ARMY RESEARCH LABORATORY

# Extremely Lightweight Intrusion Detection (ELIDe)

**by Raymond J. Chang, Richard E. Harang, and Garrett S. Payer**

**ARL-CR-0730** **December 2013**

Prepared by
ICF International
7125 Thomas Edison Drive Suite 100
Columbia, MD 21046

Under contract
W911QX-12-F-0052
US ARMY RESEARCH LABORATORY
RDRL-CIN-D (RICHARD E. HARANG)
2800 Powder Mill Rd.
Adelphi, MD 20783-119

# ERRATA SHEET

**re: ARL-TR-6776, *Extremely Lightweight Intrusion Detection (ELIDe)*, December 2013, by Raymond J. Chang, Richard E. Harang, and Garett S. Payer**

This is an errata sheet for ARL-TR-6776. Note: The report number has changed to ARL-CR-0730. Please attach this sheet to the cover page of the original document.

| Page | Reads | Should Read |
|---|---|---|
| Cover, Title page, and 298 | ARL-TR-6776 | ARL-CR-0730 |
| Cover, Title page | | ICF International, 7125 Thomas Edison Drive<br>Suite 100, Columbia, MD 21046<br>Under contract: W911QX-12-F-0052 |

[illegible]

Richard E. Harang

U.S. Army Research Laboratory
Computational and Information Sciences Directorate
ATTN: RDRL-CIN-D
Adelphi, MD 20783-1197

**Army Research Laboratory**
Adelphi, MD 20783-1197

**ARL-CR-0730** **December 2013**

# Extremely Lightweight Intrusion Detection (ELIDe)

**Raymond J. Chang, Richard E. Harang, and Garrett S. Payer**
**Computational and Information Sciences Directorate, ARL**

Prepared by
ICF International
7125 Thomas Edison Drive Suite 100
Columbia, MD 21046

Under contract
W911QX-12-F-0052
US ARMY RESEARCH LABORATORY
RDRL-CIN-D (RICHARD E. HARANG)
2800 Powder Mill Rd.
Adelphi, MD 20783-119

| REPORT DOCUMENTATION PAGE | | | *Form Approved* *OMB No. 0704-0188* |
|---|---|---|---|

| | | |
|---|---|---|
| 1. **REPORT DATE** *(DD-MM-YYYY)* December 2013 | **2. REPORT TYPE** | **3. DATES COVERED (From - To)** March to July 2013 |
| **4. TITLE AND SUBTITLE** Extremely Lightweight Intrusion Detection (ELIDe) | | **5a. CONTRACT NUMBER** W911QX-12-F-0052 |
| | | **5b. GRANT NUMBER** |
| | | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Raymond J. Chang, Richard E. Harang, and Garrett S. Payer | | **5d. PROJECT NUMBER** |
| | | **5e. TASK NUMBER** |
| | | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** U.S. Army Research Laboratory ATTN: RDRL-CIN-D 2800 Powder Mill Road Adelphi, MD 20783-197 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-CR-0730 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** | | **10. SPONSOR/MONITOR'S ACRONYM(S)** |
| | | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release; distribution unlimited.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**
The need to provide network protection and monitoring extends beyond defending conventional wired computing infrastructures to mobile ad-hoc networks. This need motivates the research and development of network defense methodologies and technologies that are applicable in a tactical environment in which resources are constrained and topologies are dynamic. The project documented by this technical report makes the contribution of prototyping a packet analysis tool named Extremely Lightweight Intrusion Detection (ELIDe) with the capability to approximate Snort-like signature matching against the inbound and outbound network traffic of a single host, while requiring less than 2% of the peak memory footprint demanded by Snort. This economy of resources makes ELIDe suitable for operation in a constrained environment, such as a tactical network that cannot support a more conventional solution like Snort.

