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-APR-14 THROUGH 01-JUL-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.

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-APR-2014 through 01-JUL-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 continued to work on the two technical implementations for pollination identified in previous report(s).

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.

This method specified four unique packets that may be in the system at any given time:

(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 is tagged as a tracer with the origin address, and contains the original packet, the machine ID number, the user ID number, and the number of the message, relative to the machine and user (we will call these three the "hash" for clarity). It uses the hash to mark its path, and lets nodes know who used them and where they were using them from.

(3) Agent packet: one of these is created for each packet that is wrapped in a message packet. It is tagged as an agent with the destination address, and contains 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.

(4) Carrier packet (optional): this type of packet would act as a wrapper for packets which are not being tracked, in order to be able to detect instances of unauthorized machines being added to the network. It would be tagged as a carrier with the origin address.

The following algorithm is an updated version of the one previously defined. It 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 hash ← H(msg)
       if msg qualifies for tracking
       then out ← encrypt {hash, msg} with K
            src ← tracer tag
            outbound[hash] ← ID of node msg is to be forwarded to
            forward(out)
            payload ← ID
            agent ← encrypt {hash, payload} with K
            dst ← agent tag
            forward(agent, outbound[hash])
            outbound[hash] ← NULL
       else
            out ← encrypt {msg} with K
            src ← carrier tag
            forward(out)
       end
  else if msg is at its destination
       then record that a message came from outside the network
                                                    to this node
            deliver msg
```

```
        end
    end
```

The following algorithm is implemented in the receive portion (i.e., when a message is received from a node) and is an amended version of the one previously defined:

