KASPERSKY lab

SECUR'IT CUP'18

NETCAP: A framework for secure and concurrent network traffic analysis

$ whoami

Student from the LMU Munich

Interests:

- Network Security Monitoring & Anomaly Detection
- Machine Learning
- Programming (Golang, C / C++ / ObjC, Swift, Haskell, Python, Rust)
- Hardware & Software Security
- Reverse Engineering
- Penetration Testing

Why network security monitoring?

Network environments have many vulnerable components

Hackers will always find a way in - “low hanging fruit”

Be prepared for the worst case!

Why anomaly detection?

Number of signatures is exploding (Kaspersky: 330 000 new samples per day)

New threats cannot be detected by signatures

Existing malware can be obfuscated to be fully undetectable again

Problems?

"The monitor will be attacked."

–Vern Paxson, 1998

A major problem for data collection

memory safety

All existing frameworks are written in C or C++

Parsing network protocols is complex.

Mistakes happen a lot.

MITRE CVE results for Snort IDS

CVE-2016-1463	Cisco FireSIGHT System Software 5.3.0, 5.3.1, 5.4.0, 6.0, and 6.0.1 allows remote attackers to bypass Snort rules via crafted parameters in the header of an HTTP packet, aka Bug ID CSCuz20737.
CVE-2016-1417	Untrusted search path vulnerability in Snort 2.9.7.0-WIN32 allows remote attackers to execute arbitrary code and conduct DLL hijacking attacks via a Trojan horse tcapi.dll that is located in the same folder on a remote file share as a pcap file that is being processed.
CVE-2015-6427	Cisco FireSIGHT Management Center allows remote attackers to bypass the HTTP attack detection feature and avoid triggering Snort IDS rules via an SSL session that is mishandled after decryption, aka Bug ID CSCux53437.
CVE-2014-4695	Multiple open redirect vulnerabilities in the Snort package before 3.0.13 for pfSense through 2.1.4 allow remote attackers to redirect users to arbitrary web sites and conduct phishing attacks via (1) the referer parameter to snort_rules_flowbits.php or (2) the returl parameter to snort_select_alias.php.
CVE-2014-4693	Multiple cross-site scripting (XSS) vulnerabilities in the Snort package before 3.0.13 for pfSense through 2.1.4 allow remote attackers to inject arbitrary web script or HTML via (1) the eng parameter to snort_import_aliases.php or (2) unspecified variables to snort_select_alias.php.
CVE-2009-4211	The U.S. Defense Information Systems Agency (DISA) Security Readiness Review (SRR) script for the Solaris x86 platform executes files in arbitrary directories as root for filenames equal to (1) java, (2) openssl, (3) php, (4) snort, (5) tshark, (6) vncserver, or (7) wireshark, which allows local users to gain privileges via a Trojan horse program.
CVE-2009-3641	Snort before 2.8.5.1, when the -v option is enabled, allows remote attackers to cause a denial of service (application crash) via a crafted IPv6 packet that uses the (1) TCP or (2) ICMP protocol.
CVE-2007-0251	Integer underflow in the DecodeGRE function in src/decode.c in Snort 2.6.1.2 allows remote attackers to trigger dereferencing of certain memory locations via crafted GRE packets, which may cause corruption of log files or writing of sensitive information into log files.
CVE-2006-6931	Algorithmic complexity vulnerability in Snort before 2.6.1, during predicate evaluation in rule matching for certain rules, allows remote attackers to cause a denial of service (CPU consumption and detection outage) via crafted network traffic, aka a "backtracking attack."
CVE-2006-5276	Stack-based buffer overflow in the DCE/RPC preprocessor in Snort before 2.6.1.3, and 2.7 before beta 2; and Sourcefire Intrusion Sensor; allows remote attackers to execute arbitrary code via crafted SMB traffic.
CVE-2006-2769	The HTTP Inspect preprocessor (http_inspect) in Snort 2.4.0 through 2.4.4 allows remote attackers to bypass "uricontent" rules via a carriage return (\r) after the URL and before the HTTP declaration.

MITRE CVE results for Bro IDS

Search Results

There are **6** CVE entries that match your search.

Name	Description
CVE-2018-17019	In Bro through 2.5.5, there is a DoS in IRC protocol names command parsing in analyzer/protocol/irc/IRC.cc.
CVE-2018-16807	In Bro through 2.5.5, there is a memory leak potentially leading to DoS in scripts/base/protocols/krb/main.bro in the Kerberos protocol parser.
CVE-2017-1000458	Bro before Bro v2.5.2 is vulnerable to an out of bounds write in the ContentLine analyzer allowing remote attackers to cause a denial of service (crash) and possibly other exploitation.
CVE-2015-1522	analyzer/protocol/dnp3/DNP3.cc in Bro before 2.3.2 does not reject certain non-zero values of a packet length, which allows remote attackers to cause a denial of service (buffer overflow or buffer over-read) via a crafted DNP3 packet.
CVE-2015-1521	analyzer/protocol/dnp3/DNP3.cc in Bro before 2.3.2 does not properly handle zero values of a packet length, which allows remote attackers to cause a denial of service (buffer overflow or buffer over-read if NDEBUG; otherwise assertion failure) via a crafted DNP3 packet.
CVE-2007-0186	Multiple cross-site scripting (XSS) vulnerabilities in F5 FirePass SSL VPN allow remote attackers to inject arbitrary web script or HTML via (1) the xcho parameter to my.logon.php3; the (2) topblue, (3) midblue, (4) wtopblue, and certain other Custom color parameters in a per action to vdesk/admincon/index.php; the (5) h321, (6) h311, (7) h312, and certain other Front Door custom text color parameters in a per action to vdesk/admincon/index.php; the (8) ua parameter in a bro action to vdesk/admincon/index.php; the (9) app_param and (10) app_name parameters to webyfiers.php; (11) double eval functions; (12) JavaScript contained in an <FP_DO_NOT_TOUCH> element; and (13) the vhost parameter to my.activation.php. NOTE: it is possible that this candidate overlaps CVE-2006-3550.

More Bro

Bro 2.5.5

Top

Bro 2.5.5 primarily addresses security issues.

