INL/EXT-10-18120

Sophia Proof of Concept Report

Gordon Rueff
Corey Thuen
James Davidson

March 2010

INL/EXT-10-18120

Sophia Proof of Concept Report

Gordon Rueff
Corey Thuen
James Davidson

March 2010

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Office of Electricity Delivery and Energy Reliability
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

ABSTRACT

Researchers at INL with funding from the Department of Energy's Office of Electricity Delivery and Energy Reliability (DOE-OE) evaluated lessons learned from cyber assessments of Industrial Control Systems (ICS) in the laboratory and the field to develop a new tool to assist the asset owner in securing their ICS. This report provides the reader with an understanding of the process used and the development of proof of concept code for this tool. Using field deployments of this proof of concept tool the report concludes with use cases as well as a roadmap for final development of the tool for use in ICS environments.

CONTENTS

FIGURES

ACRONYMS

ABB	Asea Brown Boveri
AIC	Availability, Integrity, and Confidentiality
API	Application Programming Interface
CIA	Confidentiality, Integrity, and Availability
CSV	Comma-Separated Values
DHCP	Dynamic Host Configuration Protocol
ICS	Industrial Control Systems
INL	Idaho National Laboratory
IP	Internet Protocol
IT	Information Technology
NTP	Network Time Protocol
SUID	Set User ID
TCP	Transmission Control Protocol
XML	Extensible Markup Language

1. BACKGROUND

Corporate networks are dynamic in nature where hosts, services, applications, and users are constantly changing. Industrial Control Systems (ICS) use a largely static set of communication pathways, applications, and users.

Corporate networks follow the traditional Information Technology (IT) priorities that follow the Confidentiality, Integrity, and Availability (CIA) Model. Industrial Control Systems reverse these priorities: Availability, Integrity, and Confidentiality (The AIC Model).

Finally, traditional IT systems undergo periodic hardware and software updates in the range of 3 to 5 years. The typical ICS system can have a lifespan of 15 to 20 years or more.

The dichotomy between the two environments limits the effectiveness of traditional IT tools in evaluating the cyber security profile of an ICS. The development of traditional IT tools must address a dynamic environment that likely increases tool complexity. Additionally, these tools often require specialized knowledge to use effectively and may adversely impact the availability of the ICS. Conversely, the ICS environment allows for software designs that are less complex and as a result are easier to learn and use effectively.

2. BASIS FOR DEVELOPMENT

A group of senior Idaho National Laboratory (INL) researchers involved in cyber security evaluations of ICSs and their deployments in the field evaluated these concepts. By working with industry during these assessments and other outreach activities a number of issues were identified:

1. Knowledge of an installation is often incomplete. Topologies are not fully documented, required ports and services are unknown, and the resources required to obtain this information is not always available.
2. Vendors can help with the ports and services for relatively new systems. However, older legacy systems are often unsupported by the vendor or the vendor no longer exists. This makes it the stakeholder’s responsibility to perform this evaluation.
3. Personnel responsible for most installations are dedicated to maintaining the availability of the system, subsequently skills and available resources are focused on this task. Configuration management is often overlooked as the emphasis on “keeping the system running” takes most of their time.
4. Traditional tools for evaluating network topology, identifying ports and services, and cyber security evaluations can be dangerous when used on an ICS.
5. Networking skills are not taught or emphasized, unlike a corporate network environment. In addition corporate priorities (the CIA model) are reversed in ICS networks.
6. Without concrete knowledge of the system configuration, the user cannot optimize the cyber security profile of an ICS.

What is needed is a software tool that:

- Learns about the system using online passive monitoring techniques
- Establishes what components are in a system
- Identifies how the components communicate
- Captures a configuration baseline of the component communications
- Identifies in real time any deviation from the baseline

- Provides information to build quality firewall rules
- Is easy to learn and use.

From this came the concept for Sophia. This conceptual idea was presented to a group of ICS users to validate the idea and the proof of concept development was started.

3. SOPHIA (GREEK FOR WISDOM)

The team of INL researchers wanted to create a tool that would largely automate some of the tasks involved in an ICS assessment and present information gained in an easy to understand format, and as a result impart some the team's wisdom and experience to the users of Sophia. A number of tools have been developed at INL during assessments to fingerprint an unknown system. The fingerprint in the context of Sophia is a list of all conversations on the system. A conversation is a communication pathway between devices on the network. The security of a system is defined by its weakest link. To find the weakest link all links must first be identified. One of the weakest links found in a majority of assessments is undocumented conversations without proper authentication techniques.

Because of the variety of ICS vendors, many of which create their own set of unique protocols for ICS communications, attempts to identify all ICS protocols would be complex and difficult. However, identifying a conversation and then prompting the user to identify the conversation as appropriate or non-appropriate is very useful. This encourages documentation of the system, along with a deeper understanding of the system by the user.

The Sophia proof of concept utilizes a simple Web-based interface that takes very little time to learn. It discovers and presents all of the ICS conversations in straight-forward tables or trees and allows the user to identify each pathway as good or bad. Finally, to allow quick initial use, Sophia supports a fingerprinting mode that classifies all pathways as valid until the user decides that the fingerprint is done. Deviations from this accepted baseline generate alerts that allow the user to maintain awareness, examine validity, and respond appropriately.

4. SOPHIA PROOF OF CONCEPT

Sophia stores information about the network in records of several types. A host record is defined as a unique Internet Protocol (IP) address. A conversation is a broad term that is the compilation of three distinct record types: service, channel, and session. A service record is uniquely defined by a host and service port. A channel record is uniquely defined by a service and another host acting as a client. And a session record is uniquely defined by a channel and a client port. Figure 1 depicts this relationship. Green boxes represent the records stored by Sophia. Yellow diamonds are the unique information that each record contains. In addition to unique information, records store other information such as bytes sent and received.

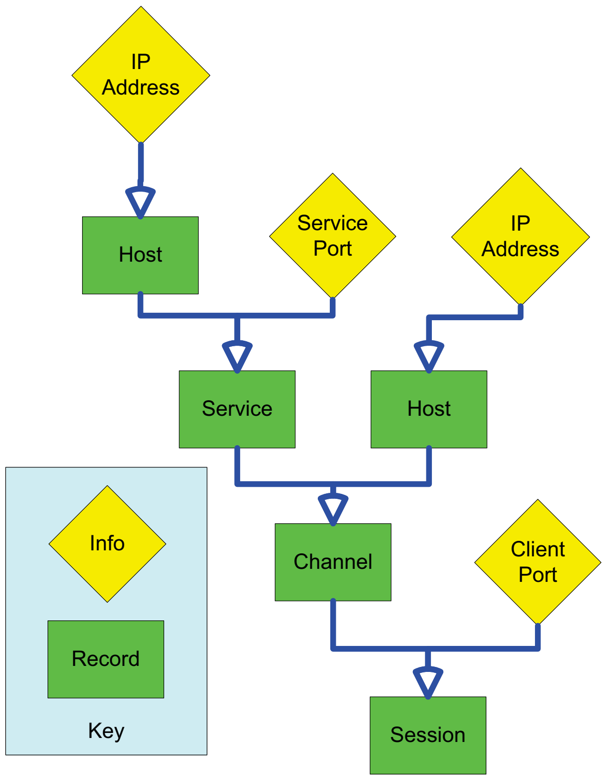


Figure 1. Conversation hierarchy.

