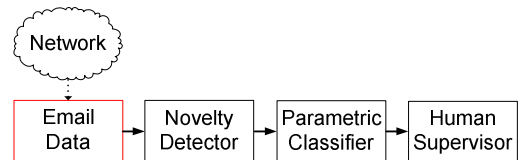
Per-User Detection Pipeline

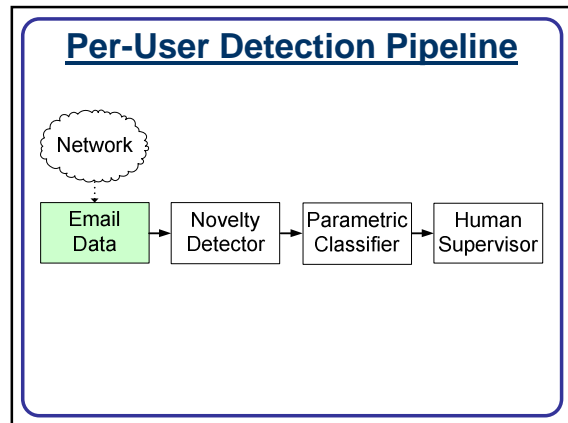
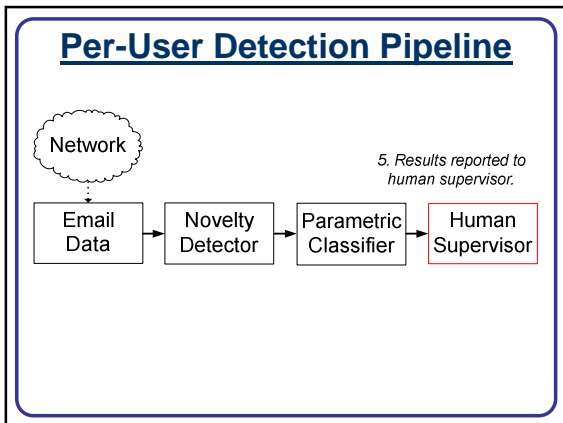
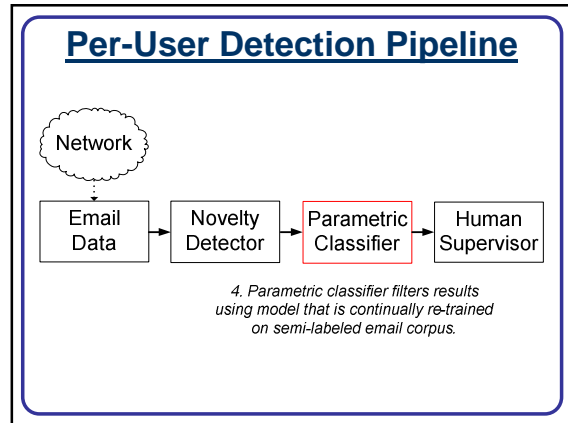
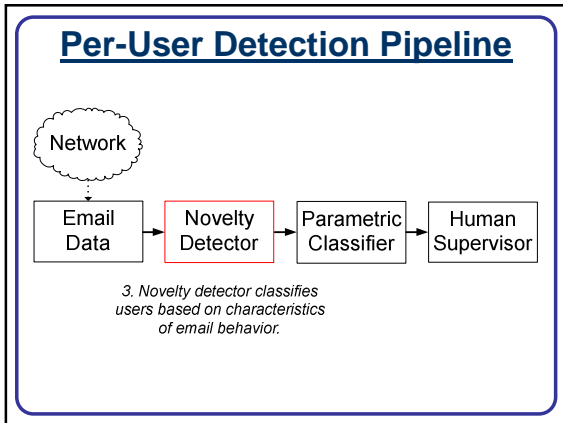