- Fix array bounds checking in BinPAC: for arrays that are fields within a record, the bounds check was based on a pointer to the start of the record rather than the start of the array field, potentially resulting in a buffer over-read.
- Fix SMTP command string comparisons: the number of bytes compared was based on the user-supplied string length and can lead to incorrect matches. e.g. giving a command of "X" incorrectly matched "X-ANONYMOUSTLS" (and an empty commands match anything).

The following changes address potential vectors for Denial of Service reported by Christian Titze & Jan Grashöfer of Karlsruhe Institute of Technology:

- "Weird" events are now generally suppressed/sampled by default according to some tunable parameters:
  - Weird::sampling_whitelist
  - Weird::sampling_threshold
  - Weird::sampling_rate
  - Weird::sampling_duration

  Those options can be changed if one needs the previous behavior of a "net_weird", "flow_weird", or "conn_weird" event being raised for every single event. Otherwise, there is a new weird_stats.log which contains concise summaries of weird counts per type per time period and the original weird.log may not differ much either, except in the cases where a particular weird type exceeds the sampling threshold. These changes help improve performance issues resulting from excessive numbers of weird events.

- Improved handling of empty lines in several text protocol analyzers that can cause performance issues when seen in long sequences.

- Add ‘smtp_excessive_pending_cmds’ weird which serves as a notification for when the "pending

!Problem?

To the rescue!

NETCAP

Traffic Analysis Framework

How does it work?

Uses gopacket library for decoding packets in pure Go

Generates audit records as compressed protocol buffers

Concurrent design: worker pool, each audit record written to a separate file

-> enables consuming applications to implement a concurrent design as well

Why Protocol Buffers?

Type safe structured data - can represent complex nested structures

Goal: Data Accessibility. Generated Data is available in almost any programming language

Huge landscape of frameworks for machine learning, in many interesting languages (R, Scala, Haskell…)

see: https://github.com/josephmisiti/awesome-machine-learning

What audit records are generated?

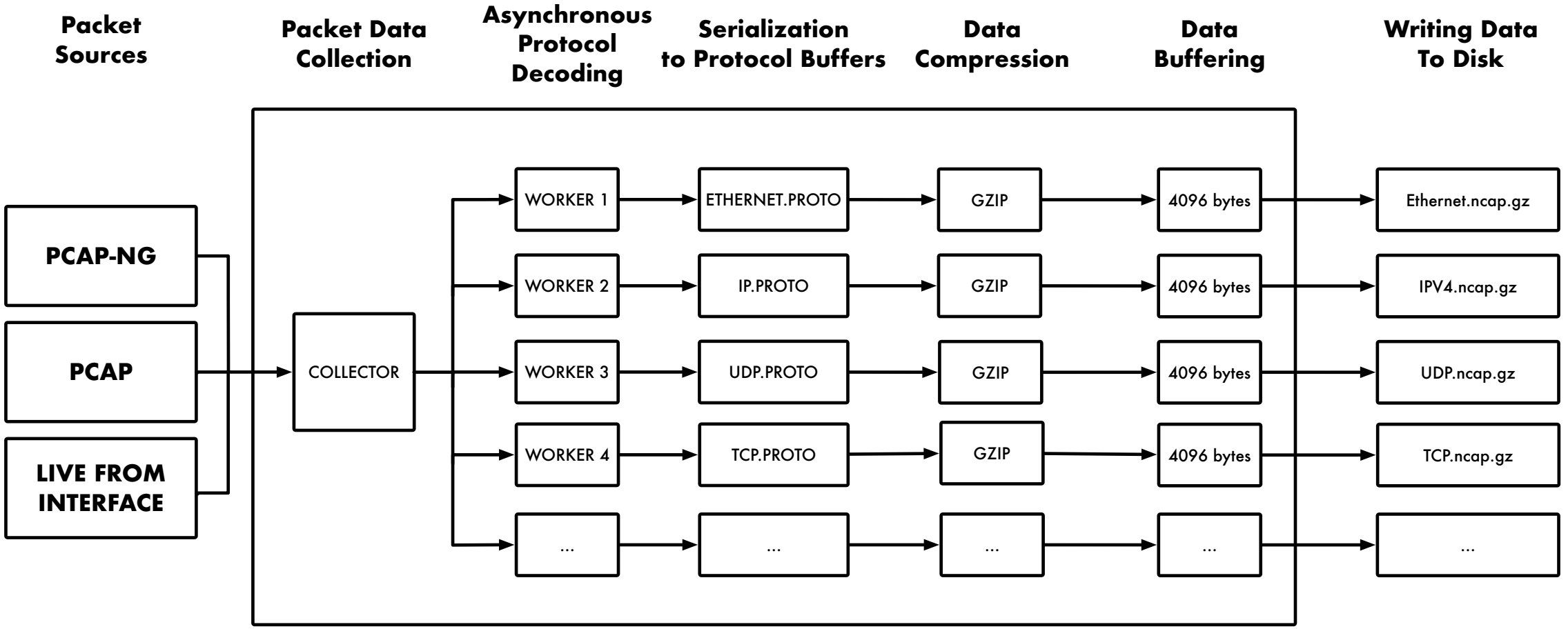
+ TLS (Client Hello Msg + Ja3)
+ LinkFlow
+ NetworkFlow
+ TransportFlow
+ HTTP
+ Flow (unidirectional)
+ Connection (bidirectional)

+ NTP
+ SIP
+ IGMP
+ LLC
+ IPv6HopByHop
+ SCTP
+ SNAP
+ LinkLayerDiscovery
+ ICMPv6NeighborAdvertisement
+ ICMPv6RouterAdvertisement
+ EthernetCTP
+ EthernetCTPReply
+ LinkLayerDiscoveryInfo

