Industrial Intrusion Detection System

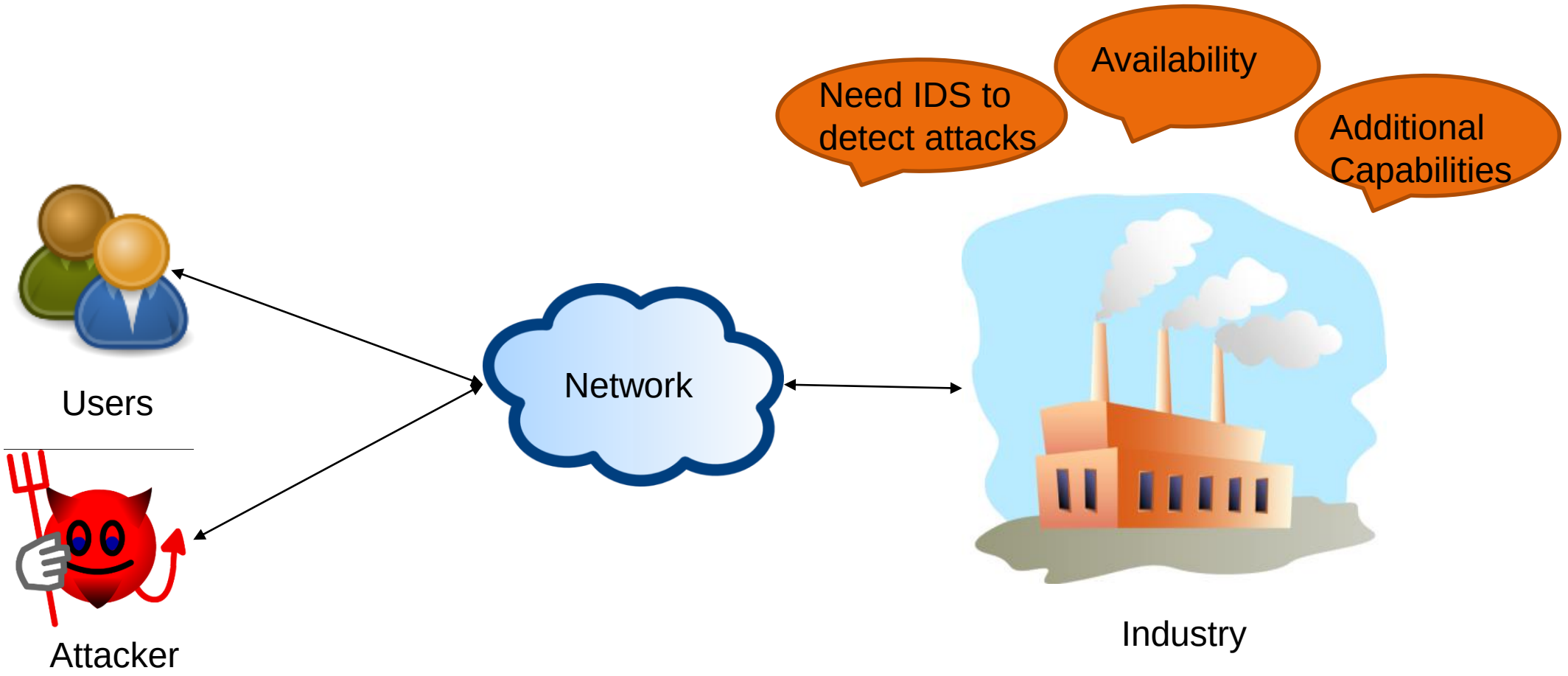
Shubham Bhardwaj
Fraunhofer Institute For Secure Information Technology SIT & Technische Universität Darmstadt