Per-User Detection Pipeline

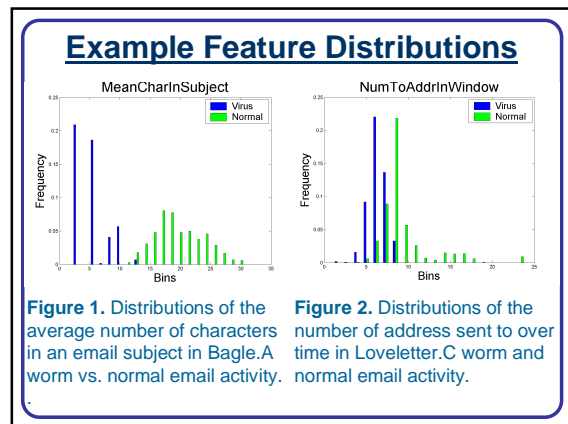


Per-User Detection Pipeline

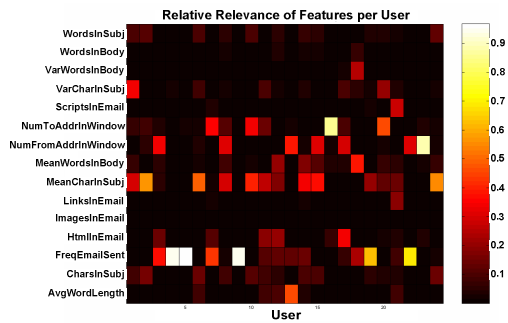




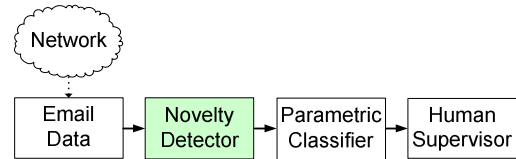
- ### Features
- Distinguish between *per-email* and *per-user* features calculated on outbound email.
 - User features capture elements of behavior over a window of time.
 - Examples: mean number of words in emails, ratio of emails sent with/without attachments, etc.
 - Email features examine individual snapshots of behavior.
 - Examples: number of characters in the email subject, presence of html in the email, etc.



Feature Relevance



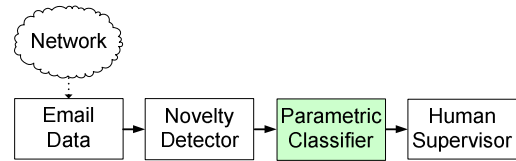
Per-User Detection Pipeline



Novelty Detector

- Any Novelty Detector can be employed.
 - SVMs, GMMs, etc
- The results presented in this talk were generated using a Support Vector Machine.
 - One SVM is trained on all normal email in the network.
- SVM performs classification by forming a tight boundary around normal email.
 - Anything outside this boundary is considered anomalous, and passed on as a potential virus.

Per-User Detection Pipeline



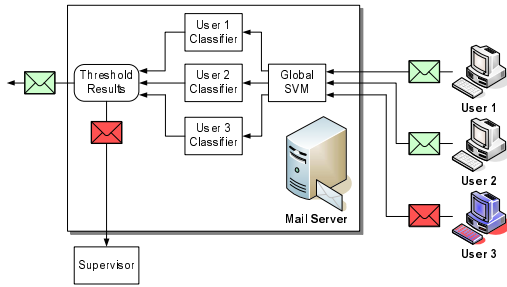
Parametric Classifier

- Exploit distinct feature distributions using a *generative graphical model*.
 - Fit a specific model for each user.
- Current implementation uses a Naive Bayes approach.
 - Assumes all features independent.
 - Fits specific distributions to infected and non-infected feature data.
- Classifier continually retrains over *semi-supervised* data.

Talk Outline

- Introduction
- Recent Research
- Our General Approach
- ***A System for Detecting Worm Infections***
- Results
- Conclusions and Future Plans
- Questions/Feedback?