**15. SUBJECT TERMS**
Network intrusion detection, hash kernel, lightweight, mobile, ad-hoc, packet analysis

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Raymond J. Chang |
|---|---|---|---|---|---|
| **a. REPORT** Unclassified | **b. ABSTRACT** Unclassified | **c. THIS PAGE** Unclassified | UU | 24 | **19b. TELEPHONE NUMBER** (*Include area code*) (301) 394-1835 |

**Standard Form 298 (Rev. 8/98)**
**Prescribed by ANSI Std. Z39.18**

## Contents

## List of Figures

## 1. Background

Snort is a widely deployed and powerful network signature-based intrusion detection technology (*1*). It boasts both a massive rule database generated and maintained by the Sourcefire Vulnerability Research Team (more than 26,000 alert definitions as of the 3 Sep 2013 snapshot [*2*]) and flexible fuzzy matching capabilities based upon Perl-compatible regular expressions. The tradeoff for this robust functionality is that Snort has significant memory demands at runtime. When benchmarked on a conventional computing platform (Dell Inspiron 15N laptop running Mint Maya as the operating system, dual-core Core i5 CPU, 8 GB RAM), Snort exhibited a peak RAM usage of approximately 1.2 GB as measured by the Massif memory profiler within the Valgrind suite (*3*). While this is very reasonable for commodity computing hardware (dedicated to running Snort) that might be found in a network operations center, it becomes a prohibitive requirement for devices that would be more commonly found in a dynamic tactical environment. As a point of reference, the Raspberry Pi single-chip computer (*4*) is equipped with only 512 MB of RAM and would, therefore, be overwhelmed by the runtime demands of Snort. Consequently, any packet analysis solution in the tactical requirement must make do with significantly fewer resources than what is needed by Snort.

In addition to the resource requirement, there are other significant disparities between the environments and use cases in which Snort is typically deployed, and a mobile tactical network. Conventional computing environments that rely upon Snort as at least part of their intrusion detection solution are typically quite static in terms of topology and composition, likely with multiple gigabits per second of bandwidth available. This known and static layout permits a small number of sensors (perhaps just one or two) equipped with Snort to reliably monitor the entire network, while the available bandwidth is able to support periodic updates to the signature databases used by each instance of Snort. In contrast, the mobile tactical network will have a highly dynamic topology with substantially less available bandwidth. In order to guarantee coverage of the entire network, the dynamic topology compels the presence of a sensor local to every single network node. Since ad-hoc mobile networks are comprised of devices such as smartphones, tablets, single-chip computers, and embedded hardware platforms, these nodes are likely to be undersized in terms of processing capacity in comparison to their data center counterparts. Updates to these sensors must be also extremely lightweight and support asynchronous delivery to accommodate the constrained and unpredictable connectivity in this environment. Finally, the possibility that the risk of the adversary physically appropriating devices from the tactical network (likely to be small and portable) implies that they should contain as little sensitive information as possible. If a standard Snort rule set file were captured in such a circumstance, the adversary would be able to exactly enumerate the network signature and alerting capability of the tactical security system.

This adds up to the need for a network packet analysis solution that can approximate the signature matching capabilities of Snort, drastically mitigate resource consumption and operating load, use extremely lightweight constructs for updates, and locally obfuscate the signature baseline. We propose the Extremely Lightweight Intrusion Detection (ELIDe) solution to meet these needs.

## 2. Related Work

Previous profiling efforts [Spyros04] have identified string comparisons as the primary bottleneck for Snort throughput, reporting that such operations consume up to approximately 70% of the total execution time and 80% of executed instructions on realistic network traffic. Several approaches have been proposed to mitigate this bottleneck, either through moving expensive or paralellizable computations to specialized hardware (*20, 21, 23, 26*), software mitigations (*22*) , or a combination of the two (*19*).

The work of (*23*) proposes an n-byte "jumping window" pattern matching scheme to pre-process lookups using ternary content addressable memory (TCAM). While this uses a similar insight to ELIDe—that n-grams of the packet form a useful proxy for sequential dependencies—they focus on multi-gigabit rates of deep packet inspection, and require specialized hardware (the TCAM itself). Their use of a jumping window (rather than a sliding window such as in our application) does significantly reduce the computational cost, but with a corresponding decrease in accuracy due to "frame shift" errors. They also focus on use of TCAM to do direct matching from raw payload to signatures, which does not mitigate the OpSec concerns that formed part of the design goal of this project. The work of (*19*) also uses TCAM in order to generalize and accelerate certain transitions in finite state machine (FSM) operations in signature matching. As with the work of (*23*), however, it focuses on direct translation of signatures of concern into FSM structures. The pattern-matching flexibility of TCAM is also exploited in (*26*) who attempts to reduce the often significant power requirements of TCAM by use of a novel set-splitting algorithm; their splitting allows for paralellism in matching, as well as a reduction in update cost; it, however, still requires the use of additional hardware, and as they report, only partially mitigates the additional cost of TCAM over standard memory.

