ARAKIS – project goals

- To create a distributed Internet threat detection system
- Identify and describe novel threats
- Tool to aid the security decision making process in an organization
- Tool to aid CERT/CSIRT type teams in their work
- To achieve a unique view of the threats that appear on the “Polish Internet”
- Analysis of network security trends
- Develop an early warning system
ARAKIS – current main features

- Data sources: honeypots, firewalls, antivirus systems
- Detection of anomalies in network traffic
  - Detection of new worms, exploits in the wild (including 0-day)
  - Extraction of signatures of the detected worms, exploits
- Signature management
- Correlation of data from different sensors
  - Monitoring of the efficiency of the latest snort ruleset
- Statistics
- Alerting system
Characteristics of a good signature (1/2)

- Detects the attack
- Low false alarm rate
- Can be generated quickly
- Independent of application level protocols
- Can be used in existing IDS/IPS systems
Characteristics of a good signature (2/2)

- Exploit vs. vulnerability
- Usage of the "de facto" standard: signatures representing a sequence of bytes that characterize a threat
- Operating at a network level allows for the quick deployment of the signature until hosts patched (important from an early warning point of view)
Comparing by hashing – sliding window across a packet
Generating signatures in ARAKIS (1/2)

- Our proposal: use Rabin windows to initially classify flows (detected anomalies), the generation of signatures transferred to other algorithms that compute signatures “off-line” (like LCS)
Generating signatures in ARAKIS (2/2)

- Define grouping rules:
  - Completed flows are periodically grouped based on their Rabin similarity (for example, group all expired flows to the same destination port that contain 30% of the same fingerprints)
  - Heuristics: for every group, check the amount of unique sources in a given period. If a threshold is reached, the group is sent for further analysis "off-line"
  - An external process computes LCS on every submitted group
Implementation (1/2)

- Base software: *snort* and *Apache*
- Rabin fingerprints implemented as *snort* plugin called *flow-rabin* on top of the standard *flow* and *stream4* plugins
- The *flow-rabin* plugin is the basis for the *flow-classifier* plugin, which implements various preliminary grouping rules
- When a threat cluster is detected, the cluster is transferred to the *mod_lcs* *Apache* module for LCS signature extraction
- Communication between *snort* and *mod_lcs* TCP based
Implementation (2/2)

SNORT
flow-classifier

flows grouped in boxes based on their Rabin similarity

APACHE
mod_lcs

LCS 1 LCS 2 LCS 3 LCS 4 LCS 5
Signature management

- Levenshtein distance between strings as a distance metric
- Use clustering algorithms (simplified *dbscan*)
- LCS signatures are periodically clustered and manually classified (with support from Bleeding Snort rules)
- For efficiency reasons, long repetitions of characters (such as NOOPs) are packed to a certain maximum length
- Dynamic radius of a cluster based on the length of the core member in order to allow for better clustering of both short and long signatures
- Implemented as a PHP5 process run periodically
Correlation of data from sensors

- Look for signature sequences, reported by different sensors, that have not been seen before and that do not trigger standard snort alarms
- Look for groups of signatures that may form a new cluster
- Look for new (not seen in a while) destination ports with payload
- Look for new (not seen in a while) destination ports that are being scanned
- All the above implemented via SEC (Simple Event Correlator)
Test results (1/2)

- 24 hours monitoring of 5 /26 subnets (honeyd/nepenthes)
- Total 775 716 packets collected
- Grouping rules: 3 distinct sources with flows that are 30% similar in a space of 5 minutes
- 408 LCS signatures generated (LCS generated per packet)
- 63 clusters formed
- 63 signatures computed (one per cluster)
- 7 signatures found to generate false positives (based on a trace of “normal” traffic)
- 21 further signatures dropped (vetting process)
Test results (2/2)

The 35 remaining clusters:
- LSA exploit (port 445/TCP) – 10 clusters
- ASN1 exploit (port 445/TCP, port 139/TCP) – 8 clusters
- Winpopup spam (ports 1026-1029 UDP) – 5 clusters
- RPC DCOM (port 135/TCP, 1025/TCP) – 4 clusters
- Shellcode x86 NOOP (port 445/TCP) – 2 clusters
- Port 1026/UDP unknown [1] – 2 clusters
- SQL Slammer (port 1434/UDP) – 1 cluster
- Port 1433/TCP unknown [2] – 1 cluster
- NetBIOS query (port 139/TCP) – 1 cluster
- HTTP OPTIONS query (port 80/TCP) – 1 cluster

[1] Probably related to Winpopup spam
[2] A large amount of short packets to the standard MS SQL Server port - possibly a brute force attempt. It was not identified by any Snort rules.
Future

- Application of new algorithms for detection of anomalies and classification of flows
- Polymorphic attacks - implementation of “off-line” algorithms other than LCS
- Correlation with external sources of information
- Early warning on threats that utilize passive methods of propagation