System Deployment



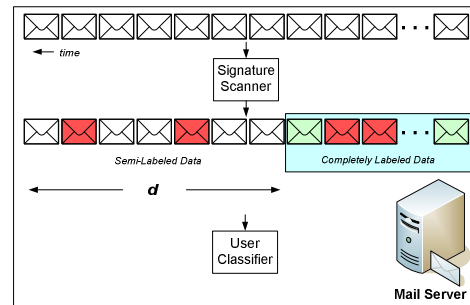
Initial System Training

- SVM trained initially on good email from all users.
- Classifier trained initially on set of completely labeled email.
- Every h hours or n emails parametric classifier retrained
 - For the results in this talk $h = 12$.
- The classifier is retrained on both the labeled and unlabeled data.

Using Scanner Feedback

- Before retraining, corpus labels updated via virus scanner.
 - Corpus is a large sliding window of emails.
- For each email within last d days:
 - If the scanner returns virus, we label virus
 - If the scanner returns clean, we leave the current label (typically nothing).
- Outside the last d days of emails, apply scanner label directly.
 - In this talk, $d = 7$.

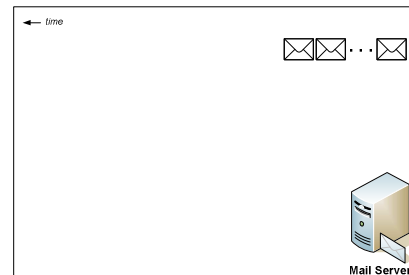
Scanner Labeling Process



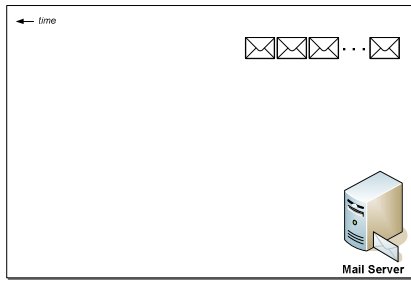
Using Human Feedback

- Threshold number of emails classified as virus to detect user infection.
 - Machine is quarantined
 - Infected emails placed in a queue
- Supervisor accepts or denies machine is infected.
 - If no infection, queue of emails released.
- If infection confirmed, i random messages from queue are labeled by the supervisor.
 - Model is retrained
 - Labels retained until virus scanner corrects them.

