



# ***Information System Security Operation***

## **Automatic Generation of Protocol Analyzers**

### **Support For A Growing And Evolving Set Of Network Protocols**

#### **Overview**

Network protocols and their implementations present numerous problems from a software engineering life-cycle perspective. IETF protocols, such as TCP, IP, IPSec, HTTP, BGP, and many others are specified inconsistently. Protocol specifications, especially in regard to behavior, are often defined with text-based English prose, and subject to varying interpretations. IETF RFC 2234 describes an Augment Backus-Naur Form (ABNF) for use in formally specifying network protocols, but not every protocol in current use is defined by such a formal specification.

Protocol Analyzers, designed to capture, decode and monitor network traffic, are inconsistently implemented. These software tools are typically implemented as large, hand-crafted bodies of code, subject to individual programmer interpretation of the protocols. Security mechanisms in firewalls and intrusion detection systems suffer from the same problems, which also results in inconsistent implementations that are exploitable by knowledgeable attackers. The overall result of this current state of affairs is to make large suites or families of protocol implementations highly duplicative, expensive to maintain, and extremely expensive to secure. The goal of the AGPA project is to efficiently generate efficient network protocol analyzers. These automatically generated protocol analyzers would support a large (growing and evolving), set of protocols at multiple layers in the ISO network stack. As such, these implementations would be especially well-suited to the purposes of network monitoring, and for incorporation within network intrusion detection and intrusion prevention systems. Moreover, the automatic protocol generators readily and transparently enable such a suite of protocol implementations to be migrated to multiple platforms without incurring repeated development costs for each protocol on each

platform. Instead, only the automatic protocol compiler generator needs to be ported.

#### **Solution**

Our objective is to investigate and implement a representative suite of protocols among layers two through seven of the OSI protocol stack, employing BNF and formal specifications coupled with automatic code generation techniques. Under this project, we will develop, demonstrate, and release an automatic protocol generator as well as a small, but representative suite of formal protocol specifications that, when input to the automatic protocol generator, are compiled into efficient table-driven implementations. Our implementations will be complete, in the sense that they address lexical analysis, grammar-based parsing, and finite state engine transitions in a consistent, systematic fashion.

SPARTA makes these innovative claims for the AGPA project:

- Separates protocol specifications from low-level implementation mechanisms.
- Separates specifications, of lexical analyzers, grammars, and finite state machine specifications enables a separation of concerns that facilitates optimization of each layer separately, and more completely than possible with current technology.
- Enables protocol analyzers to stratify their analysis, re-using the same table-drive engineers for all layers.
- Enables the application of 35 years of compiler research, embodied within tools such as Flex and Bison, augmented to handle protocol specification notations currently beyond the capability of these tools.
- Seamlessly incorporates modeling of protocol state engines with a labeled transition system analyzer to provide

---

This work was sponsored by DARPA and performed by McAfee Research, which is now the Security Research Division of SPARTA.

<http://www.issp.sparta.com/research>

## Automatic Generation of Protocol Analyzers

Support For A Growing And Evolving Set Of Network Protocols

detection of protocol fairness, progress, and deadlock constraints during the protocol design process.

- Enables the formal description of protocol exceptions and responses, enabling parser error recovery to deal with truly unanticipated errors.
- Enables network protocol implementations to respond to the problems encountered at the appropriate layer within the system.

The overarching goal of this effort is to develop an approach for efficiently generating efficient network protocol analyzers, for use in network monitoring, network intrusion detection, prevention and response, and other network-centric applications. Over the course of the six month effort, we will investigate, design, implement, and experiment with automatic protocol generation technology. This technology will support the automatic implementation of network protocol stacks using ABNF grammar specifications augmented with semantic action descriptions.

A paradigm shift in protocol stack implementation – for both protocol analyzers and end-host stacks – is needed, or the current dismal state of networking implementations will

continue to degrade. To move the technology base forward, we advocate a shift to implementing protocols more formally, by using context free grammars to describe both the frame format of messages transmitted over the wire, and end-system protocol behavior. All of the syntax, and a large part of the semantics of a network protocol, would be specified in this manner. Frame formats, required for marshalling and unmarshalling packets to and from the network transmission media, would be directly represented by lexical analyzer and grammar descriptions.

In turn, classic approaches to translation of these descriptions into efficient table driven automata, analogous to those used for decades in standard Unix tools such as lex and yacc, would provide production ready implementations of protocol stacks. Moreover, extended grammars would describe end-system protocol behavior – capturing the semantics of legal protocol frame sequences and state transitions, automating the generation of nearly the entire protocol stack implementation. Grammars would enable the handling of exceptions, errors and malicious non-compliance with specified protocol behavior.