In (*20*), graphics processing units (GPUs) are examined to paralellize string matching operations. As the signature matching process is, itself, embarassingly paralellizable, it results in a significant reduction of computational time for signature bases that can fit within GPU memory. However, as with TCAM-based approaches, this does require additional hardware, does not meet the obfuscation design goal, and has the additional shortcoming that larger signature sets require additional hardware to be purchased for each device. Similarly, in (*21*) (among many others), field-programmable gate array (FPGA) processors are used to implement high-speed matching of Snort rules. They develop an FPGA implementation of the "BV-TCAM" architecture, combining a binary vectorization (BV) of the input data with TCAM emulation. Notably, however, their (test) implementation is limited to 512 rules, with additional hardware required to support additional rules.

Our research is differentiated from the aforementioned efforts in four significant ways. First, we propose an algorithm that is implementable on a general-purpose computing platform, without requiring specialized hardware to be attached to an existing device. Second, we explicitly consider the obfuscation of signatures as a design goal, and exploit the pre-image resistance of hashing functions to achieve this. Third, all of the methods previously described that focus directly on emulating Snort signature matching require either storage or processing time that scales linearly with the size of the rule database, whereas our approach (given fully trained weight vectors) operates in constant time with respect to the size of the rule database. Finally, we focus specifically on on-device intrusion detection for resource-constrained systems operating in mobile, ad-hoc networks, rather than on standard fixed networks with higher throughputs and the ability to dedicate a computer to the exclusive task of acting as a NIDS sensor.

Kachirski and Guha have previously proposed an ad-hoc network intrusion detection structure in which agents residing within the network nodes perform the functional tasks of monitoring the state and traffic of the network, making decisions based upon the output of the monitoring, and acting upon the decisions (*5*). Cognizant of the same resource and bandwidth environmental limitations already discussed, their proposed structure uses a peer-election strategy to distribute network monitoring duties amongst the participants in the network with the goal of conserving the network's overall computational demand. Building upon both the agent framework characterized previously, as well as a case-based reasoning approach to network intrusion analysis developed by Schwartz et al (*6*), Guha et al. (*7*) propose an alternative implementation of Snort rule sets as an archive of cases with associated case features to be used as a basis for reasoning by the network nodes that have been designated for monitoring duty. Our contribution is differentiable from this work by proposing a scalable implementation of packet analysis that eschews the need for selective network monitoring node designation to maximize throughput and fault tolerance in the event one or more nodes in the ad-hoc network cease their participation.

Antrosiom and Fulp have previously proposed a strategy of continuously monitoring and scanning both conventionally wired and wireless ad-hoc networks for vulnerabilities (*8*). By periodically updating the vulnerability assessment of the network as a whole, nodes that have recently become vulnerable or been compromised can be quarantined via logical network segmentation, mitigating their residual impact upon the other participants in the network. The policy management and network configuration decisions in the proof-of-concept of this proposed solution are made by a centralized network node that hosts the defense system. Once again, our contribution is differentiable because of its emphasis upon decentralizing the network defense functionality by distributing a lightweight approximation of the packet analysis functionality normally provided by Snort to every node in the network.

Iland et al (*9*) have proposed techniques for detecting the presence of malware in ad-hoc mobile infrastructure with specific emphasis on the Android operating system (*10*). The proof-of-concept implementation of this work relies upon simulating both an ad-hoc network containing compromised Android hosts, as well as multiple "bot-herder" hosts participating in command-and-control traffic with the compromised hosts with a collection of virtual machines. Wireshark (*11*) is then deployed for packet and protocol analysis of the captured virtual network traffic after the conclusion of the simulation, and malware is detected by identifying distinguishable characteristics of both the Hypertext Transfer Protocol (*12*) and Domain Name Service protocol (*13*). Our contributions are differentiable because of the motivation for a more generalized defensive technology that is simultaneously able to identify any malicious traffic describable as a Snort rule regardless of the application protocol, as well as operate within a comparatively resource-depleted mobile network environment.