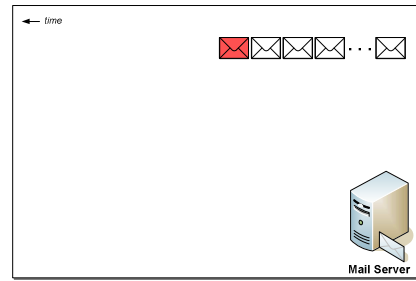
Infection Labeling Process



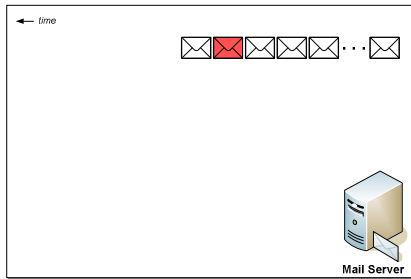
Infection Labeling Process



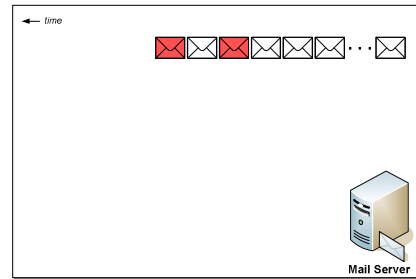
Infection Labeling Process



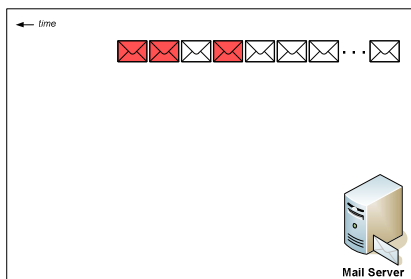
Infection Labeling Process



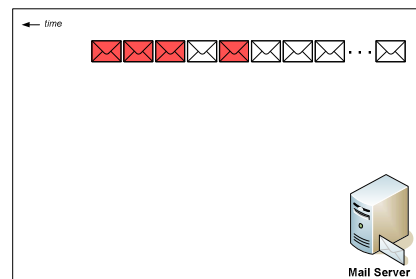
Infection Labeling Process



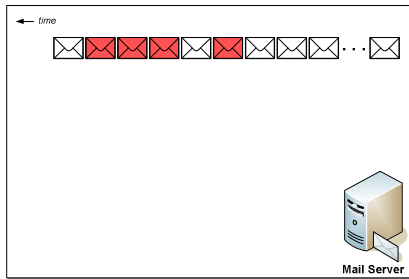
Infection Labeling Process



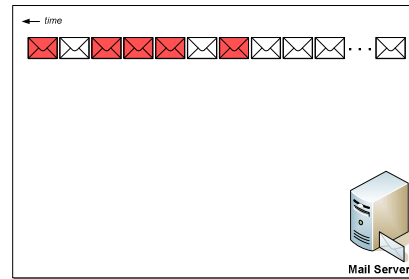
Infection Labeling Process



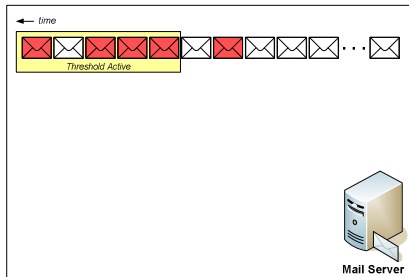
Infection Labeling Process



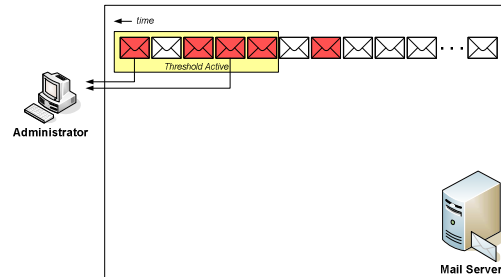
Infection Labeling Process



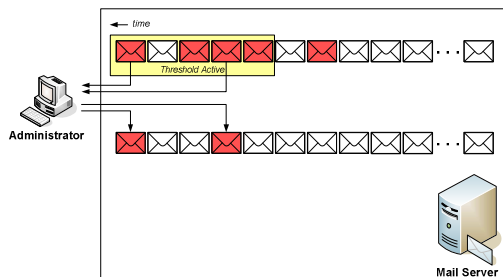
Infection Labeling Process



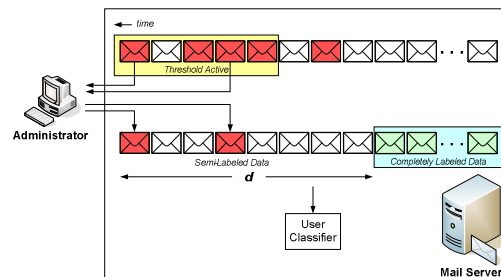
Infection Labeling Process



Infection Labeling Process



Infection Labeling Process



Why Semi-Supervised?

- Use all available data to improve results.
- During update lag period, may not be enough supervised data to accurately capture model parameters.
 - Need updated model to detect novel worms.
 - Use EM on unlabeled data to improve estimates.
- Caveat: only helps if distributions chosen to model data are a good fit.
 - Otherwise, can decrease accuracy!

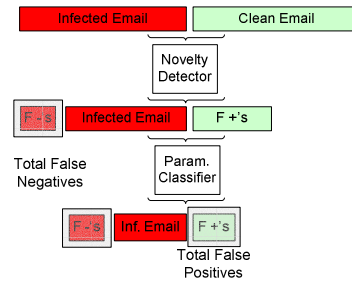
Preliminary Evaluation

- Instrumented a mail server to calculate feature data on live email.
 - Conducted a live study with 20 users to collect data and learn distributions of our features.
 - Used the Enron data set for further evaluation.
- Collected virus data from several email worms using virtual machines.
- Constructed training set of real email traffic artificially 'infected' with one or more viruses.