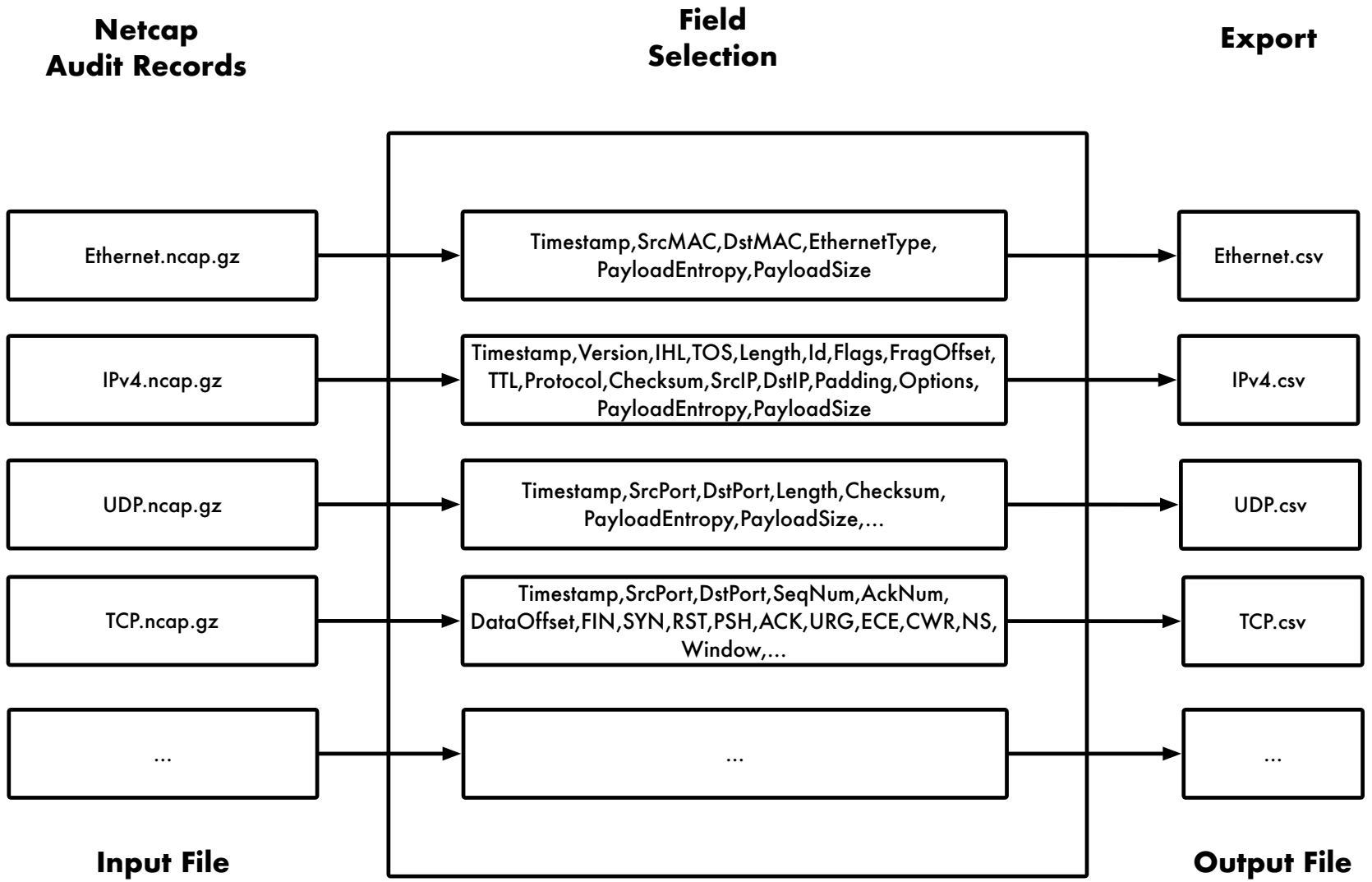
+ TCP
+ UDP
+ IPv4
+ IPv6
+ DHCPv4
+ DHCPv6
+ ICMPv4
+ ICMPv6
+ ICMPv6Echo
+ ICMPv6NeighborSolicitation
+ ICMPv6RouterSolicitation
+ DNS
+ ARP
+ Ethernet
+ Dot1Q
+ Dot11

Tell me more.

NETCAP in a nutshell



NETCAP filtering & csv export



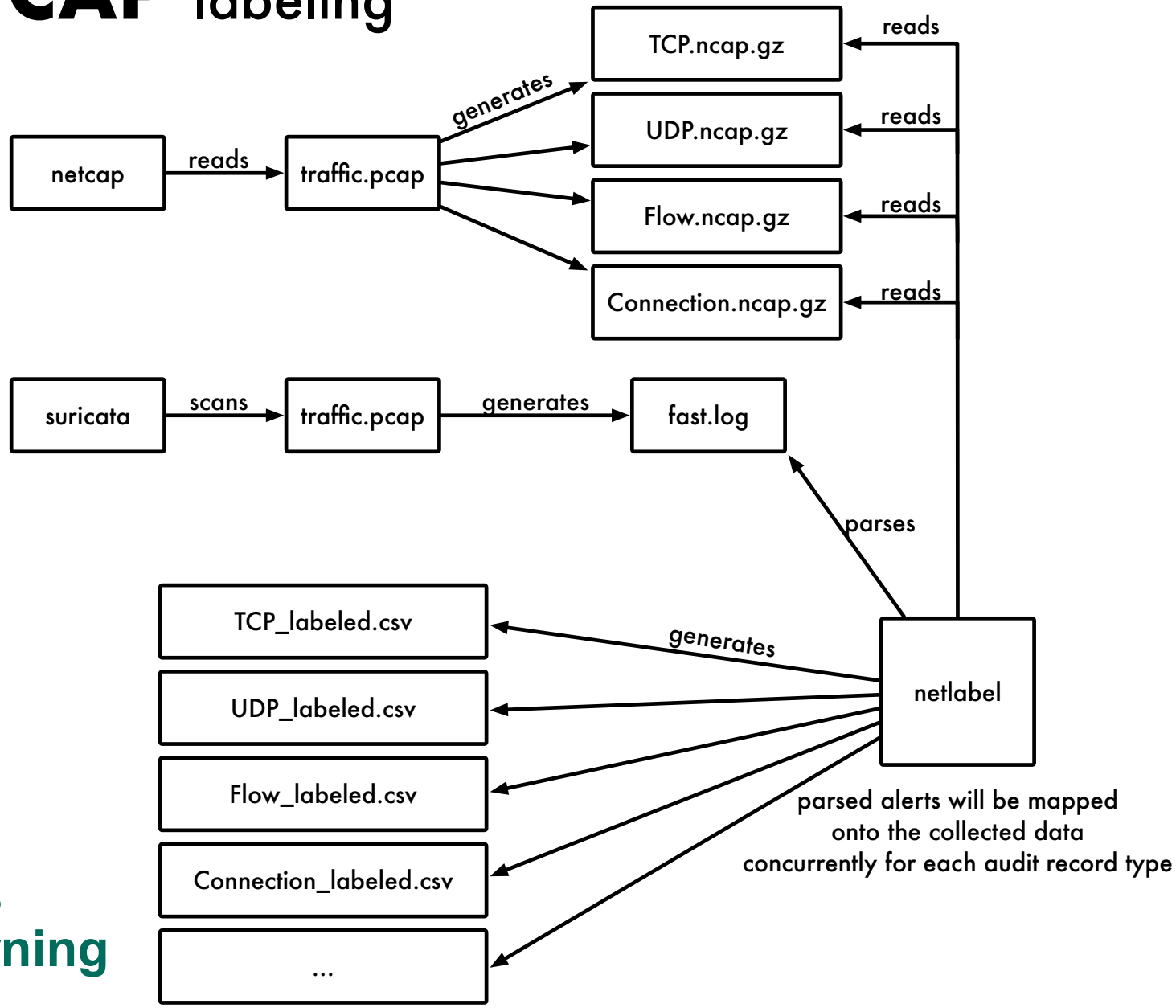

```
$ netcap -r TCP.ncap.gz    -select Timestamp,SrcPort,DstPort,SeqNum,Window,ACK,SYN,RST   > TCP.csv
```