Because a session is defined as a channel plus another piece of unique information, there are often many sessions that all reference the same channel. Similarly many channels may all reference the same service. Thinking about it from a top down perspective, a service is comprised of one or more channels, and a channel is comprised of one or more sessions. This view is depicted in Figure 2.

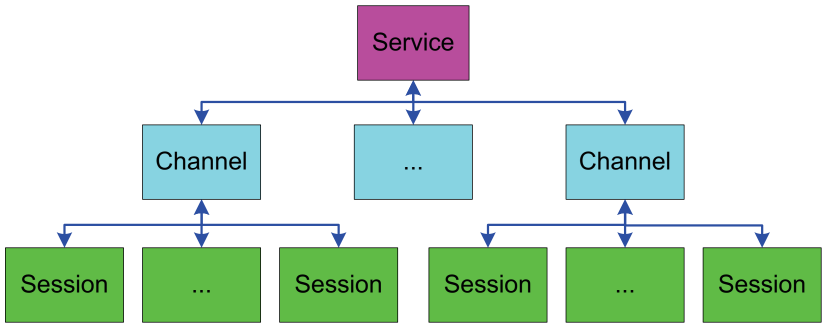
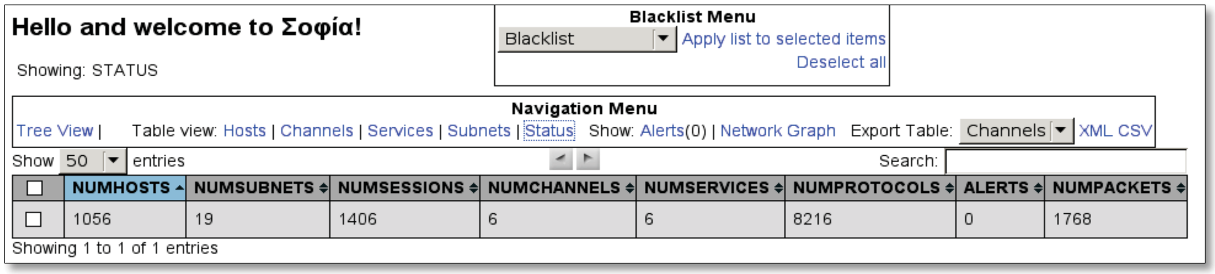


Figure 2. Conversation composition.

4.1 Sophia Web Interface

The Sophia proof of concept provides an Extensible Markup Language (XML) Application Programming Interface (API) for use by software not directly involved in traffic analysis. The graphical user interface facet of the Sophia proof of concept uses this API to create dynamic Web pages for the user. The end result is that Sophia's data can be accessed from a Web browser like Chrome or Firefox. For the purpose of the proof of concept, security was handled by limiting the Web server to localhost only connections.

Figure 3 shows the status page of Sophia. The one row in the table lists the number of items Sophia has detected of each type. For example Sophia has identified 1406 sessions that it combined into six channels. Each channel fits into a different service since there are also six services.


Hello and welcome to Σοφία!
Showing: STATUS
Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Navigation Menu
Tree View | Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
Show 50 entries
Search:

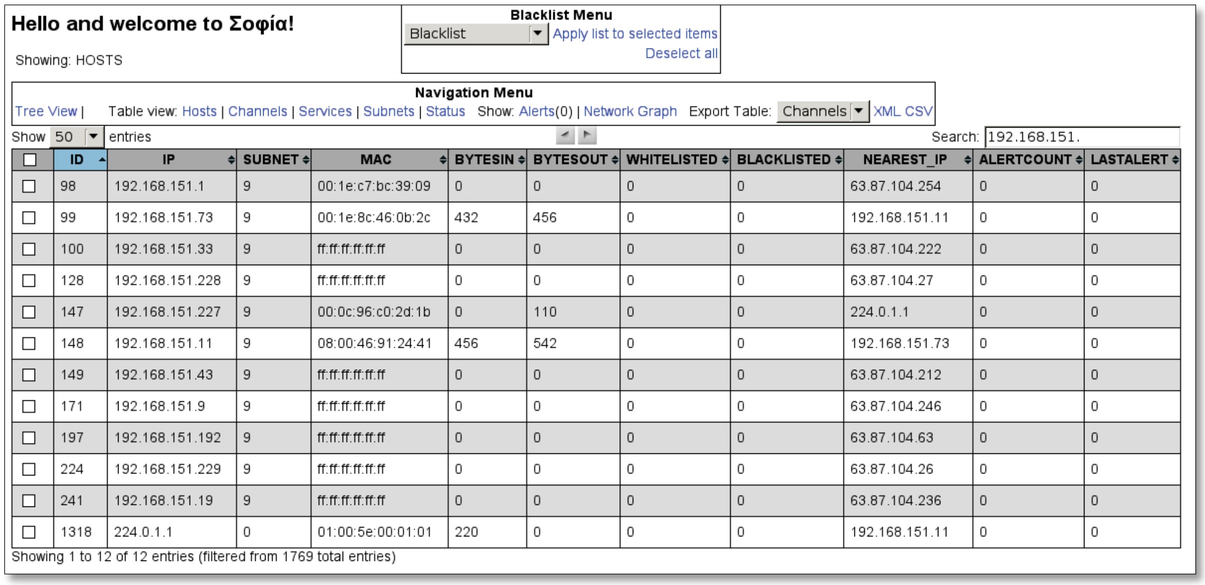
	NUMHOSTS	NUMSUBNETS	NUMSESSIONS	NUMCHANNELS	NUMSERVICES	NUMPROTOCOLS	ALERTS	NUMPACKETS
☐	1056	19	1406	6	6	8216	0	1768

Showing 1 to 1 of 1 entries

Figure 3. Sophia status screen.