First Presentation For Thesis Registration

Fraunhofer SIT

Problem Definition

Industrial Intrusion Detection System



Contents

Industrial Intrusion Detection System

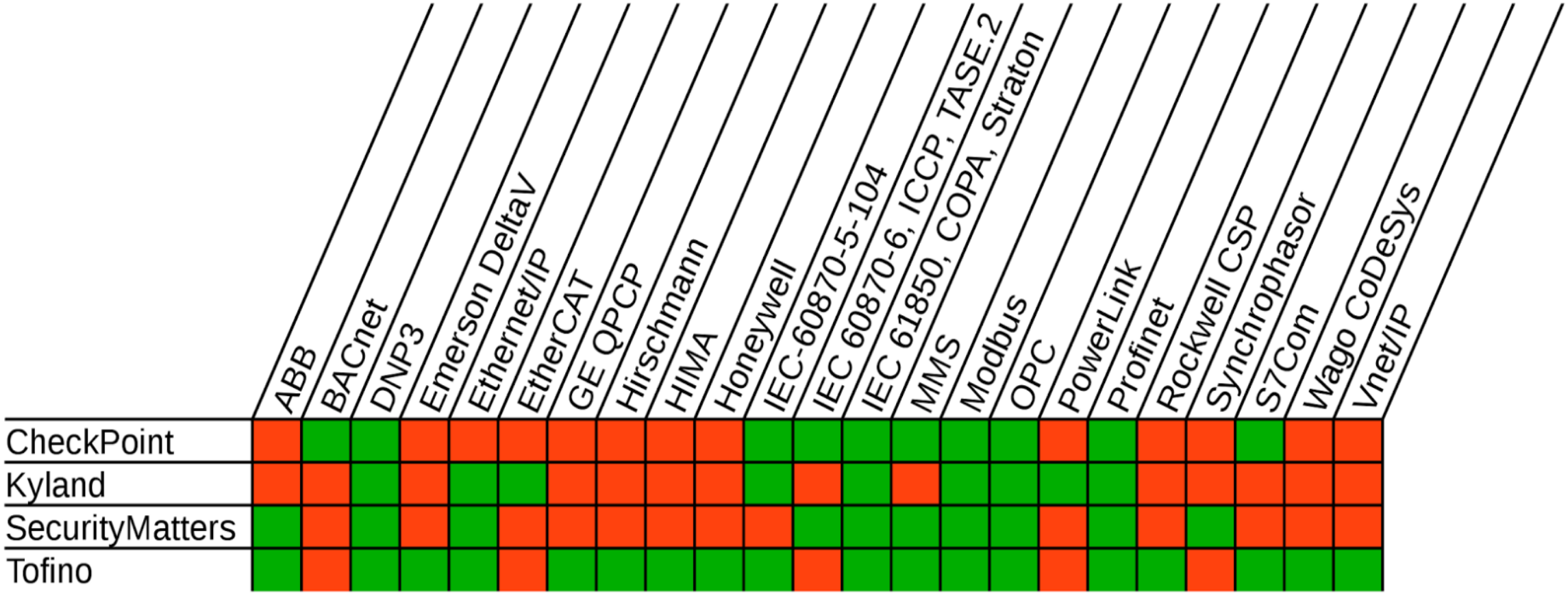
Industrial IDS
Motivation

Industry IDS needs additional capabiltities for examining data streams due to diversity of the protocols.

Main Focus:

- Building parser based on parser definiton language for most industrial protocols.
- Extend these languages to interpret the values process by these languages.
- Generate alerts on interpreted values.

Industrial Protocols



Fraunhofer SIT

Protocol parsing

- Each protocol has a buffer message which is a small logical record of information.
- From this information we can generate events.
- We can create logs or notification from the generated events.
- Protocol parsing is basically looks at the incoming data and interprets it according to the need of the user or standards.

.

Protocol Parsing Languages

- P4
- BinPAC with Bro
- Spicy/binPAC++
- GAPAL
- Frentic
- NAIL
- CLICK
- NETKAT
- PROLAC
- PacketC
- MAPLE
- CPPPO

P4

- **Released in 2013**
- **Based on C or Python**

Main Goals:

1. Protocol independence
2. Target Independece
3. Reconfigurability

It supports Controller/Switch architecture.