NETCAP labeling

Phase 1:
Data generation with netcap

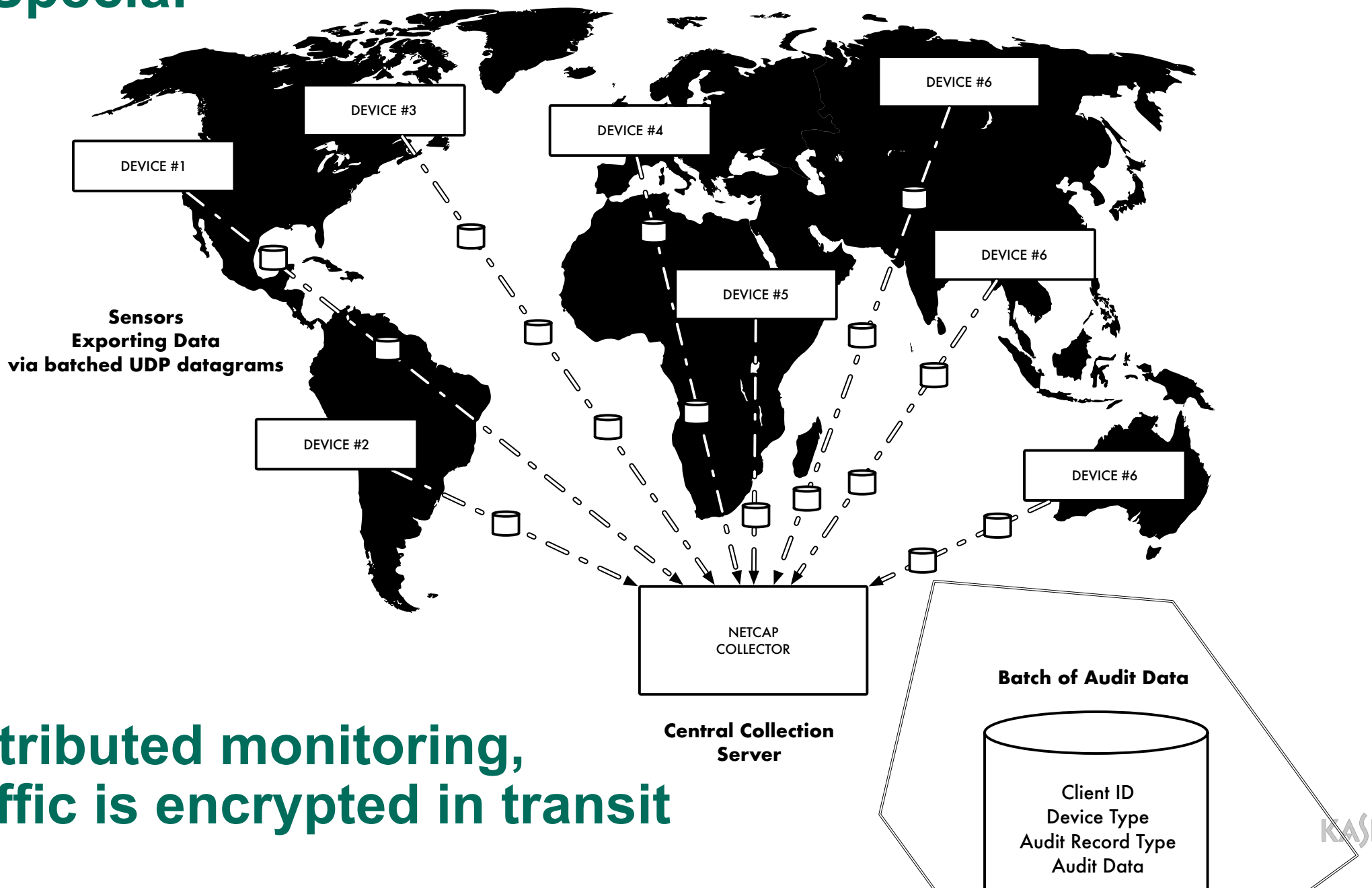
Phase 2:
Label extraction with suricata

Phase 3:
Mapping alerts with netcap



Creation of labeled datasets for supervised machine learning

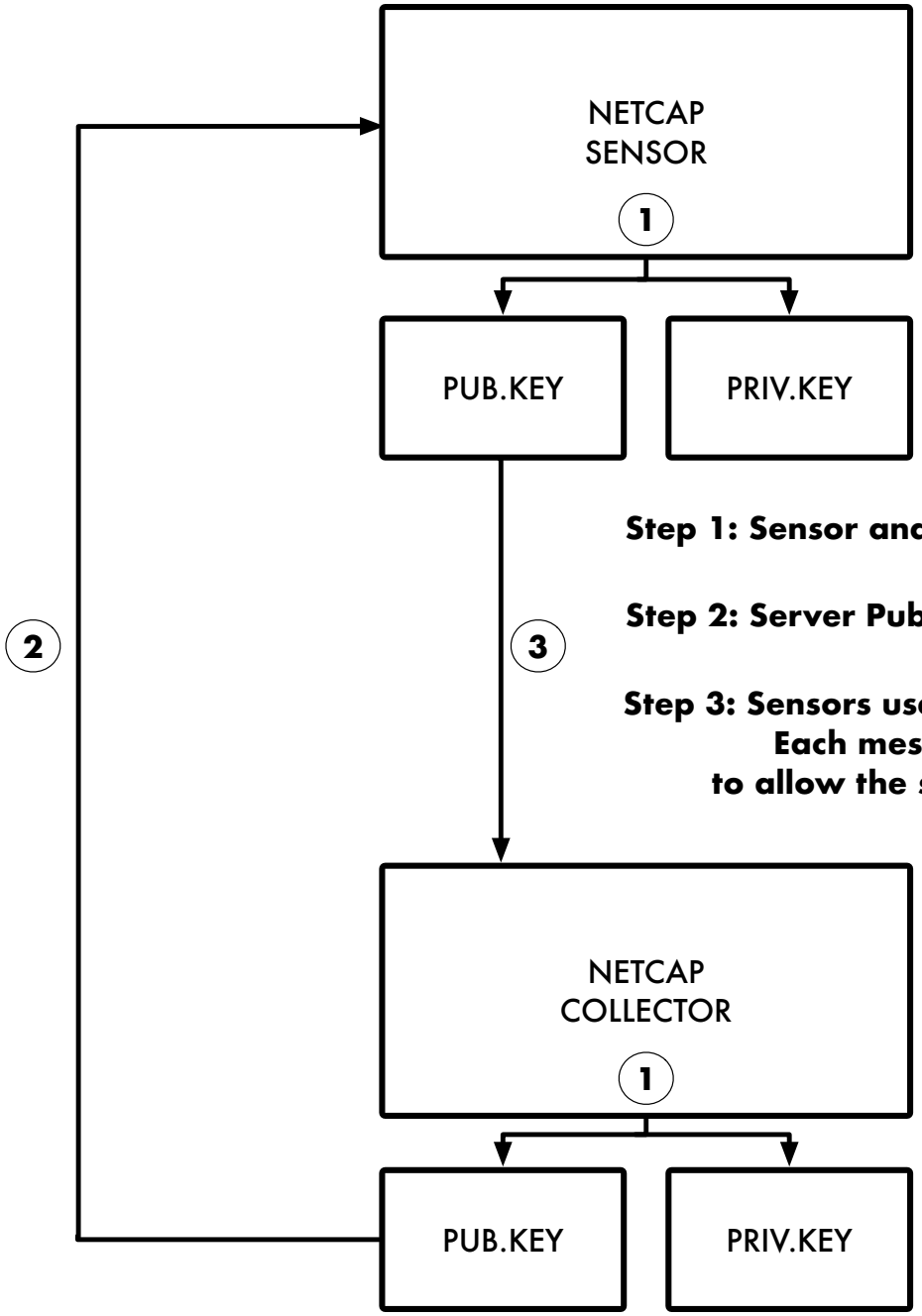
Cup Special



Distributed monitoring, Traffic is encrypted in transit

Cup Special

Data encryption



Step 1: Sensor and Collector server generate keypair

Step 2: Server Public key is distributed to sensors

**Step 3: Sensors use the servers public key to send encrypted messages.
Each message has the sensors public key prepended,
to allow the server to authenticate and decrypt the message**

Why monitor IoT devices?

A majority is compromised by brute force attacks - easy to detect with monitoring

Embedded applications are written in low level languages (C / C++) and may contain exploitable bugs

Many devices are infrequently updated and are vulnerable for a long period of time

State of IoT 2018?

From home and business security, industrial equipment and healthcare, to air travel, insurance and entertainment, the Internet of Things has spread rapidly across business sectors. Node.js is especially useful in IoT as this field continues to grow in popularity. Anytime a user has endpoints running Node.js in the field — whether it be in chipset or a kiosk/device — they need to ensure their apps are running, and doing so effectively. Collecting all that endpoint data effectively, and being able to repurpose it as well - is critical for the success of IoT implementations.