4.1.1 Table Views

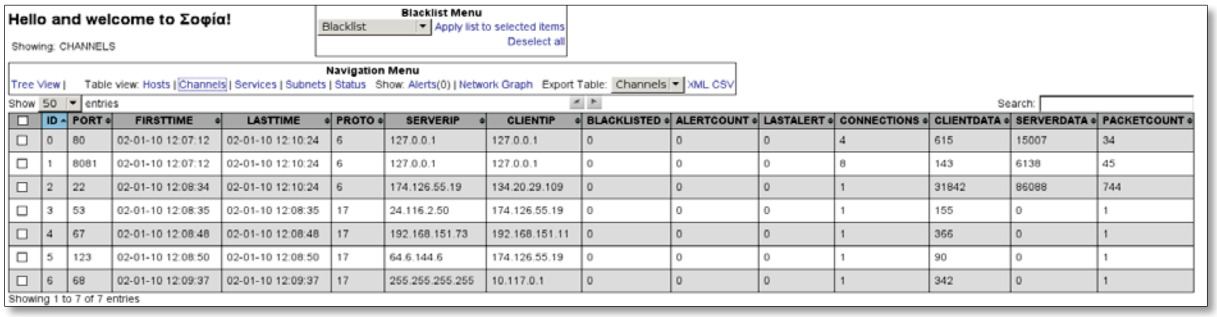
Sophia provides table views for all of its record types. Table view examples for hosts and channels are shown in Figure 4 and Figure 5, respectively. Hosts and channels are probably the most important forms of information kept by Sophia. The host table roughly produces a list of devices found on the network. Channels are more succinct than sessions; for example a single Web browser may produce thousands of sessions even when only connecting to a single server, but all of those sessions to the same server will be collected into a single channel. Channels provide both end points of a conversation whereas a service stores only one end point; thus, a channel can be identified as crossing subnet boundaries.


Hello and welcome to Σοφία!
Showing: HOSTS
Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Navigation Menu
Tree View | Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
Show 50 entries
Search: 192.168.151.

	ID	IP	SUBNET	MAC	BYTESIN	BYTESOUT	WHITELISTED	BLACKLISTED	NEAREST_IP	ALERTCOUNT	LASTALERT
☐	98	192.168.151.1	9	00:1e:c7:bc:39:09	0	0	0	0	63.87.104.254	0	0
☐	99	192.168.151.73	9	00:1e:8c:46:0b:2c	432	456	0	0	192.168.151.11	0	0
☐	100	192.168.151.33	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.222	0	0
☐	128	192.168.151.228	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.27	0	0
☐	147	192.168.151.227	9	00:0c:96:c0:2d:1b	0	110	0	0	224.0.1.1	0	0
☐	148	192.168.151.11	9	08:00:46:91:24:41	456	542	0	0	192.168.151.73	0	0
☐	149	192.168.151.43	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.212	0	0
☐	171	192.168.151.9	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.246	0	0
☐	197	192.168.151.192	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.63	0	0
☐	224	192.168.151.229	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.26	0	0
☐	241	192.168.151.19	9	ff:ff:ff:ff:ff:ff	0	0	0	0	63.87.104.236	0	0
☐	1318	224.0.1.1	0	01:00:5e:00:01:01	220	0	0	0	192.168.151.11	0	0

Showing 1 to 12 of 12 entries (filtered from 1769 total entries)

Figure 4. Sophia's host table.



Hello and welcome to Σοφία!

Showing: CHANNELS

Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Navigation Menu
Tree View | Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
Show 50 entries
Search:

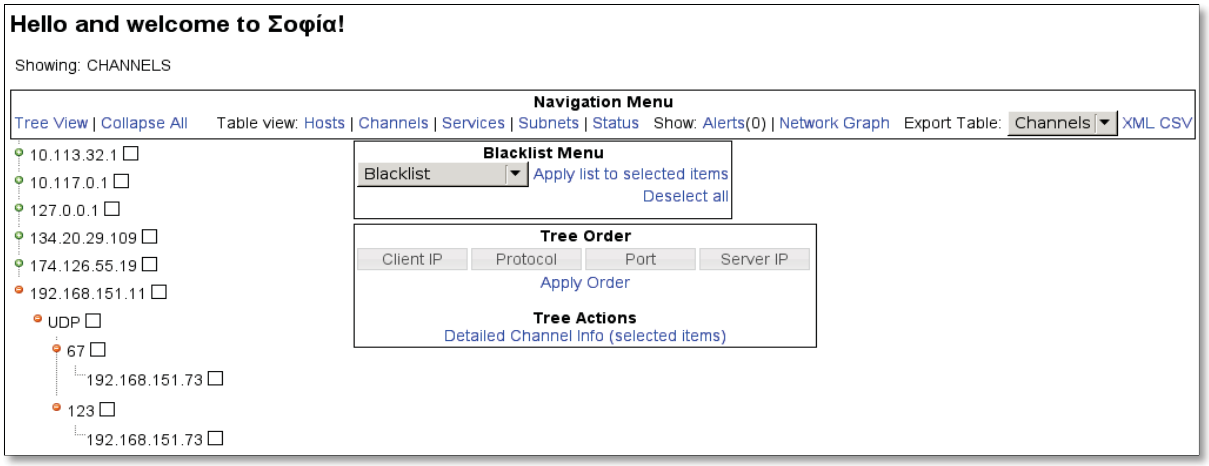
	ID	PORT	FIRSTTIME	LASTTIME	PROTO	SERVERIP	CLIENTIP	BLACKLISTED	ALERTCOUNT	LASTALERT	CONNECTIONS	CLIENTDATA	SERVERDATA	PACKETCOUNT
☐	0	80	02-01-10 12:07:12	02-01-10 12:10:24	6	127.0.0.1	127.0.0.1	0	0	0	4	615	15007	34
☐	1	8081	02-01-10 12:07:12	02-01-10 12:10:24	6	127.0.0.1	127.0.0.1	0	0	0	8	143	6138	45
☐	2	22	02-01-10 12:08:34	02-01-10 12:10:24	6	174.126.55.19	134.20.29.109	0	0	0	1	31842	86088	744
☐	3	53	02-01-10 12:08:35	02-01-10 12:08:35	17	24.116.2.50	174.126.55.19	0	0	0	1	155	0	1
☐	4	67	02-01-10 12:08:48	02-01-10 12:08:48	17	192.168.151.73	192.168.151.11	0	0	0	1	366	0	1
☐	5	123	02-01-10 12:08:50	02-01-10 12:08:50	17	64.6.144.6	174.126.55.19	0	0	0	1	90	0	1
☐	6	68	02-01-10 12:09:37	02-01-10 12:09:37	17	255.255.255.255	10.117.0.1	0	0	0	1	342	0	1

Showing 1 to 7 of 7 entries

Figure 5. Sophia's channel table.

4.1.2 Channel Tree View

The proof of concept provides a channel tree view that allows the user to explore their system by organizing the channels into different trees. Two related questions often arise during an assessment: "What are all of the communications leaving this client?" and "What are all of the communications coming into this server?". If the tree order is set to "Client IP":"Protocol":"Port":"Server IP", then the client tree is expanded for the client in question. The result is Figure 6 that shows Client "192.168.151.11" is using Dynamic Host Configuration Protocol (DHCP) and Network Time Protocol (NTP) services, both from Server "192.168.151.73".



Hello and welcome to Σοφία!

Showing: CHANNELS

Navigation Menu
Tree View | Collapse All Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Tree Order
Client IP
Protocol
Port
Server IP
Apply Order
Tree Actions
Detailed Channel Info (selected items)

```
10.113.32.1 ☐
10.117.0.1 ☐
127.0.0.1 ☐
134.20.29.109 ☐
174.126.55.19 ☐
192.168.151.11 ☐
  UDP ☐
    67 ☐
      192.168.151.73 ☐
    123 ☐
      192.168.151.73 ☐
```

Figure 6. Inspect client communications.