Protocol Independence

- The switch is not attached to a particular packet formats.

Instead Controller can

1. Describe packet parser to extract header fields with specific names and types.
2. It also specify a collection of typed match+action table that process these headers.

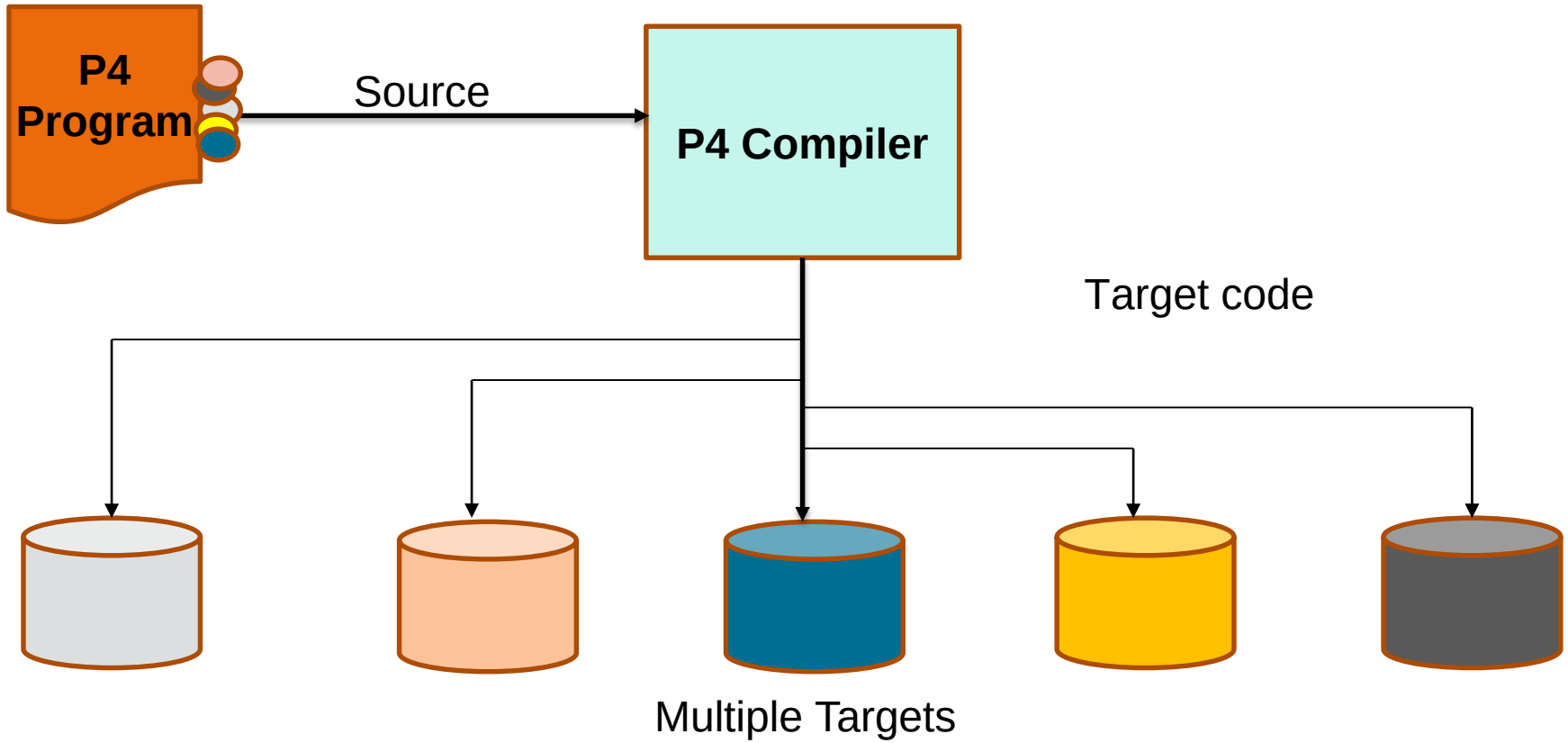
Reconfigurability

Controller can reassign the packet parsing and processing in the field of the header.