Traditionally, it's been hard to debug hardware. Writing software that enables hardware to "phone home" to web-based services can be especially tricky. Node.js is being adopted in IoT use cases for rapidly prototyping before products get hardened and mass produced. However, troubleshooting open source Node.js by itself in a debugging environment or as part of quality assurance can be difficult, if not impossible.

Wait… Node.js on IoT devices?

Sounds like a bad idea.

Casper Beyer Follow
A slightly bitter bearded developer with a background in arts.
Mar 30 · 4 min read

The Node.js Ecosystem Is Chaotic and Insecure

Home > Security

NEWS ANALYSIS

Malicious code in the Node.js npm registry shakes open source trust model

Bad actors using typo-squatting place 39 malicious packages in npm that went undetected for two weeks. How should the open source community respond?

By Fahmida Y. Rashid
Contributor, CSO | AUG 8, 2017 4:03 AM PT

Lucky for me, we live in an age where people install npm packages like they're popping pain killers.

Software

How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

By Chris Williams, Editor in Chief 23 Mar 2016 at 01:24 167 SHARE ▼

David Gilbertson Follow
I like web stuff and cryptocurrency stuff.
Jan 6 · 10 min read

I'm harvesting credit card numbers and passwords from your site. Here's how.

Abdulrasaq Nasirudeen FOLLOW
Full stack software developer, Security researcher and an excellent communicator. I am highly solution o...