## 3. ELIDe Approach

ELIDe is proposed as a linear machine learning classifier that relies upon a conventional Snort implementation as its training oracle. It employs the "hash trick" of (*24*, *25*) in order to approximate a classifier in an extremely high-dimensional space with a lower-dimensional space, as will be described, thus gaining most of the benefits of classification in the high dimensional space without being forced to pay the price of performing computations in that space. The approach begins with characterizing a network packet as a collection of N-grams (N bytes of contiguous data from the packet). The feature vector that represents a particular packet is then constructed by counting the number of occurrences of each unique N-gram appearing in the data and indexing these counts with a collision-resistant hash digest (using the N-gram and an arbitrary salt as the input). Classification is then performed by computing the dot product of the representative feature vector with an internal weight vector (of equal size) and using the sign of the result as the classifier's decision. Due to the previously identified resource demands of Snort, ideally supervised learning would take place in a more conventional computing environment and not in the tactical network. The resulting weight vector can then be transmitted to ELIDe instances in the field as lightweight updates.

This approach has the benefit of projecting the network packet representation into a space with highly elevated dimensionality ($256^N$ possible N-grams), where performing linear classification to separate "good" packet data from "suspicious" packet data is far more tractable. However, operating natively in this high-dimensional space would also impose unacceptable memory and processing requirements within the draconian constraints of a device likely to be found in a tactical network (whose resources which will also likely be shared between the ELIDe solution and other applications).

In practice, the N-gram features in this high-dimensional space will be extremely sparse, particularly given the bandwidth constraints of a tactical network. The ELIDe approach, therefore, avoids storing the native N-grams and instead represents them as the lower-order bits of their respective hash digests. This effectively re-lowers the dimensionality of the problem space down to the only size required to represent the hash digests (for example, $2^{10}$ dimensions for 10-bit hash outputs). The length of the N-gram hash digest output becomes an implementation detail that represents the tradeoff between resource consumption (shorter hashes will be computed more quickly and take up less storage in memory) and accuracy (longer hashes provide more detail and lower the likelihood of two distinct N-grams colliding with the same hash output).

Finally, the weight vector used in the final stage of classification will be updated through supervised learning of a training data set (a sequence of packet data). Snort will be used as the oracle for this learning process: if an alert triggered a Snort alert, the desired outcome of the ELIDe classifier will be positive. Otherwise, the desired outcome will be negative. This approach also acts as obfuscation of the "signature" data in the event the device hosting the ELIDe instance is captured. While it is theoretically possible for the adversary to reverse-engineer the operation of ELIDe and determine its response to individual packets, it will be very difficult to exactly enumerate the full range of packets for which ELIDe fires an "alert" with just the weight vector alone.

## 4. Implementation and Results

The implementation of an ELIDe solution prototype consisted of three stages. The initial stage took the form of a prototype implemented in CPython version 2.7.3 (*14*) that used the NumPy (*15*) library for vector mathematics and the standard Python implementation of MD5 (*16*) as the N-gram feature hashing mechanism. MD5 is not considered a cryptographically secure hashing algorithm, but it is suitable as a hashing mechanism for the ELIDe concept since it is collision-resistant. Snort version 2.9.4, accompanied by the February 2013 release of the Sourcefire

Vulnerability Research Team (VRT) rule set, was used as ELIDe's training oracle for all exercises.

This implementation was able to functionally perform the required operations and achieved a true positive rate of 99.9% for hash digest lengths longer than 8 bits (see the previous discussion regarding the tradeoff of hash lengths). In addition, its peak memory consumption was profiled by Massif as 196 MB, or 16.3% of Snort's requirement of 1.2 GB. However, its runtime latency did not compare favorably with Snort when analyzing controlled packet data and amounted to approximately 5–20 times longer than that of Snort, depending upon the length of the hashed N-gram features (the mean runtime of Snort to analyze a packet capture dataset of 26 MB was approximately 30 s). After analyzing the prototype with standard Python performance profiling tools, the bulk of the latency was represented by the process of computing MD5 hash digests. The decision was then made to replace MD5 with the Murmurhash (*17*), a hashing algorithm known to have better performance characteristics than MD5.

In order to characterize the dependency of ELIDe's runtime performance and accuracy upon the configurable parameters of hash length and N-gram size, the Python implementation was tested using a data set consisting of 345320 packets captured from a synthetic virtual network known to contain traffic that triggers alerts from the Snort VRT rule set. The classifier was exercised using five different N-gram sizes (5 bytes, 10 bytes, 15 bytes, 20 bytes, and 25 bytes) as well as 13 different hash lengths (the inclusive range of 4 bits through 16 bits), and five trials were executed for each distinct configuration producing a randomized sequence of 325 trials (5 N-gram sizes x 13 hash lengths x 5 repetitions). In all cases, the response elicited from Snort for each packet (alert vs. no alert) was used as the ground truth to supervise the learning of the classifier.

