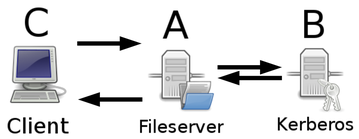
NC STATE UNIVERSITY

Identification of Network Service Relationships in Distributed Systems: Improvements and Additions to NSDMiner

Barry Peddycord III and Dr. Peng Ning
Cyber Defense Lab, Department of Computer Science, North Carolina State University

Motivation

- Determining the relationships of machines in a production network is a non-trivial task.
- One such relationship is the **dependency relationship**.



- If *A* depends on *B, A* is unable to continue its operation if it fails to contact *B*. This is denoted $(ip_A, port_A, proto_A) \rightarrow (ip_B, port_B, proto_B)$.
- When a client *C* accesses an application *A*, it is expected that traffic will be observed to the service *B* that *A* depends on.
- NSDMiner *[Natarajan, et al., INFOCOM 2012]* is a tool that accurately identifies network service dependencies by observing this relationship in network traffic.
- This research extends NSDMiner by improving its accuracy and recognizing other network relationships.

Future Works

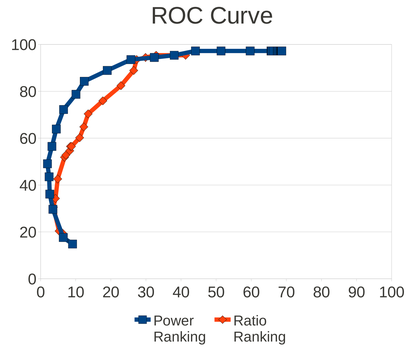
- Enable NSDMiner to respond to dynamic changes in the network configuration.
- Discovery and identification of hidden or rogue services that administrators can't identify.

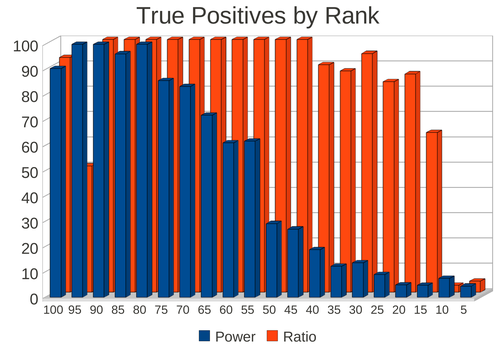
Acknowledgements

- Thanks to Arun Natarajan for his continued help providing support for the NSDMiner code.
- Thanks to the CSC IT Staff for their assistance with our data collection infrastructure and ground truth.
- This work is supported by the U.S. Army Research Office (ARO) under MURI grant W911NF-09-1-0525.

Candidate Ranking

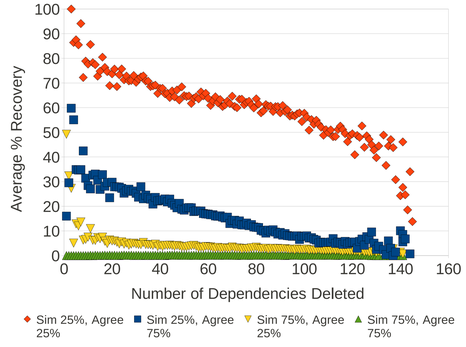
- If an application accesses a service every other time it runs, does that suggest a 50% chance of being a dependency?
- The number of times that an application contacts its dependencies scales **logarithmically**, not linearly.
- We count the number of times that the service *A* is used (*calls*), and the number of times that *A* contacts *B* (*dweight*).
- The confidence ranking of $A \rightarrow B$ is equal to $log_{calls}(dweight)$, and all candidates that fall below α confidence are dropped.
- To validate our approach, we run NSDMiner on traffic obtained from the CSC department's network in EB II and compare it to a ground truth developed by our IT staff.

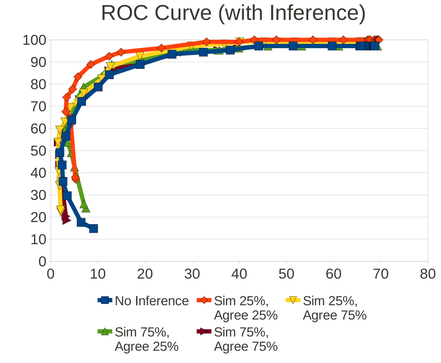




Service Inference

- In networks, there are many instances of particular applications (i.e., multiple mail servers, web servers). If two applications are similar, their dependencies should be similar.
- If an application is seldom used, its dependencies can be **inferred** from the dependencies of similar applications.
- We find *similar* services, and then identify the dependency candidates they *agree* on. Services that are missing any of agreed candidates have them inferred as new candidates.
- Effectiveness of this approach was evaluated by removing traffic from known true positives and attempting to re-infer the dependencies. It's possible to get high recovery without incurring false positives.





Cluster Detection

- Some services are part of redundant clusters for backup or load-balancing purposes.
- If B_1 and B_2 are part of a cluster, then when B_1 appears as a dependency candidate, it's probable that B_2 will also appear.
- The number of times that they appear together in a service's dependency candidates is the **support** for the cluster.
- The cluster contains all of the services with high pairwise support, creating a connected graph with $B_1 \ldots B_n$.
- Individual references to $B_1 \ldots B_n$ in the candidate list are replaced with a reference to the *B* cluster, reducing the total number of candidates reported by NSDMiner.