Few ways I could hijack your node.js applications

Published Feb 26, 2018 Last updated Aug 25, 2018

OpSecX

MORE BY OPSECX

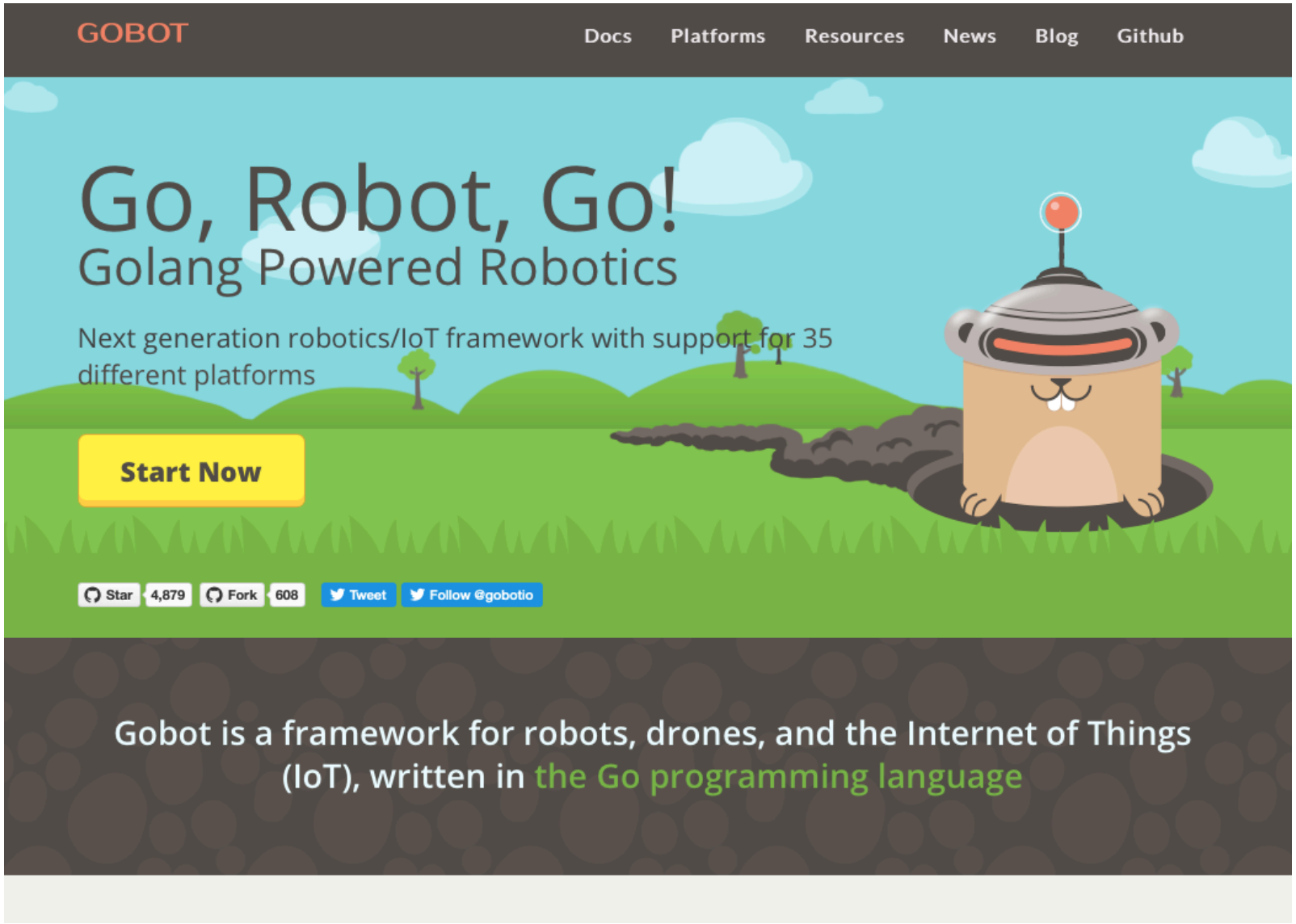
Exploiting Node.js deserialization bug for Remote Code Execution

FEBRUARY 8, 2017 BLOG

tl;dr

Untrusted data passed into `unserialize()` function in node-serialize module can be exploited to achieve arbitrary code execution by passing a serialized JavaScript Object with an Immediately invoked function expression (IIFE).

The good news: Go is gaining popularity for IoT applications



What about embedded systems?

Go depends on features of the operating system.
However, there is work in progress on bringing it to embedded devices as well.
Also the go authors are aware of this - maybe Go 2 will change the game?

README.md

TinyGo - Go compiler for microcontrollers

docs passing build passing

> We never expected Go to be an embedded language and so it's got serious problems [...].

-- Rob Pike, GopherCon 2014 Opening Keynote

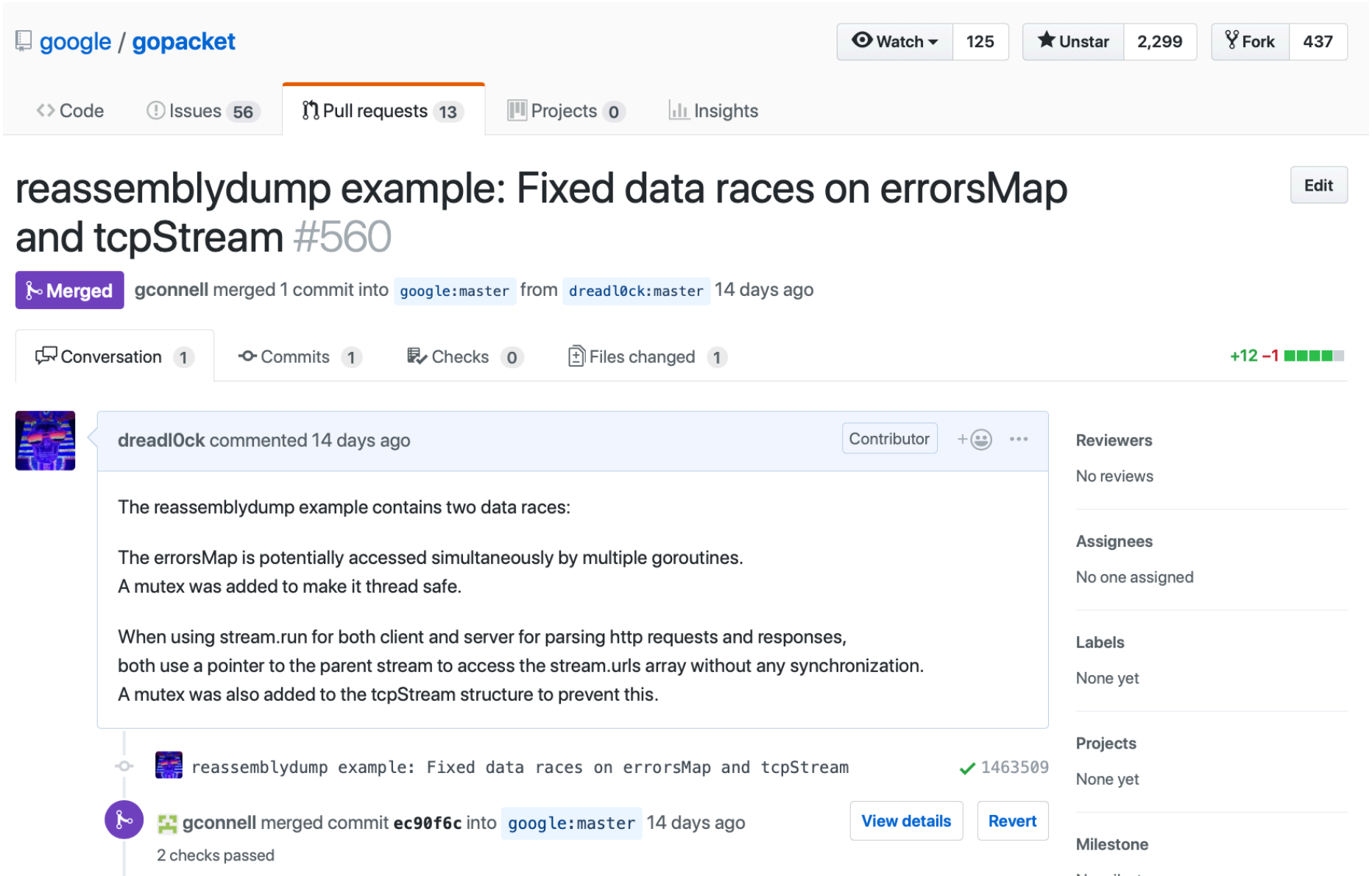
TinyGo is a project to bring Go to microcontrollers and small systems with a single processor core. It is similar to emgo but a major difference is that I want to keep the Go memory model (which implies garbage collection of some sort). Another difference is that TinyGo uses LLVM internally instead of emitting C, which hopefully leads to smaller and more efficient code and certainly leads to more flexibility.