If the tree order is set to "Server IP":"Protocol":"Port":"Client IP", and expanding on the desired server, as shown in Figure 7, it shows that the Server "192.168.151.73" is providing DHCP and NTP services.

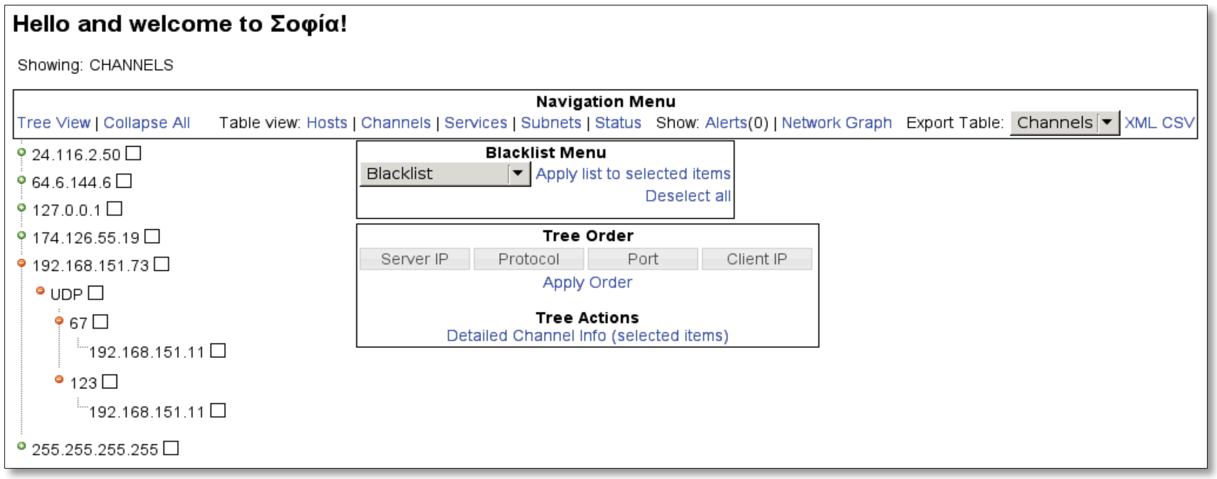

Hello and welcome to Σοφία!
Showing: CHANNELS
Navigation Menu
Tree View | Collapse All Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
24.116.2.50
64.6.144.6
127.0.0.1
174.126.55.19
192.168.151.73
UDP
67
192.168.151.11
123
192.168.151.11
255.255.255.255
Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Tree Order
Server IP
Protocol
Port
Client IP
Apply Order
Tree Actions
Detailed Channel Info (selected items)

Figure 7. Inspect server communication.

Finally, another often asked question is "Which hosts are using a certain protocol?". The tree view can be organized in the order "Protocol":"Port":"Server IP":"Client IP" as shown in Figure 8 to quickly obtain a list of servers servicing a specific protocol. In this case, Hosts "64.6.144.6" and "192.168.151.73" are both NTP servers.

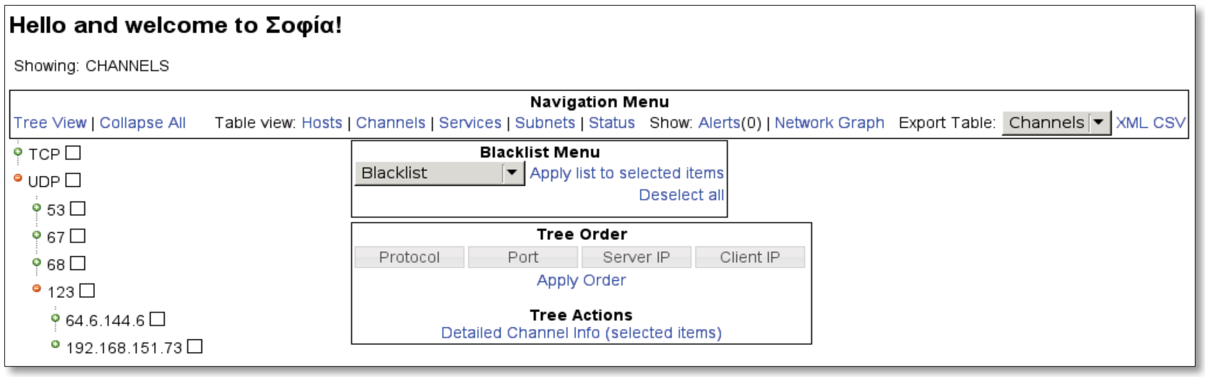
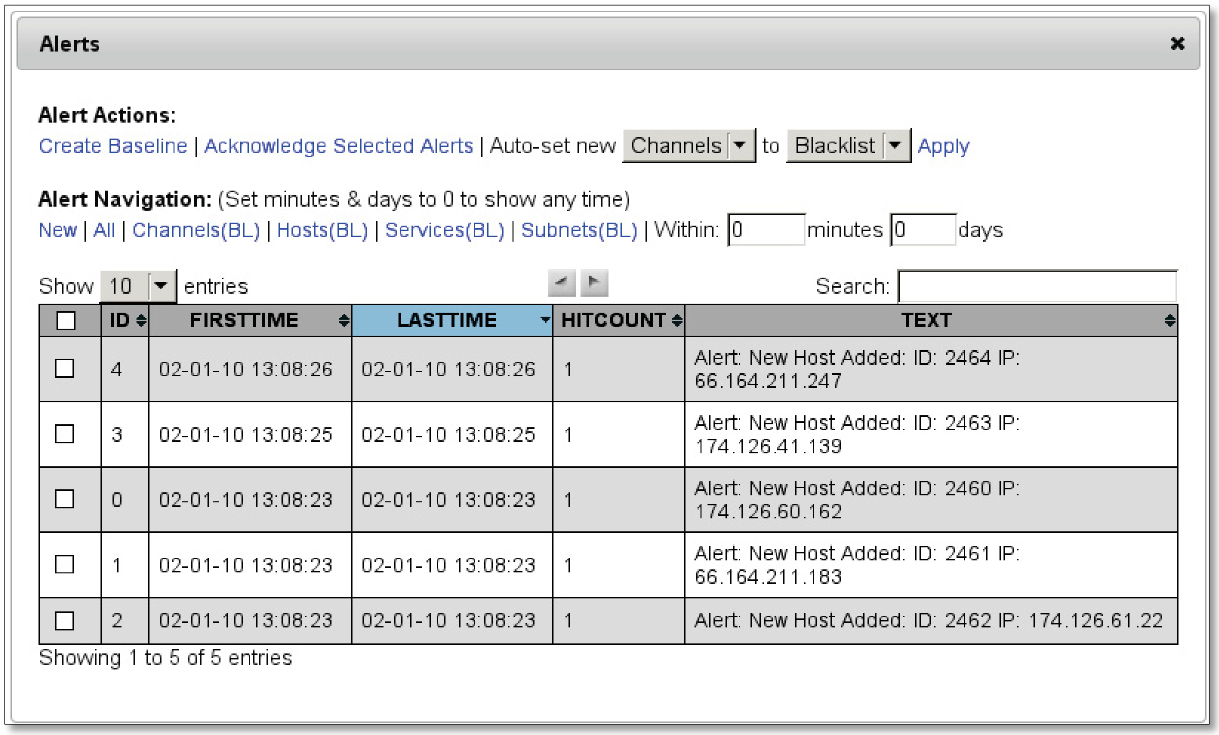

