AFRL Research Collaboration Program

Contract: FA8650-13-C-5800

Project Title: Extraction of Social Context via Synthetic Pollination for Information Tracking and Control

University: Louisiana Tech University

REPORT COVERS PERIOD 01-JAN-14 THROUGH 01-APR-14

PROJECT TEAM MEMBERS

Lead University POC
Dr. Jean Gourd
318-257-4301
jgourd@latech.edu

Students
Thomas Bozeman
Computer Science undergraduate student, senior

Shawn Killeen
Computer Science undergraduate student, sophomore

AFRL Technical POC
Kenneth Littlejohn

TECHNICAL DISCUSSION

Background

This work centers around a synthetic biological mechanism that is based on the collection and analysis of network data designed to assist in the tracking and control of information flow in a LAN. This mechanism can be used to aid in the detection and mitigation of insider threats, for example, through the early prediction of adversarial behavior. Behaviorally, it generates additional context from base information in the form of social meta-data mined from interactions between users and nodes in the network. There is an ancillary benefit in that the additional context is expected to reduce the effective capabilities of the insider threat due to the increased risk of detection.

At the core of this mechanism is a concept called "pollination" that is modeled biologically after the activities of bees. In the course of utilizing a node, users will leave "pollen" indicating their interests in terms of other nodes, users, and external entities. This meta-data is distinct in that it is contextual in nature and ultimately reduces the amount of information an analyst must process while providing contextual connections that are either missing from data or difficult and expensive to correlate. Once analyzed, it can be used for early prediction of malicious behavior for the purpose of detecting insider threats early enough to assist in preventing attacks.

The proposed work centers on a mechanism called "pollination," a distributed host-based sensor network that utilizes intelligent agents to dynamically process raw network data into meta-data. This meta-data is not designed to replace existing data, but is rather intended to provide additional context via distributed preprocessing. The goal is to measure specific social behaviors as they happen, as opposed to measuring everything and then proceeding to work backwards as is typically done in traditional forensics. This data takes the form of social context and provides information about the relationships formed between nodes and users, and allows the tracking of information through the network.

Pollination is a scheme that relies upon the biological metaphor of bees. For the sake of the metaphor, messages within the network implementing pollination are bees while nodes are flowers. This allows for the tracking of communicating peers at the end points. This concept can be extended to infer the entire social network of a machine; however, the purpose of this work is to track and provide some measure of control of the flow of information in the network. To that end, the linking of pollen to a specific user identity within the local network is conceived. This is an additional tier of information that lies beyond the host. In essence, both the operator and the machine used are recorded. Thus, for every machine and user a set of pollen will exist at the network barrier indicating the other users and machines that form its peer group. A more thorough introduction of the pollination scheme and its potential applications can be found in our introductory paper on the topic [3].

The inclusion of host classification can allow for a less social- and more activity-oriented view of a user's communications. Additionally, pollen does not present a binary record of social interactions, and the quantity of pollen present would be indicative of the relative strength of a social connection. The pollen collected on individual messages provides for additional forensics capabilities in the form of a visit history. This visit history of individual packets could potentially be used to detect attackers or insiders trying to stealthily sniff packets even if they were not necessarily changing them. In addition, the need to pollinate incoming information at the network edge leads to the opportunity to automatically classify WAN (wide area network) traffic and thus generate a social selection of users that are interested in that traffic. This idea can also be applied to LAN traffic, with its simplest implementation being functional in that these groups of users have functional reasons to associate.

A major component of the pollination mechanism revolves around an intelligent agent-based command and control (C2) system. This component is necessary to house and display the aggregated information from the pollination network. Through the mechanism of collecting the information, data can be correlated providing for additional context and some data integrity checking. There is an inherent anomaly detection included in this correlation. Because of the double-ended nature of pollination, the reports from sender and receiver should more or less agree. Furthermore, this data can be used as a behavioral use pattern although it does not give insight into whether these patterns are normal or abnormal. However, it is likely that classification mechanisms will assist with this. The meta-data provides additional information that would normally be time-consuming to mine out of lower level data. This leads to a net reduction in the amount of data a higher level algorithm would have to consider in a first pass situation particularly with the inclusion of more sophisticated classification algorithms. Via user input and interaction, the classification mechanisms can be further modified allowing for the dynamic tweaking of the distributed sensor network.