Figures 1 and 2 visualize collected data representing the time required for the ELIDe classifier to process and classify the data set after supervised training has been completed as a function of the feature hash length and N-gram size, respectively. The timing data suggests the following conclusions:

- The use of Murmurhash instead of MD5 improved ELIDe's runtime latency to approximately 2–3 times greater than that of Snort (previously the latency was 5 to 20 times greater than that of Snort).
- There is noticeable correlation between runtime performance and the length of the feature hashes.
- There is NO noticeable correlation between runtime performance and the size of the N-grams.

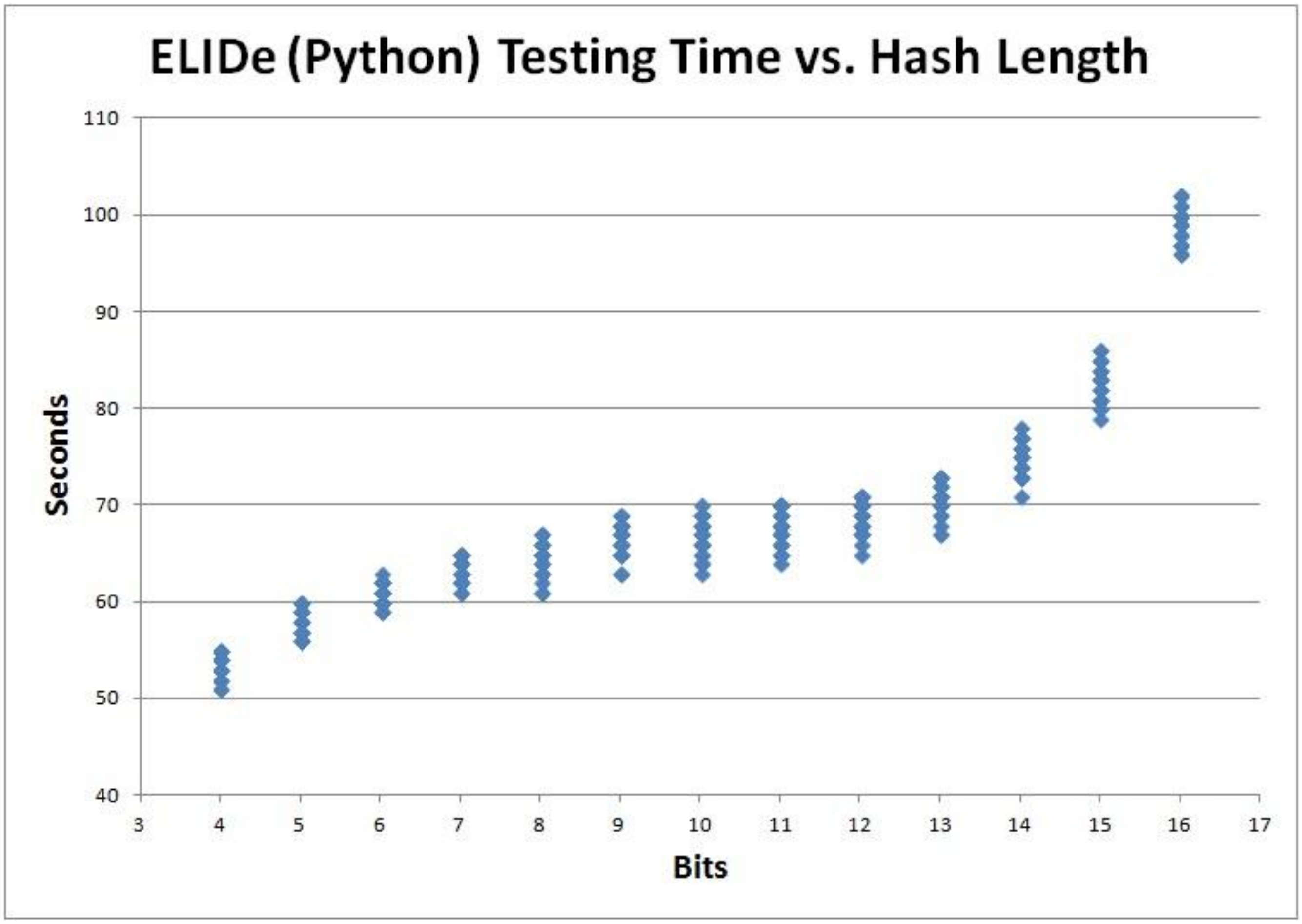

Figure 1. Time required by the Python ELIDe prototype to classify 26 MB of packet data as a function of feature hash length.