My original reasoning was: if Python can run on microcontrollers, then certainly Go should be able to and run on even lower level micros.

Whats your business model?

GPLv3 :)

Further open source contributions



google / gopacket

Watch 125 | Unstar 2,299 | Fork 437

Code | Issues 56 | Pull requests 13 | Projects 0 | Insights

reassemblydump example: Fixed data races on errorsMap and tcpStream #560

Edit

Merged gconnell merged 1 commit into google:master from dreadl0ck:master 14 days ago

Conversation 1 | Commits 1 | Checks 0 | Files changed 1 | +12 −1

dreadl0ck commented 14 days ago — Contributor

The reassemblydump example contains two data races:

The errorsMap is potentially accessed simultaneously by multiple goroutines.
A mutex was added to make it thread safe.

When using stream.run for both client and server for parsing http requests and responses,
both use a pointer to the parent stream to access the stream.urls array without any synchronization.
A mutex was also added to the tcpStream structure to prevent this.

reassemblydump example: Fixed data races on errorsMap and tcpStream ✓ 1463509

gconnell merged commit ec90f6c into google:master 14 days ago — View details | Revert

2 checks passed

Reviewers: No reviews

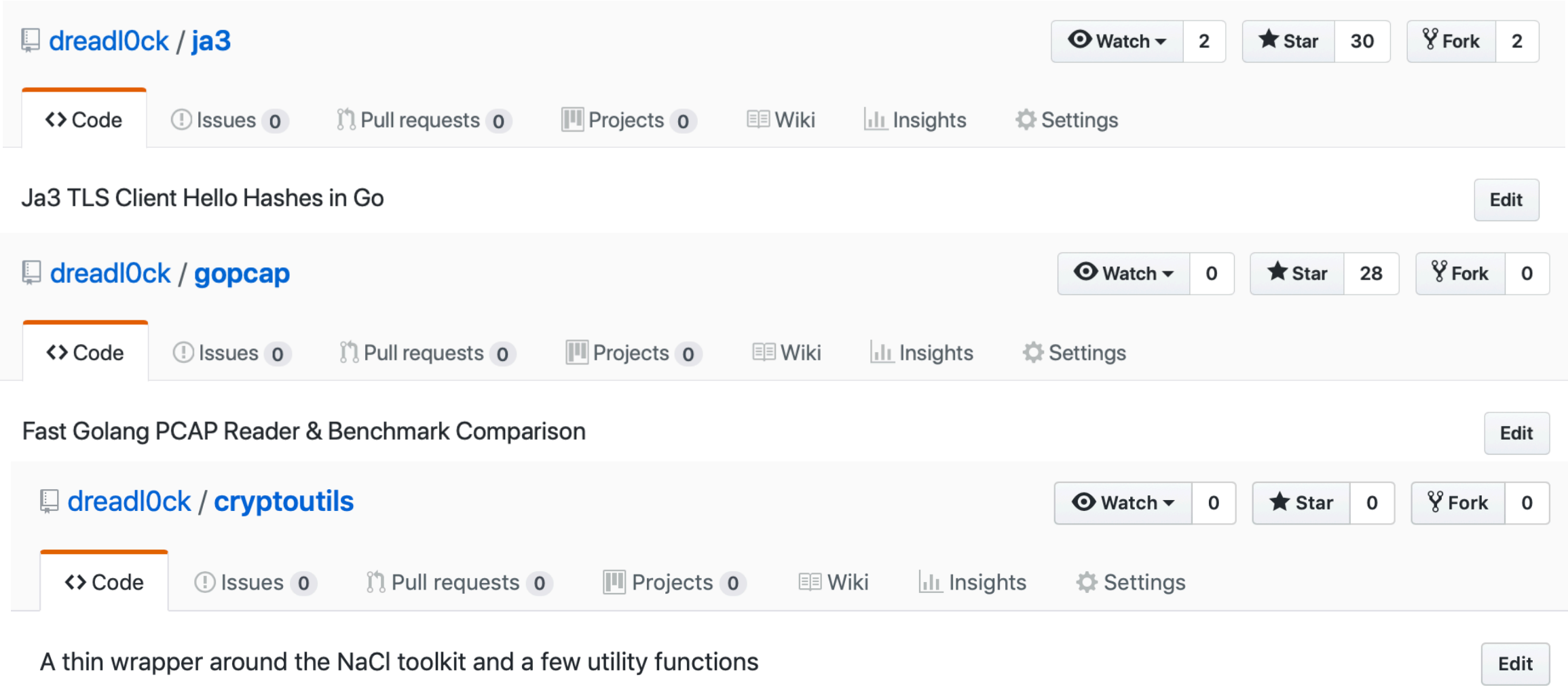
Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone

Further open source contributions



Other use cases?