Once a comprehensive view of the system is aggregated, other techniques can be applied to the meta-data. One application is the creation of valid pollen patterns. This can be accomplished by simply keeping track of all valid types of pollen. If a packet is encountered with invalid pollen, it would be a very strong indicator of malicious behavior. In fact, it would most likely be an indicator of either an intruder who did not understand pollen or an insider attempting to cover his tracks.

Although there are clear social context aspects of pollination, at its technical core it is a mechanism for tracking packets through the network. Thus it is closely linked to digital forensics, and one of the aspects of interest is the ability to backtrack the path of information in the network. That is, it is conceivable that, should information be leaked from the network (or even simply reach the network edge), its exact path back to the point of origin could be easily mapped via pollination (see Figure 1). This is because pollen is bidirectional: in addition to dropping off pollen, it can be picked up by a packet. Thus, for every packet we have a trace of its travels through the network. This trace would, at a minimum, provide the order of nodes visited. By combining a higher level algorithm, it would be possible to observe irregularities in the pollen pattern. There are some technical questions of the possible granularity this trace can take while not affecting performance; however, should the performance costs be significant, granularity can be varied depending upon the origin of the message. That is, packets originating from machines with more sensitive data or from machines exhibiting suspicious information may have greater granularity.

The uniqueness in our approach centers on the pollination mechanism. It provides socially-oriented data as opposed to traditional approaches to data organization (e.g., topographic or