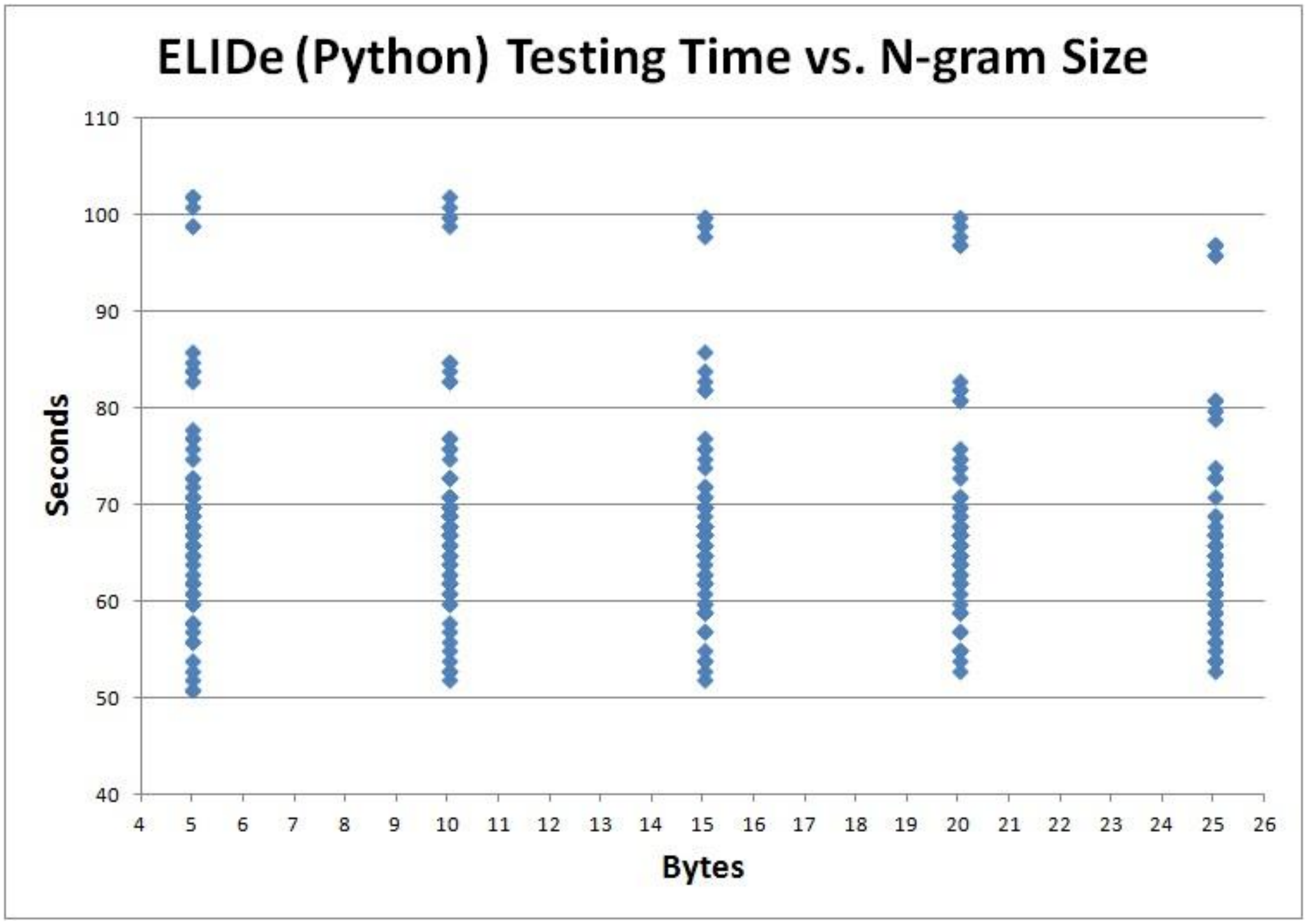

Figure 2. Time required by the Python ELIDe prototype to classify 26 MB of packet data as a function of N-gram size.