Hello and welcome to Σοφία!
Showing: CHANNELS
Navigation Menu
Tree View | Collapse All Table view: Hosts | Channels | Services | Subnets | Status Show: Alerts(0) | Network Graph Export Table: Channels XML CSV
TCP
UDP
53
67
68
123
64.6.144.6
192.168.151.73
Blacklist Menu
Blacklist
Apply list to selected items
Deselect all
Tree Order
Protocol
Port
Server IP
Client IP
Apply Order
Tree Actions
Detailed Channel Info (selected items)

Figure 8. Inspect protocol communications.

4.1.3 Fingerprinting and Alerts

Complete passive record collection performed by Sophia produces a fingerprint of the known system. After the user operates in fingerprint mode long enough to collect a representative operation of their system, they have the option to create a baseline fingerprint. This puts all current records into a white list, meaning that these communications are acceptable and normal. If a new communication is detected it will be grey listed and generate a one-time alert notifying the user that something new has appeared on their network and they need to address its arrival. Figure 9 shows an example of new host alerts generated by Sophia after creating a baseline fingerprint.


Alerts

Alert Actions:
Create Baseline | Acknowledge Selected Alerts | Auto-set new Channels to Blacklist Apply

Alert Navigation: (Set minutes & days to 0 to show any time)
New | All | Channels(BL) | Hosts(BL) | Services(BL) | Subnets(BL) | Within: 0 minutes 0 days

Show 10 entries Search:

☐	ID	FIRSTTIME	LASTTIME	HITCOUNT	TEXT
☐	4	02-01-10 13:08:26	02-01-10 13:08:26	1	Alert: New Host Added: ID: 2464 IP: 66.164.211.247
☐	3	02-01-10 13:08:25	02-01-10 13:08:25	1	Alert: New Host Added: ID: 2463 IP: 174.126.41.139
☐	0	02-01-10 13:08:23	02-01-10 13:08:23	1	Alert: New Host Added: ID: 2460 IP: 174.126.60.162
☐	1	02-01-10 13:08:23	02-01-10 13:08:23	1	Alert: New Host Added: ID: 2461 IP: 66.164.211.183
☐	2	02-01-10 13:08:23	02-01-10 13:08:23	1	Alert: New Host Added: ID: 2462 IP: 174.126.61.22

Showing 1 to 5 of 5 entries

Figure 9. Web alerts popup.

At this point the user should investigate the new communication and decide if it is appropriate or inappropriate for the ICS. If the communication is appropriate the user should add the records to the white list and acknowledge the alert. If it is inappropriate the user should black list the record and acknowledge the alert, followed by fixing the problem that caused the alert.

Here are some reasons new communications may appear after the fingerprint is created.

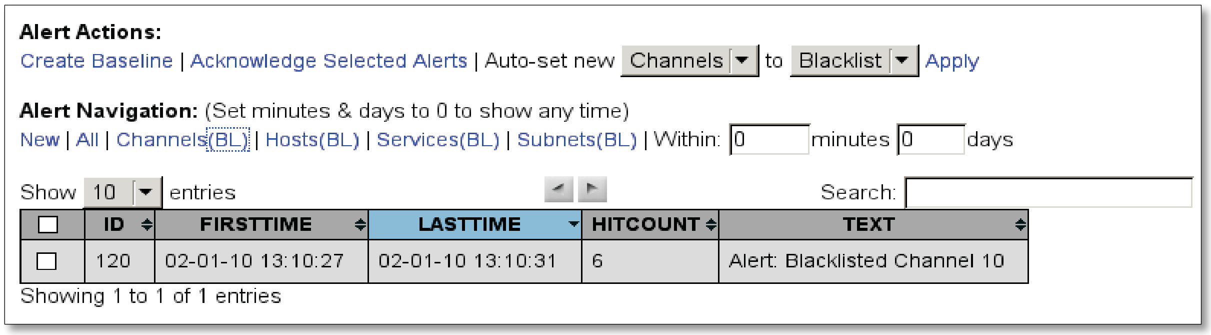
Appropriate:

- New Device. A new database server has been added to the network and should be documented as such.
- Failover. The primary and secondary servers have switched roles resulting in new traffic patterns.
- Software Update. The vendor has just updated the real-time database and a new Transmission Control Protocol (TCP) port shown up on the server.

Inappropriate:

- Device Failure. A server's hard drive is failing causing the network configuration file to be lost and the IP address of the server has changed as a result.
- Intruder. An intruder has gained a foothold on the network and is scanning the network as part of their reconnaissance.
- Windows. Windows has decided to reactivate automatic updates and it is trying to reach Microsoft's update servers.

If the user white lists the record then nothing else is required. If, however, the user black lists the record, then Sophia will begin generating alerts every time that record is reconfirmed with traffic on the network. Figure 10 shows an example of an alert generated from a black-listed channel. *Since its black* *listing, Sophia has detected six packets as part of that channel.* Black-list alerts are useful to notify the user that a problem has reoccurred after the first event.


Alert Actions:
Create Baseline | Acknowledge Selected Alerts | Auto-set new Channels to Blacklist Apply

Alert Navigation: (Set minutes & days to 0 to show any time)
New | All | Channels(BL) | Hosts(BL) | Services(BL) | Subnets(BL) | Within: 0 minutes 0 days

Show 10 entries Search:

	ID	FIRSTTIME	LASTTIME	HITCOUNT	TEXT
☐	120	02-01-10 13:10:27	02-01-10 13:10:31	6	Alert: Blacklisted Channel 10

Showing 1 to 1 of 1 entries

Figure 10. Blacklisted alerts popup.

5. SOPHIA DEPLOYMENT

The Sophia proof of concept was deployed at two utilities and one vendor site. The two utilities were picked to bracket the range ICS network complexity. One utility had a simple network and the other utility had a highly segmented network. The manner in which the utilities used Sophia led to the deployment at the vendor site.

5.1 Utility 1: Idaho Falls Power

Sophia's first deployment was at the small Idaho Falls Power utility. This initial deployment quickly identified three items that needed to be addressed, one of which was addressed during continued proof of concept development. All three items arose because of the utility's need to restart the server running Sophia due to system upgrades.