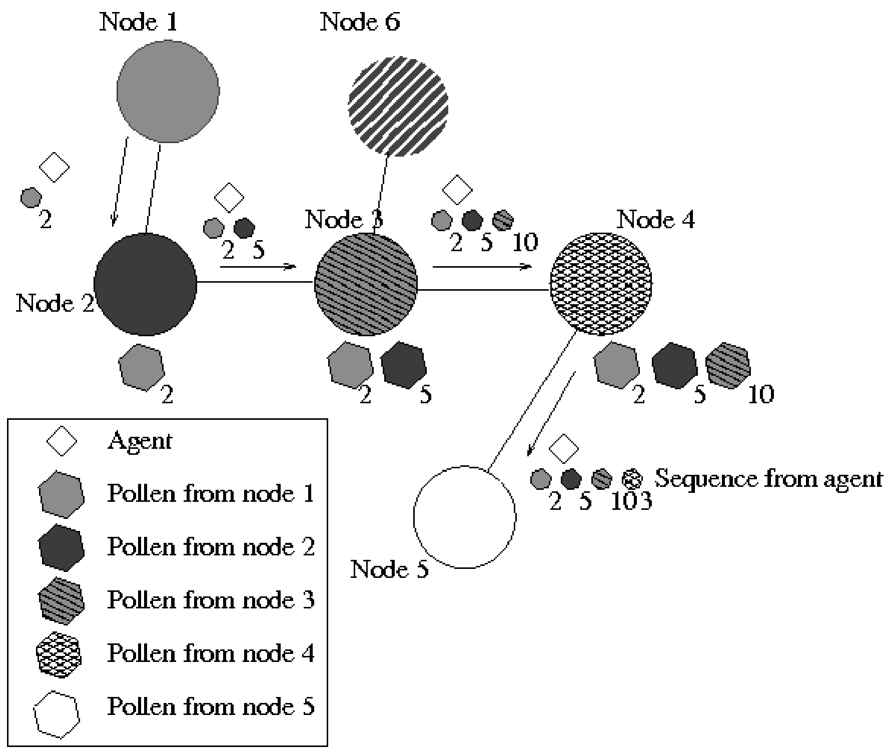

Figure 1: Path of information flow via pollination

temporal). The key advantage of this is that it allows for the study and observation of collaborative efforts. This can directly be used to detect and mitigate insider threats in addition to controlling the flow of information. The pollination mechanism also provides for bidirectional interest tracking from both the source and destination without significant additional computational overhead or data processing. It is extremely scalable both with respect to the size of both the computing infrastructure and potential user base while also being widely deployable on a diverse variety of complex computer environments. This is largely due to the system-neutral nature of the mobile agent infrastructure. This infrastructure provides for its own security as well as a great deal of potential integration with existing and future algorithms.

Current Work

There are three main goals of the project this year (01-SEP-2013 through 31-DEC-2014):

(1) Develop and analyze the optimal technical implementation for pollination.
(2) Design the multi-agent system (MAS) that will ultimately form the C2 system.
(3) Generate appropriate network traffic data for testing the pollination scheme on a testbed.

Specifically during the current period 01-JAN-2014 through 01-APR-2014, tasks have primarily covered item (1) above. Note that there has been concurrent work, primarily related to item (2) above; that is, we are simultaneously designing the MAS that will implement pollination. In many respects, both must be designed together.

We have identified several technical implementations for pollination. At this time, both seem to possess the ability to provide the desired functionality.

Method #1: Payload Hash with Agent Follower

In this method, each node in the LAN is required to have a unique identifier (ID), a symmetric encryption key (K), the identifier(s) of the neighbor node(s) (i.e., those directly connected), and a database for the IDs of nodes that particular messages have been forwarded to (i.e., outbound messages). For any message created within the LAN (or any message entering it), a wrapper message is created that allows the system keep track of where it has been. After wrapping the original content and forwarding it, an agent is created to follow behind it, track what nodes have been visited, and let subsequent nodes know where it is coming from.

Figure 2 shows the types of packets that are in the system at any given point:

(1) Original packet: this will only be forwarded if the message is leaving the LAN.
(2) Tracer packet: this is a packet created once a packet enters the LAN, but only if it will continue through the LAN. It contains the MSG tag, the original packet, and a hash of the original packet. It uses the hash to mark the path that it takes.
(3) Agent packet: one of these is created for each packet that is wrapped in a message packet. It contains the AGENT tag, the same hash as the packet it was created for, and the IDs of the nodes it visits along the way. It uses the hash to follow behind the message packet.

Figure 3 shows what pollination might look like in a simple system, passing a packet from node A to node C:

(1) The original packet has just been created at node A.
(2) The hash of the original packet and the ID of the node that it will be forwarded to are stored at node A. A message packet, containing the hash and the original packet, is created.
(3) The message packet is forwarded to node B; the hash and the next forwarding location are stored at node B. An agent packet, containing the hash and the ID of the current node, is created at node A.
(4) The message packet is forwarded to node C. The agent packet is forwarded to node B; node B saves the visited nodes from the agent packet. The hash and forwarding address at node A are deleted.
(5) The message packet is stripped, leaving the original packet. The agent packet saves that it has been to node B.
(6) Node C uses the original packet. The agent packet is forwarded to node C; node C saves the visited nodes from the agent packet. The forwarding address at node B is deleted. The agent packet is discarded.