Figures 3 and 4 visualize collected data representing the true positive rate of the classifier as a function of the feature hash length and N-gram size, respectively. The classification accuracy results suggest the following conclusions:

- There is noticeable correlation between the true positive rate and the length of the feature hashes. If the feature hash length is greater than or equal to 8 bits, the classifier achieves a true positive rate between 99% and 100%.
- There is NO noticeable correlation between the true positive rate and the size of the N-grams.

Further performance profiling led to the conclusion that the majority of remaining latency was attributable to ELIDe's materialization as a Python prototype. This led to the second major phase of implementation that converted ELIDe to a C++ application. Murmurhash has both C++ and Python interface bindings, and was, therefore, retained as the N-gram feature hashing mechanism. The responsibility for performing vector operations was handed to the BLAS library, which is itself the underlying engine beneath the Python Numpy library previously used for this purpose. Profiling the memory usage of the C++ manifestation of ELIDe with Massif indicated that it required only 17 MB, or 1.42%, of Snort's RAM requirement. It was then subjected to the same set of trials (325 trials over five different N-gram sizes and 13 different hash lengths) previously executed by the Python implementation.

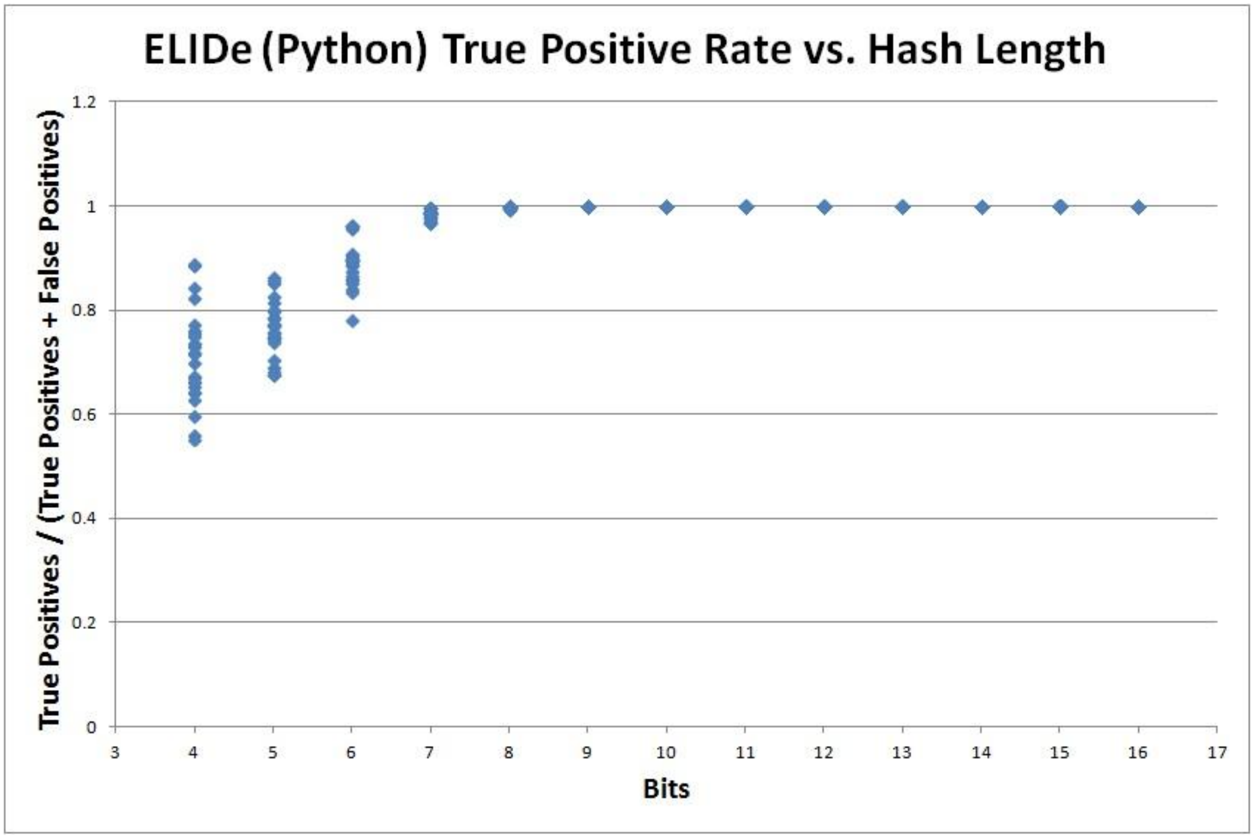

Figure 3. True positive classification rate of the Python ELIDe prototype as a function of feature hash length.