Preliminary Evaluation

- Used Bagle.f, MyDoom.u, MyDoom.m, Netsky.d and Sobig.f
- Infections interleaved while preserving intervals between worm emails.
- Worm email traces captured using same address book as real email used.
- Trained on 2 worms, tested on one.

Explanation of Results



Preliminary Results

Table 1. Results using SVM

Train/Test	Test Emails	False Positives	False Negatives	Accuracy
Bf, Mu / Nd	1400	40.96%	0.50%	70.57%
Bf, Nd / Mu	1400	41.06%	0.50%	70.50%
Bf, Sf / Mu	1400	41.96%	0.50%	69.86%
Bf, Mm / My	1400	41.76%	0.50%	70.00%
Mu, Nd / Bf	1400	41.06%	0.50%	70.50%
Mu, Sf / Mm	1400	41.49%	0.51%	70.07%
Mu, Mm / Bf	1400	41.46%	0.50%	70.21%
Nd, Sf / Mm	1400	42.49%	0.51%	69.36%
Nd, Mm / Bf	1400	41.06%	0.50%	70.50%
Sf, Mm / Bf	1400	42.06%	0.50%	69.79%

- Average Accuracy: 70.14%

Preliminary Results

Table 2. Results using SVM and Sup. Classifier.

Train/Test	Test Emails	False Positives	False Negatives	Accuracy
Bf, Mu / Nd	1400	3.20%	0.50%	97.71%
Bf, Nd / Mu	1400	5.80%	0.50%	95.86%
Bf, Sf / Mu	1400	2.90%	48.25%	84.29%
Bf, Mm / My	1400	6.10%	0.50%	95.64%
Mu, Nd / Bf	1400	6.10%	0.50%	95.64%
Mu, Sf / Mm	1400	3.40%	0.51%	97.57%
Mu, Mm / Bf	1400	6.20%	0.50%	95.57%
Nd, Sf / Mm	1400	3.70%	0.51%	97.36%
Nd, Mm / Bf	1400	6.30%	0.50%	95.50%
Sf, Mm / Bf	1400	3.40%	0.50%	97.57%

- Average Accuracy: 95.27%

Preliminary Results

Table 3. Results using SVM and Semi-Sup. Classifier.

Train/Test	Test Emails	False Positives	False Negatives	Accuracy
Bf, Mu / Nd	1400	6.00%	0.50%	95.71%
Bf, Nd / Mu	1400	6.20%	0.50%	95.57%
Bf, Sf / Mu	1400	3.10%	0.50%	97.79%
Bf, Mm / My	1400	6.40%	0.50%	95.43%
Mu, Nd / Bf	1400	6.50%	0.50%	95.36%
Mu, Sf / Mm	1400	3.40%	0.51%	97.57%
Mu, Mm / Bf	1400	6.60%	0.50%	95.29%
Nd, Sf / Mm	1400	3.50%	0.51%	97.50%
Nd, Mm / Bf	1400	5.00%	0.50%	96.43%
Sf, Mm / Bf	1400	3.40%	0.50%	97.57%

- Average Accuracy: 96.42%

Conclusions

- Bottom line: our two-layer approach decrease false positives by an order of magnitude.
 - Supervisor overhead minimal
 - Time to model convergence is short
- Improves on well-known strengths of novelty detection with minimal cost.
- This technique could be applied to other problems in security.
 - Masquerade detection
 - Spam-bot detection

Some Future Work

- Further analyze individual contributions by features.
 - Perform sensitivity analysis, leverage Enron data set.
- Examine additional features and refine model
 - Incorporate time dependencies between features
- Measure detection lag before machine classified as infected.
- Consider other methods of filtering.
 - Our model is parametric; non-parametric methods may work better (decision trees, etc).

Feedback?

Comments and questions welcome!