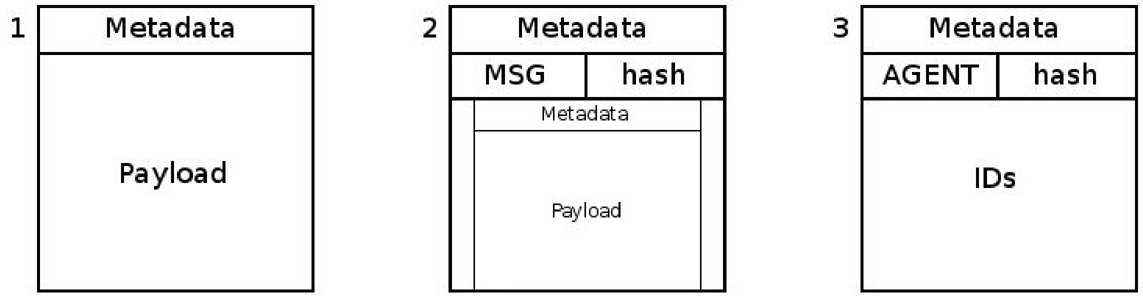

Figure 2: Packet types

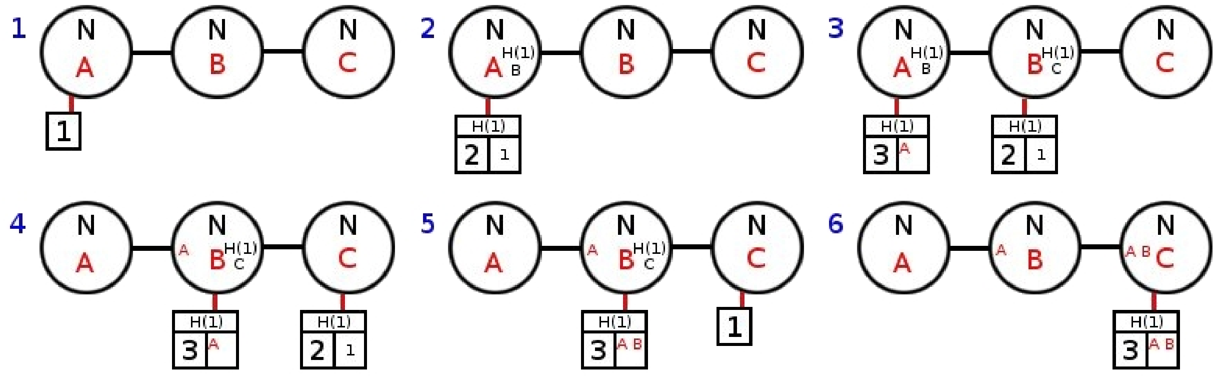

Figure 3: Pollination with payload hashing and an agent follower

The following algorithm is implemented in the send portion (i.e., when a message is sent to a node):

```
Send(msg)
  if msg is not about to be forwarded outside of the LAN
  then tag ← MSG
       hash ← H(msg)
       out ← encrypt {tag, hash, msg} with K
       outbound[hash] ← ID of node msg is to be forwarded to
       forward(out)
       tag ← AGENT
       payload ← ID
       agent ← encrypt {tag, hash, payload} with K
       forward(agent, outbound[hash])
       outbound[hash] ← NULL
  else if msg is at its destination
       then record that a message came from outside the network
                                                      to this node
            deliver msg
       end
  else
       forward(msg)
  end
```

The following algorithm is implemented in the receive portion (i.e., when a message is received from a node):

```
Receive(msg)
  if msg source inside of LAN
  then {tag, hash, payload} ← decrypt msg with K
       if tag = AGENT
       then store payload in pollen database
            if outbound[hash] exists
            then append ID to payload
```

```
                    out ← encrypt{tag, hash, payload} with K
                    forward(out, hash)
                    outbound[hash] ← NULL
                else
                    discard(msg)
                end
            else
                if tag = MSG
                then if msg to be forwarded outside of LAN
                    then forward(payload)
                    else if msg is at its destination
                        then deliver payload
                        end
                    else
                        outbound[hash] ← forwarding address
                        forward(msg)
                    end
                end
            end
        else
            Send(msg)
        end
```

Method #2: Cascading Signatures
This method can be summarized as appending a node identifier to packets and cryptographically signing the new payload at each host in the transmission. Additionally each node stores the node the packet was received from, the node it will be sent to, and optionally the current signature of the packet. The choice of hashing algorithm is left undefined but recommended to be a member of the SHA2 family. Likewise, the choice of encryption algorithm is left undefined: public key encryption would be valuable from a security standpoint, but would be orders of magnitude slower than symmetric key encryption.