First - Each time the server was restarted, someone had to log on to the server and manually start Sophia. The proof of concept does not include a run level daemon that gets started during the booting process. Sophia needs to have this capability.

Second - Due to Sophia's adaptation from a run once tool used as part of an assessment, Sophia did not record state of operation or restore the state upon its next execution. These capabilities were added to the proof of concept to allow Sophia to run without reentering baseline mode every time Sophia starts.

Third - Since Sophia uses privileged operations, in particular the opening of a raw socket, it had to run with root's permission level. For security reasons, Sophia will need to automatically drop it privileges once it attains the necessary resources. Other security options that will need to be evaluated are using a Set User ID (SUID) version similar to how Wireshark gets its permissions and a chroot environment daemon that will isolate Sophia from other processes.

After using Sophia, Idaho Falls Power had the following comment: "Idaho Falls Power was given the opportunity to test Sophia over the last few months and found it to be a great asset to our utility. Sophia adds the characteristics of a full-time employee. With Sophia running 24 hours a day, it continually monitors network traffic and flags anything that is out of the ordinary. We found it to be very robust, easy to install, and most importantly easy to use. Sophia would be a great addition to any network that has a need to monitor critical traffic."

5.2 Utility 2: Austin Energy

Austin Energy has a very complex and segmented network that resulted in very different lessons. These lessons ranged from how data needs to be collected, how much information is stored, and how information will get disseminated to the necessary people.

The network architecture at Austin Energy is segmented so that running span lines from all networks back to Sophia is not a realistic solution for data acquisition. However, the high segmentation level results in many firewalls and all the firewalls were already logging network activity to Syslog servers. This situation resulted in developing a Syslog interpreter for Sophia that can be configured to read different Syslog messages and extract Sophia relevant information from them.

While the proof of concept solution was to integrate a new capability directly into Sophia, the long-term solution for handling diverse environments will need to be the development of a standardized interface to Sophia's core. This standardized interface will allow for the breakout of interpretation tasks into separate processes that then send the results into Sophia's core. Additionally, this will allow Sophia to be distributed when necessary based on system architecture.

Austin Energy also wanted Sophia to be able to report alerts via Syslog, since they have many capabilities based around Syslog. Another utility may need the messages in the form of an e-mail, and yet another utility may need messages posted on an IRC channel. Further development on Sophia will need to result in an output interface that can be adapted based on system architecture.

Since Sophia's Syslog input layer was developed after initial deployment, the utility started using Sophia in a limited fashion. Simultaneous with deploying Sophia, a new control system arrived at the utility. The utility used Sophia to fingerprint the new system and that fingerprint was used to help develop the firewall rules necessary for further segmentation of the control system. This particular use led the team to contact a vendor to inquire about fingerprinting a system before it arrives at a utility.

Austin Energy used a lot of scripting to help manage the complex ICS network. For Sophia to interact smoothly within a scripted environment, along with supporting various output capabilities, a command line feature is needed. Not all utilities want or need scripting, Sophia development of a common command library that various interfaces, such as a command line and a graphical interface, can use to interact with Sophia's core is required.

Austin Energy's main compliment of Sophia was that it automated a task that was already being done at their site and as such it freed up man power to work on other tasks. The main complaint about Sophia was regarding its proof-of-concept nature, and as such it is not ready to be widely used and integrated. Two features Austin Energy would need before wide spread use of Sophia would be data storage for event re-creation and a zero downtime design. In the case of data storage, this utility would need to be able to trace back the exact packets or Syslog messages that caused an alert. Without this feature the utility would not be comfortable using the tool since it could lead to a dead end investigation of an alert. The zero downtime is simply the incorporation of common control system design that includes running with zero downtime.

After using Sophia, Austin Energy had the following comment: "Austin Energy is pleased that INL has proactively begun development of its Sophia tool, which will enhance our ability to determine and document our industrial control system network infrastructure for hardening. This functionality enhances our ability to properly secure and monitor our SCADA/EMS and DMS systems."

5.3 Vendor: ABB Network Manager

After working with Austin Energy, INL approached Asea Brown Boveri (ABB) to explore using Sophia on an EMS control system still at the vendor. Even before deploying Sophia at ABB a new feature was needed. The vendor would need Sophia to produce a common file format for disseminating fingerprint results to their customers. The development team added an export feature that allows the fingerprint to be exported to a Comma-Separated Value (CSV) file. The lesson here is similar to that of Syslog output request; Sophia needs to be able to support many different types of output to support the diverse environments of ICSs.

ABB identified several potential use cases for Sophia in their environment. The information contained in the fingerprint generated by Sophia has been requested by several customers and is difficult to generate accurately for each variation of their system that gets released. Quality assurance could also use Sophia in several ways. First, Sophia could help identify problems in the systems during development and while getting ready to deploy a new system. Sophia could also help track changes between revisions of a product that would help customers update firewall rules when applying system updates. Sophia could also help track changes between the in-house reference systems and the deployed systems. Finally, Sophia could be included as part of an audit service provided by the vendor to measure the health of a deployed system. ABB could see a number of potential uses for Sophia.

Sophia also hit several problems during this deployment. The control system by this vendor, like many other ICS vendors, use a significant number of dynamically allocated service ports. This meant that Sophia created a new service record each time the service restarted and would break the current fingerprinting concept. Sophia will need to develop a smooth means of handling dynamic service ports to properly fingerprint these systems.

After using Sophia, ABB had the following comment: "While ABB's experience with Sophia to date is somewhat limited, we plan to integrate it into our system build-up and checkout process flow. We view Sophia as a valuable tool to understand the network traffic in our systems and to indicate if components of the system are improperly configured. One of the strengths of Sophia is that it collects its information for as long as we choose. It is not a short-term test that can miss significant traffic associated with infrequently-used processes."

6. FEEDBACK SUMMARY

A lot of useful feedback was gathered from the deployment at various sites:

- A command line interface for Sophia control would be good for interacting with remote servers running Sophia, or setting up scripts to interact with Sophia.
- Creation of various output records and streams such as Syslog during execution would allow easy additional processing of Sophia records.
- A distributed architecture is necessary for some systems that are too disparate to monitor using a single server.
- For the purpose of event re-creation, Sophia needs to be able to store at least some history of the data it receives. The goal of the stored data being to allow the user to track the cause of alerts and possibly help during a forensics investigation.
- Sophia needs to run with zero downtime, meaning that redundancy must be considered in its design. Also part of zero downtime is the need for Sophia to support being started as a service at boot time.
- Sophia needs to develop a means to handle dynamic service ports because these currently break the fingerprinting system or create a very large fingerprint that is not a good representation of the actual activity.
- The forwarding of received data was a desired feature, though not a requirement. For example, allow one network interface card to received span traffic and recreate the span traffic on another card so that another server has access. On many switches, span ports are a limited resource.
- Security which was overlooked and willfully omitted in the proof of concept will need to be designed into Sophia from the start. Three security related items identified during trial use are: privilege dropping, SUID executable instead of running Sophia as root, and running Sophia in a chroot environment.