Monitor honeypots

Monitor medical devices

Forensic Analysis

Research! :)

Core benefits of netcap?

Memory Safety (pure Golang codebase)

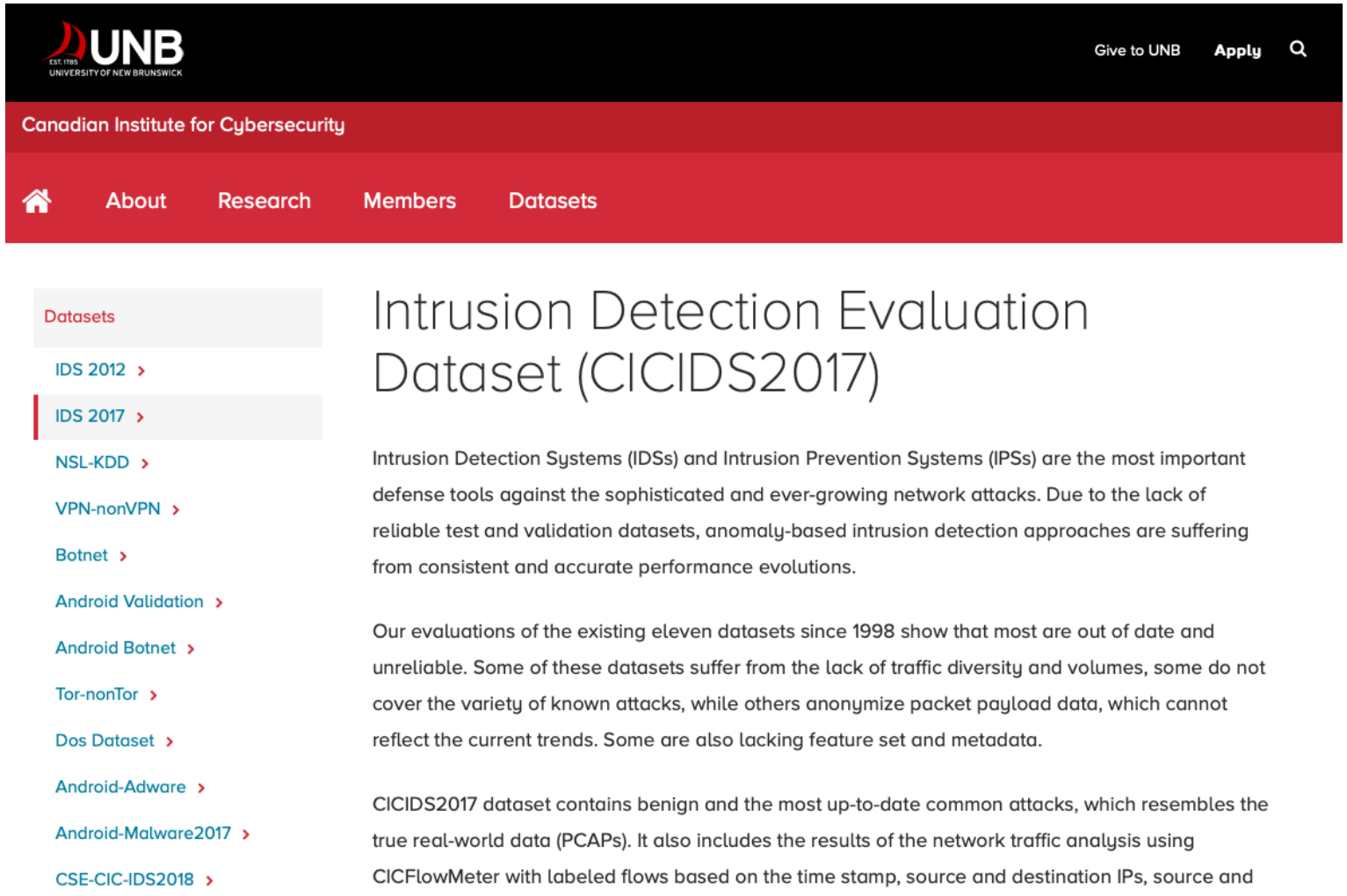
Data Accessibility (Protocol Buffers)

Performance (all 40+ encoders, ~1GB/min processing time)

Documentation (2k lines of comments on 8k lines of Go code)

130+ pages thesis with in depth technical documentation and usage

PoC || GTFO



Up-to-date, well documented dataset
~50GB pcap dumps

PoC || GTFO

Thesis excerpt: Evaluation of extracted and labeled data with netcap using Tensorflow and a Deep Neural Network

Tuesday-WorkingHours.pcap

File	Num Records	Size	Labels	Exec Time	Validation Score
Connection_labeled.csv	284166	51M	1809	1m 14s	0.993623490329664 1
DNS_labeled.csv	700447	144M	37	2m 3s	0.99996573621453 7
Ethernet_labeled.csv	11551954	880M	3551	32m 24s	0.99969909857689 9
Flow_labeled.csv	647294	112M	3340	1m 57s	0.99468558433854 06
HTTP_labeled.csv	45800	14M	4214	39s	0.93633187772925 76
NTP_labeled.csv	15507	3.0M	13	5s	0.99871034304874 91
NetworkFlow_labeled.csv	29094	4.0M	1292	9s	0.95724498212812 76
TCP_labeled.csv	10710230	1.5G	3450	30m 59s	0.99967208926940 14
TransportFlow_labeled.csv	212613	20M	1721	43s	0.99153403186590
UDP_labeled.csv	787015	43M	46	2m 9s	0.99993901013448 26

Table 6.35: Classification results experiment 6 Tuesday-WorkingHours.pcap

Thesis? Coming soon!

INSTITUT FÜR INFORMATIK

DER LUDWIG–MAXIMILIANS–UNIVERSITÄT MÜNCHEN

Bachelorarbeit

Concept and implementation of memory safe feature collection for network intrusion detection

Conclusion

Use memory safe programming languages for the development of critical software infrastructure

This allows us to focus on the creation of solid logic, rather than solid memory management.

Golang || Rust


SECUR'IT
CUP'18
Thank you!
Questions?
philipp.mieden@protonmail.ch
github.com/dreadl0ck
twitter.com/dreadcode
KASPERSKY lab