When a packet is generated by a node to be sent to a remote host, it is intercepted before being sent. A bit string of 0s equal to the size of the chosen hash is appended to the payload of the packer. The intent is to later replace it with the cryptographic signature. Next a relative ID is appended to the bit string (again at the end of the payload) which represents the current host.

When the network is first formed, each host will communicate with its adjacent hosts to obtain short (7-bit) relative addresses for each. That is, if host B connects to a network and is adjacent to host A, then host B will choose an ID for A and communicate this ID to host A. Host A will then choose an ID to pair with the foreign ID it has just received, save this ID-pair, and then communicate this ID back to host B. Both hosts now have a local and a foreign relative host ID. If it should happen that a host has more than 127 neighbors (the maximum possible in 7 bits), then the 8th bit may be set to indicate that next byte to follow should be interpreted as part of the same relative address. This functionality may prove to either reduce pollination overhead or unnecessarily complicate matters.

Whether a relative addressing scheme is found to be appropriate or not, this pollination method dictates that an address corresponding to the current host followed by an address corresponding to the next host in the connection is to be appended to the packet. A hash of the packet is the calculated (including the reserved bit string of 0s and the appended addresses) which is then encrypted. This encrypted hash is then placed in the reserved (bit string of 0s) space, and the packet is forwarded to the next host. The current host then records for itself the address of the remote host and waits for a new packet to be sent.

When the packet is received at the next host, the signature is first checked to ensure the integrity of the packet. The host then appends the address of the next host, updates the signature, and routes the packet to the next host. The host finally logs the forward and backward foreign addresses. The procedure at the final host is much the same, minus the forward address.

We believe that the Cascading Signature method provides a way to uniquely identify a packet without having to completely duplicate it. In addition, it appears to provide an extra layer of security to prevent malicious interference.

Conclusions/Analysis to Date

The Payload Hash with Agent Follower method of pollination requires more overhead than some other methods and will increase the overall network traffic; however, it has the advantage of being able to quickly deliver the original message once it has left the originating node. This is because the pollen is handled by the agent which does not need to move through the system quickly, and can thus be put on hold whenever a new message is coming through. Due to the extra overhead, this method would likely be limited to application on low-to-mid-trafficked networks.

Thomas, provide a brief analysis of your method and its prospects for use as the final pollination method (we're looking for about 1 paragraph.

WORK FORECAST AND PLANS

Immediate work will focus on continuing to analyze the proposed technical implementations of pollination in order to better establish which works best. This will require the full implementation of a network testbed (an additional task over the next quarter).

In addition, we plan to investigate another unique method for implementing pollination, in addition to a variety of methods related to the two proposed above.

Figure 4 provides a planned schedule for the project throughout its duration this year (01-SEP-2013 through 31-DEC-2014).

Extraction of Social Context via Synthetic Pollination for Information Tracking and Control	Q1			Q2			Q3			Q4			Q5			
	Sep-13	Oct-13	Nov-13	Dec-13	Jan-14	Feb-14	Mar-14	Apr-14	May-14	Jun-14	Jul-14	Aug-14	Sep-14	Oct-14	Nov-14	Dec-14
OVERALL PROJECT	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Pollination Design				🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦
Pollination Technical Implementation										🟦	🟦	🟦	🟦	🟦	🟦	🟦
Testbed Design	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
Testbed Implementation								🟦	🟦	🟦	🟦	🟦	🟦	🟦		
Testing of Pollination within Testbed													🟦	🟦	🟦	🟦
C2 System Design													🟦	🟦	🟦	🟦
C2 System Implementation															🟦	🟦
Quarterly Reports					🟦			🟦			🟦			🟦		
Final Report															🟦	🟦
Published Paper (projected)															🟦	🟦
								Today								

Figure 4: Project schedule