7. HIGH-LEVEL SOPHIA DESIGN

With the proof of concept, all input and output of record information is handled by the Sophia backend, resulting in additional code paths for every input and output method. For example, there is code that handles libpcap processing, other code that handles Syslog processing, and additional code that handles XML processing. This means that the Sophia core has to be rewritten every time Sophia gets a feature addition. This is a poor design for an application that, based on deployment feedback, needs to be very flexible and support many different formats, some of which are proprietary.

Figure 11 shows the high-level design of the Sophia proof of concept. It shows that each input and output capability gets put into the same code base as the main Sophia process. In contrast, Figure 12 shows how Sophia will need to concentrate on developing three core libraries: Record Protocol Library, Command Library, and Record Library. The record protocol library will be an IP-based library for transferring Sophia records between processes. The command library will be used for controlling the behavior of Sophia, for example black listing certain channels or exiting fingerprint mode. The record library will be used for manipulating the on-disk records Sophia uses for state saving and restoring.

The figures use magenta to highlight the Sophia core, yellow for input code, blue for output code, and green for redundancy code. The magenta and green parts of the figures are meant to be fairly static so that updates are not needed frequently. The yellow and blue parts show the input and output code that will be constantly growing to support new input and output formats. The proof of concept has the problem that yellow and blue blocks are inside a magenta block thus making the magenta block not static in design or code.

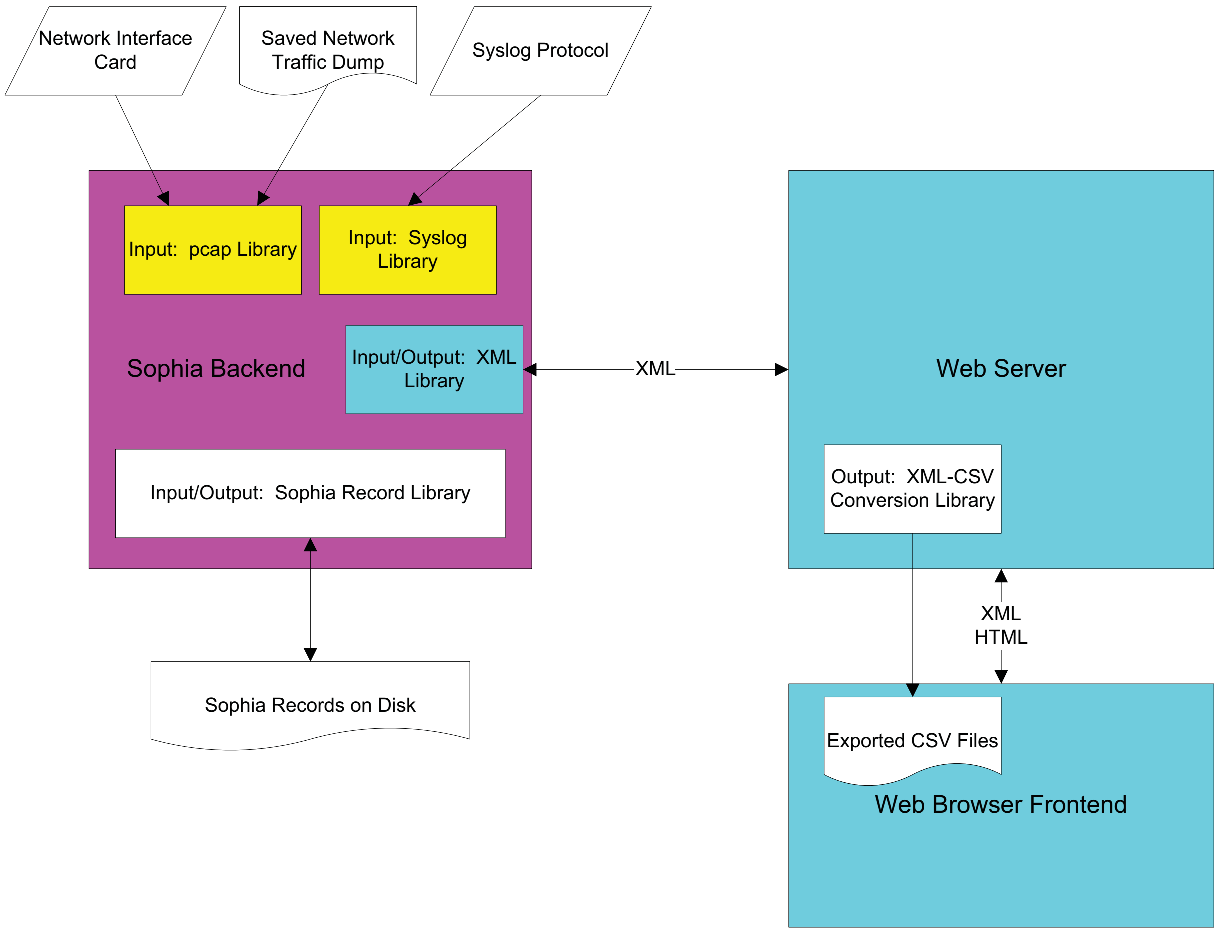
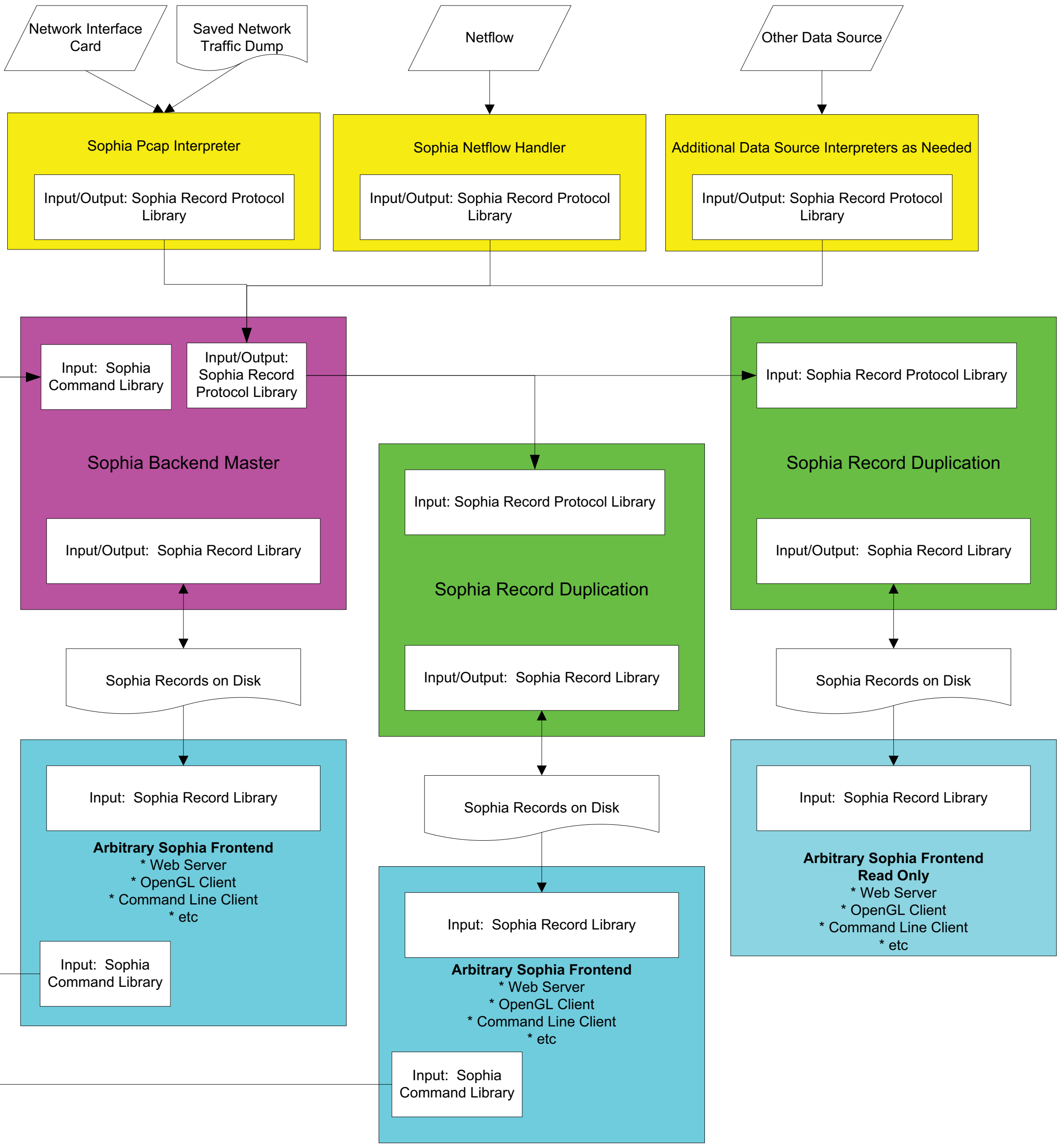