```
Receive(msg)
    if msg dst is the agent tag
    then
        {hash, payload} ← decrypt msg with K
        store payload in pollen database
        if outbound[hash] exists
        then append ID to payload
            out ← encrypt{hash, payload} with K
            forward(out, hash)
            outbound[hash] ← NULL
        else
            discard(msg)
        end
    else if msg src is the tracer tag
        then {hash, payload} ← decrypt msg with K
            if msg to be forwarded outside of LAN
            then forward(payload)
            else if msg is at its destination
                then deliver payload
                else
                    outbound[hash] ← forwarding address
                    forward(msg)
                end
            end
        else if msg src is the tracer tag
            then if msg is at its destination
                then payload ← decrypt msg with K
                    deliver payload
                else
                    forward(msg)
                end
            else
                flag communication attempt
            end
        end
    end
```

Method #2: Cascading Signatures

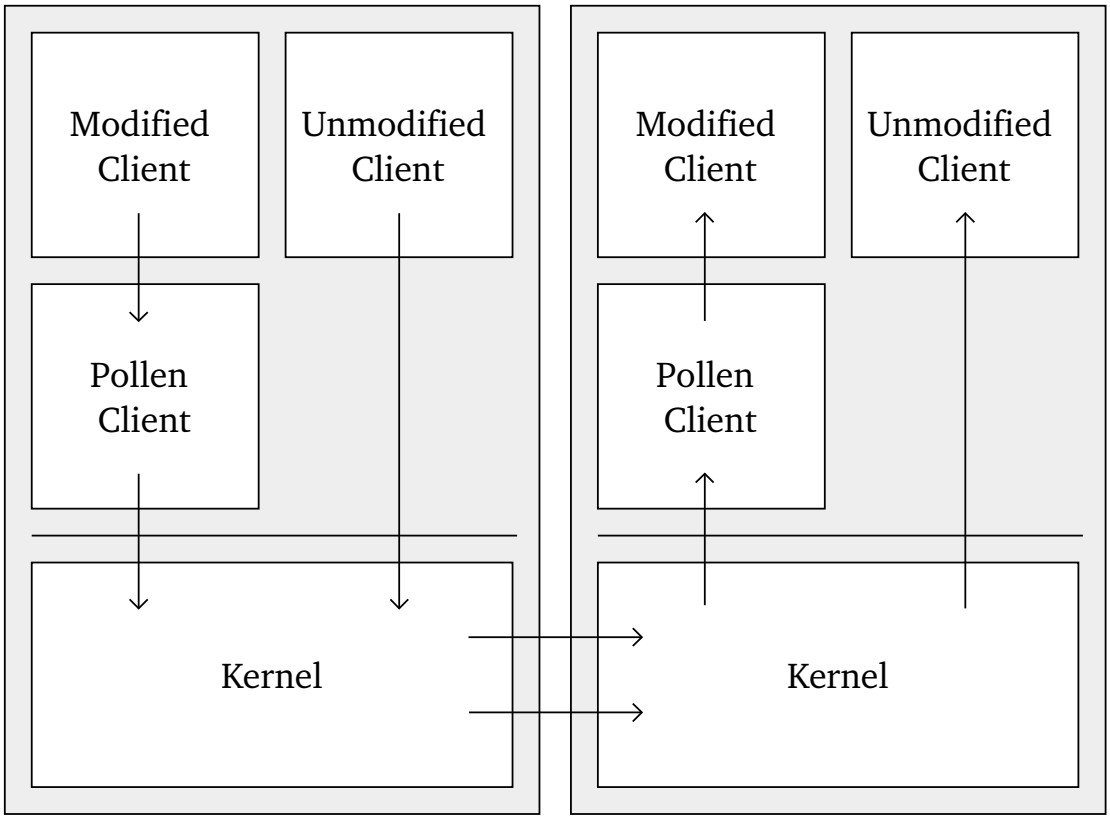
The cascading signatures method can be quickly summarized as a new network protocol which adds host identifiers to data transmissions, and cryptographically signs each transmission. The

protocol layer will consist of a 256-bit field to hold a cryptographic hash, a 16-bit field to hold a host-list-length field, a variable length list of 16-bit host-id fields, and finally the data payload.

The choice of hashing algorithm is left undefined, but recommended to be SHA256. Similarly, the choice of encryption algorithm is left undefined, but recommended to be 1024-bit RSA (or possibly a fast elliptic-curve cipher) for key negotiation during network creation, and 256-bit AES during data transfer. The possibility of algorithm negotiation during network creation has not been dismissed.

The initialization phase of the header generation begins by reserving 32 bytes of zeros, followed by populating the host ID sections, and appending our payload. Next we calculate the cryptographic signature of the entire packet (including reserved zeros and appended addresses), and encrypt this hash. This signature is then placed in the reserved space, and the packet is forwarded across the network 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.

At the next host, when the packet is received, the signature is first checked to assure the integrity of the packet. The host then appends the address of the next host, updates the signature, and sends the packet on its way. Finally, the host logs the forward and backward foreign addresses. The procedure at the final host is much the same, simply without having a forward address.



The current phase of the project is focused on hammering out details related to protocol interactions (IPv4/IPv6, TCP/UDP/ICMP/etc, and pollen), as well as the creation of testbed implementation for both testing and resource usage profiling. Due to the complexity of implementation, the testbed will utilize a user-land tool reminiscent of "netcat" with which to send and receive packets. The tool is currently being implemented in python in order to leverage previous work on the packet header definitions.

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. The decryption of the tracer packets at each node will likely cause a fairly substantial slowdown of network traffic, even without the agent packets. To remedy this, we may consider only applying this method to a certain percentage of the packets passed through the network, the exact number of which will be determined by trial. In order to maintain security, we may use carrier packets for the packets that are not being tracked, so that all untagged messages are immediately flagged and denied admittance to the network from anywhere that is not a network boundary.

We continue to believe that the Cascading Signatures method, on the other hand, would reduce the number of packets on the network but increase the amount of processing required per packet including the packet's size. Again, some benchmarking would be required in a number of network conditions and topographies in order to determine which method provides a better performance, both in terms of native traffic and pollination. This method does however provide an advantage in that traffic captured live on the network would contain a complete history of the hosts the packets representing the traffic had visited without need to search for an accompanying agent.

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).

With respect to the Cascading Signatures method, we will begin to write a Linux kernel module to intercept incoming and outgoing TCP connections and to wrap them in our protocol, and to begin a proof-of-concept implementation of a C2 system.

Figure 1 provides a planned schedule for the project throughout its duration this year (01-SEP-2013 through 31-DEC-2014).

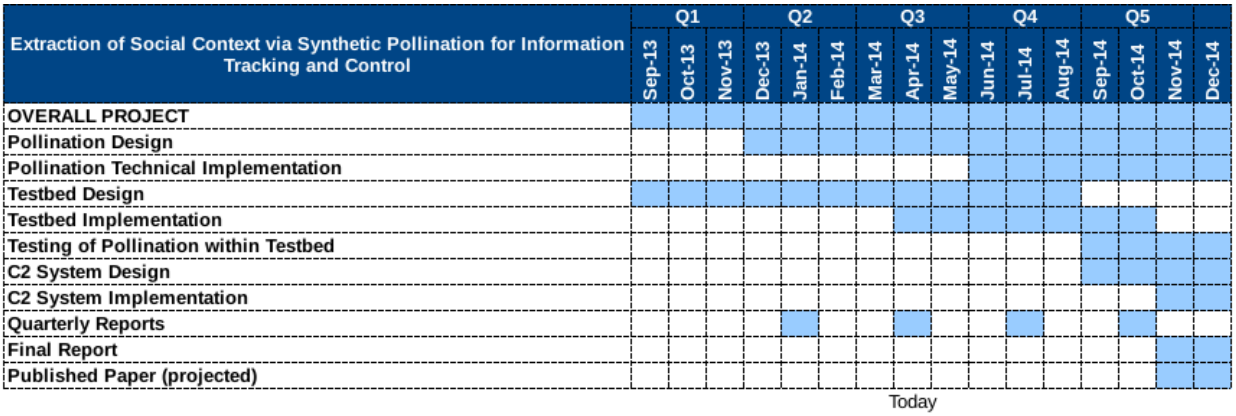


Figure 1: Project schedule