Target Independence

1. The Controller programmer should not need to know the the details of the target switch.
2. It can specify everything from high performane ASIC to Software Switch.

Target Independence



Code Snippet:

```
parser anyprotocol {
  extract(anyprotocol); //takes the incoming pipestream and define it
  return select(latest){ case 0x8100: x; default:ingress // Other cases
} }
parser x {..}
Control ingress{
  apply(x_match);
  apply(forward)
}
```

BinPAC with Bro

- BinPAC is a declarative language which is based on c++.
- It also supports the controller/Switch architecture.
- It works in the conjuction of bro network security monitor but not tied to it.
- It takes care of all the low-level tasks, such as byte-order handling, appliction-layer fargment reassembly and support for debugging.
- Build ‚types‘ to represent logical data structures.
- Generally targets application-level traffic analysis.
- Only solves the syntax problem not semantics.
- But it requires substantial effort on specifying new protocol.

Code Snippet:

```
Analyzer <ContextName> withcontext {
… context members…
}  //pointer to the top level analyzer and connection definition
Connection <analyzername>{
upflow; downflow;
} //entry point to the analyzer
Type <tapeName> = <compositor or primitive class>
{
cases
}; //describe structure of a bytesegment
```

Spicy/BinPAC++

- Spicy is a domain specific language which is based on c++.
- It is not similar to Binpac instead it is an extended version of BinPAC, It integrates syntax and semantics into a unified processing model.
- It analyzes both network protocols and file formats
- Spicy toolchain built top on hilti(which is an intermedite language for traffic inspection)
- Binpac++ can work standalone.
- Debugging Support.
- It can work outside of Bro for example with wireshark.
- But still in prototype state not production ready.

Code Snippet:

```
Module anyprotocol;  //Starts with module name.
import Binpac; //Library
export Type Message = unit {
  op: uint<16>;  //parse 2 byte data, we can increase according to the need
on %done{
  print self.op;
}
}; //save as anyprotocol.pac2
grammar anyprotocol.pac2;
protocol analyzer anyprotocol over TCP/UDP;
parse with anyprotocol: Message,
port ../tcp/udp;
```

GAPAL

- It is an high level language evaluated by c++ interpreter.
- It is a self contained system, handles both protocol parsing and traffic analysis but both systems runs on very low speed.
- Gapa was designed to satisfy three main goals that are safety, real-time analysis and response, and rapid development of analyzers.
- To avoid crashes or buffer overruns due to memory errors, GAPAL is strictly typed, with boundary checks on array accesses and no dynamic memory allocation.
- No dynamic memory allcation also means restriction to implement certain kind of logics.
- It has more overhead and can can process very low network traffic in compare to other languages.

Code Snippet:

```
protocol anyprotocol { transport = (port no./TCP);
/* Session variables */ int32 content_length = 0;
bool chunked = false; bool keep_alive = false;
/* message format specification in BNF-like format */
grammar { WS = "[ \t]+"; CRLF = "\r\n";
%%
```

NAIL

- Based on c.
- It uses a single grammar to define external format and internal representation to avoid the vulnerabilities like Android master key bug.
- Can handle complicated encodings such as compressed data.
- IT hides the unneccesary and redundant information from the application.
- It has minimum support for dependent field.
- It removes the idea of semantic actions, which reduces the expressive power of the grammar.

Code Snippet:

struct anypacket *parse_anypacket(NailArena *arena, const uint8_t *data, size_t size); //Its a parser function to parse a protocol grammar.

int gen_anypacket(NailArena *tmp_arena, NailStream *out, struct anypacket *val); //Its a generator function.