Figure 11. Design overview of Sophia proof of concept.



Network Interface Card
Saved Network Traffic Dump
Netflow
Other Data Source
Sophia Pcap Interpreter
Input/Output: Sophia Record Protocol Library
Sophia Netflow Handler
Input/Output: Sophia Record Protocol Library
Additional Data Source Interpreters as Needed
Input/Output: Sophia Record Protocol Library
Input: Sophia Command Library
Input/Output: Sophia Record Protocol Library
Sophia Backend Master
Input/Output: Sophia Record Library
Input: Sophia Record Protocol Library
Sophia Record Duplication
Input/Output: Sophia Record Library
Input: Sophia Record Protocol Library
Sophia Record Duplication
Input/Output: Sophia Record Library
Sophia Records on Disk
Sophia Records on Disk
Sophia Records on Disk
Input: Sophia Record Library
Arbitrary Sophia Frontend
* Web Server
* OpenGL Client
* Command Line Client
* etc
Input: Sophia Command Library
Input: Sophia Record Library
Arbitrary Sophia Frontend
* Web Server
* OpenGL Client
* Command Line Client
* etc
Input: Sophia Command Library
Input: Sophia Record Library
Arbitrary Sophia Frontend
Read Only
* Web Server
* OpenGL Client
* Command Line Client
* etc

Figure 12. Anticipated design overview of Sophia.

This design will allow additional input and output functionality to be added to Sophia without alteration to the core Sophia code. For instance a Syslog input program would be written to convert Syslog data into Sophia records that would be transmitted using the record protocol library. The Sophia core would never need to change based on the input source. Similarly, a new output format could be supported by simply writing a program that can interpret Sophia's records on disk.

At first view the proposed design appears more complicated, but it achieves many simplification goals. First the pcap, Syslog, Netflow, etc., parsers do not need to interact. The same thing applies to the various output streams; they also do not need to interact. During installation, only the pieces necessary for that installation need to be installed. A system that does not use Netflow, for example, will never be touched by the Syslog code and processes. This design should reduce coding errors and the attack surface of Sophia.

The development of Sophia needs to concentrate on solidifying the magenta and green boxes of Figure 12, which will require significant changes to the proof of concept code. In addition, the pcap and Syslog code will need to be extracted into their proper yellow boxes and another yellow box will need to be added to support Netflow. Finally two blue boxes need to be developed: a bare bones command line interface for controlling Sophia, and a more eye-pleasing OpenGL interface to accomplish the same thing plus experiment with new ways to display the available information in meaningful ways.

Before future development can begin, the Sophia protocols will need to be designed with built-in security, rather than as an afterthought. After the current capabilities of the proof of concept are properly modularized, the core Sophia code needs to expands its fingerprinting capabilities. For example, communication periodicity needs to be tracked along with communication correlation. Periodicity would track how often certain communications occur, provide time span summation of the occurrences and provide an alert when a communication fails to respond within a predefined dead band of periodicity. Correlation would detect communications and devices that are related, such as server redundancy.

7.1 Use Cases

The primary use case of Sophia was to provide a real-time application that fingerprints a running ICS and monitors that ICS for deviation from the fingerprint. During deployment, several other use case possibilities have arisen. Sophia can:

- Monitor an ICS and alert based on deviation from the established fingerprint
- Monitor an ICS system during deployment and use the conversation records as a basis for firewall rules during integration
- Fingerprint an ICS system at the vendor and then monitor that system during deployment for changes
- Fingerprint the QA system at the vendor and compare that fingerprint to a deployed system to help identify changes in the deployed system
- Use fingerprints from Sophia to program switches, routers, and firewalls
- Harden ICS components by disabling unnecessary services that have been identified by Sophia.

8. CONCLUSIONS

The Sophia proof of concept had numerous tests to pass. The foremost of these is the need to vet the usefulness of Sophia with the future users of Sophia, such as utility companies. These utility users found applications for Sophia that were not foreseen. As a result, Sophia was also vetted at a vendor site. The vendor continued to identify additional potential uses for Sophia.

Sophia needed to passively monitor and learn about the components and communication pathways in an ICS network. Sophia monitors a span port using libpcap or parses Syslog records to learn about the devices and communications on an ICS network.

The proof of concept needed to capture a baseline fingerprint of a system and in real time detect deviations from the fingerprint resulting in alerts for the user. This was accomplished through an easy to use interface that supports a one-click method of baselining a system.

Sophia needed to provide information to build quality firewall rules. Sophia provided several ways to inspect the fingerprint to facilitate firewall rule development including a tree view and CSV file exportation for offline analysis.

The proof of concept proved Sophia’s potential usefulness to the industry but also identified many items that will need to be changed before Sophia is ready for industry adoption. The process INL used to develop a low-cost proof of concept and receive feedback from the users early provided a low engineering risk roadmap for the final development of Sophia to meet the diverse needs of the ICS community.