Comparison of Protocol Parsing Languages

Nos	NAME	P4	binPAC	SPICY/BINPAC++	NAIL	GAPAL	PJKundert/CPPPO	UltraPAC	CLICK	PacketC	Frentic
1	URL	p4.org	bro.org/sphinx/binp	icir.org/hilti	github.com/jbangert/nail		pypi.python.org/pypi/cpppo				frentic-lang.org/
2	Release date	2013	2006	2014	2014	2007	2015	2010	2000	2009	2010
3	Version	1.1.0	binPAC 0.44				CPPPO 3.9.4				https://github.com/frenetic-lang/frenetic
4	relevant programmng language	C	c++	c++	c	high level language evaluated by c++ interpreter	Python	embedded c++	c++	C99 variant of C Language	Python
5	based network detection tool	wireshark	Bro(Not tied to Bro)	Hilti/Spicy		not required	wireshark				
6	Network emulation Platform	mininet	mininet	mininet	mininet	mininet	mininet	mininet	mininet		
7	Main Goals	1.Reconfirgurability 2. Protocol Independence 3. Target Independence	generate c++ parsers intended to process traffic of much higher volume at network gateways.	extended version of binPAC, it unifies syntax and semantics in a single language.	can handle complicated encodings such as compressed data	self contained system,handles both protcol parsing and traffic analysis	reproducibility, realisitc simulation of industrial protocls,	replace binpac's tree parser with stream parser	Very Expressive and suitable to express how the packets are proessed.	deep packet inspection not nly header but also the contents of the packet	Designed to solve major Openflow/NOX problems.
8	Controller/Switch Architecture	Supports	supports	supports			supports	supports	no support		supports
9	Main issues	implementation relies on specialized hardware.	requires considerable effort on specifying new protocol	not much support, still updating	Nail had limited support for dependent field	more overhead, runs relatively on low speed		requires considerable effort on specifying new protocols, not much work has been published	difficult to infer dependencies that constrain paralle execution		Flexibility is limited bt the current parser, needs to update the language.

Fraunhofer SIT

Conclusion

- In my research work, I have studied around 15 – 18 protocol parsing languages, I have mentioned some of them in the previous slides others are: FlowLog, Merlin, NetCore, NICE, COPY, PADS
- There can be more languages for protocol parsing, but I have found that these are the most relevant for our work and most of them have been tested on the network traffic.
- On the basis of my research, I have selected P4 language for my thesis work because of the following points:

1. It works with the conjuction of control plane and data plane protocols.
2. High level domain specific language, Can parse high volume network traffic.
3. Parse packet headers easily and efficiently.
4. Protocol independent.
5. Target Independent.
6. The support community of P4 language is very vast.
7. Match+action for the header parsing.
8. Supports the software defined networking.

Thank You…

References

- Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." *ACM SIGCOMM Computer Communication Review* 44.3 (2014): 87-95.
- Pang, Ruoming, et al. "binpac: A yacc for writing application protocol parsers." *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM, 2006.It analyzes both network protocols and file formats
- Zhang, Kai, et al. "Building high-performance application protocol parsers on multi-core architectures." *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*. IEEE, 2011.They both are still in prototype state means not production ready.
- Sommer, Robin, Johanna Amann, and Seth Hall. "Spicy: A Unified Deep Packet Inspection Framework Dissecting All Your Data." (2015).
- Li, Zhichun, et al. "Netshield: massive semantics-based vulnerability signature matching for high-speed networks." *ACM SIGCOMM Computer Communication Review*. Vol. 40. No. 4. ACM, 2010.
- Li, Hao, et al. "Parsing application layer protocol with commodity hardware for SDN." *Architectures for Networking and Communications Systems (ANCS), 2015 ACM/IEEE Symposium on*. IEEE, 2015.
- Borisov, Nikita, et al. "Generic Application-Level Protocol Analyzer and its Language." *NDSS*. 2007.