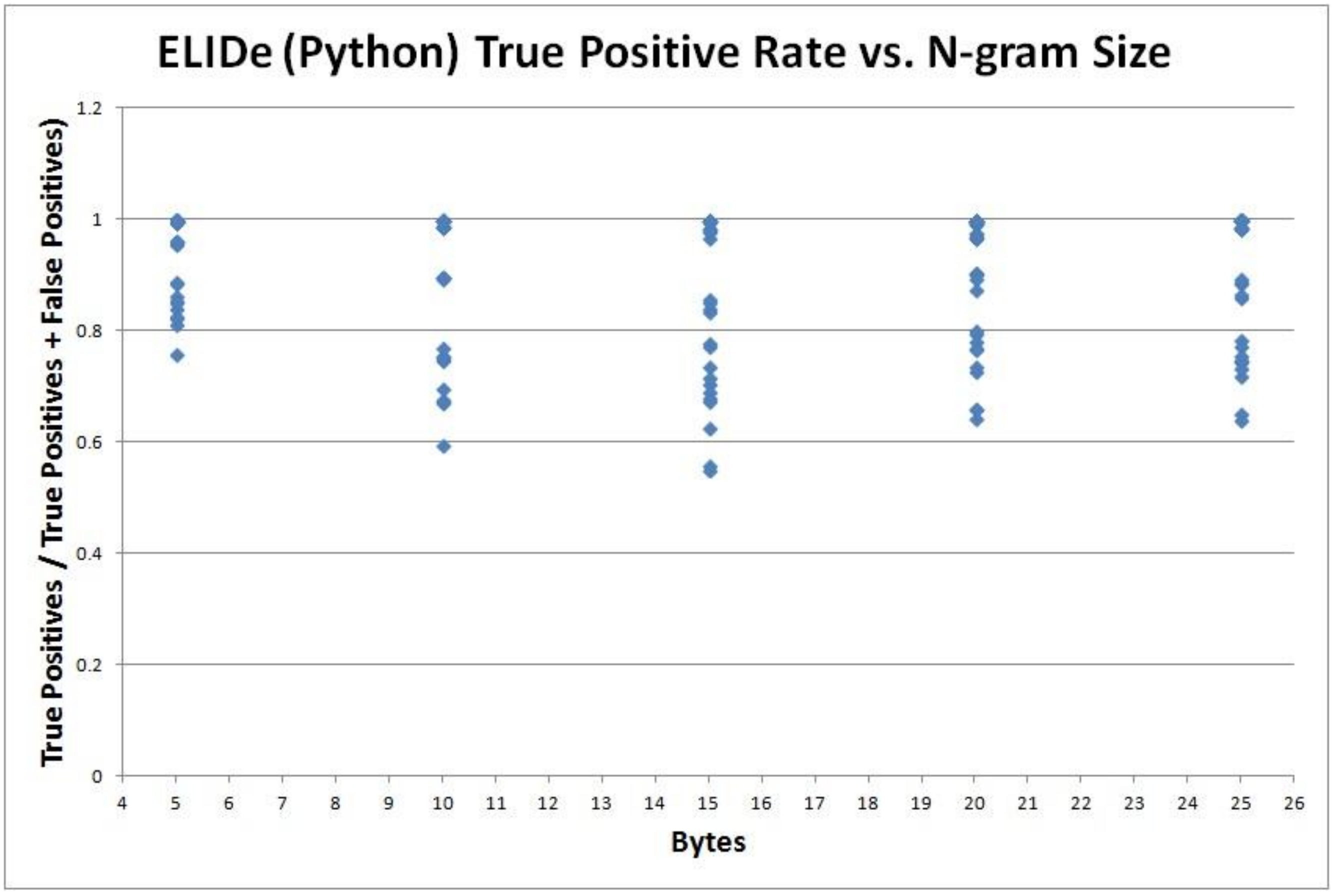

Figure 4. True positive classification rate of the Python ELIDe prototype as a function of N-gram size.

Figures 5 and 6 visualize collected data representing the time required by the C++ implementation to process and classify the same data set with which the Python prototype was evaluated. The results suggest that eliminating the overhead introduced by the Python interpreter resulted in a throughput that was 30 times faster than Snort in the best case and almost identical to Snort's throughput in the worst case.

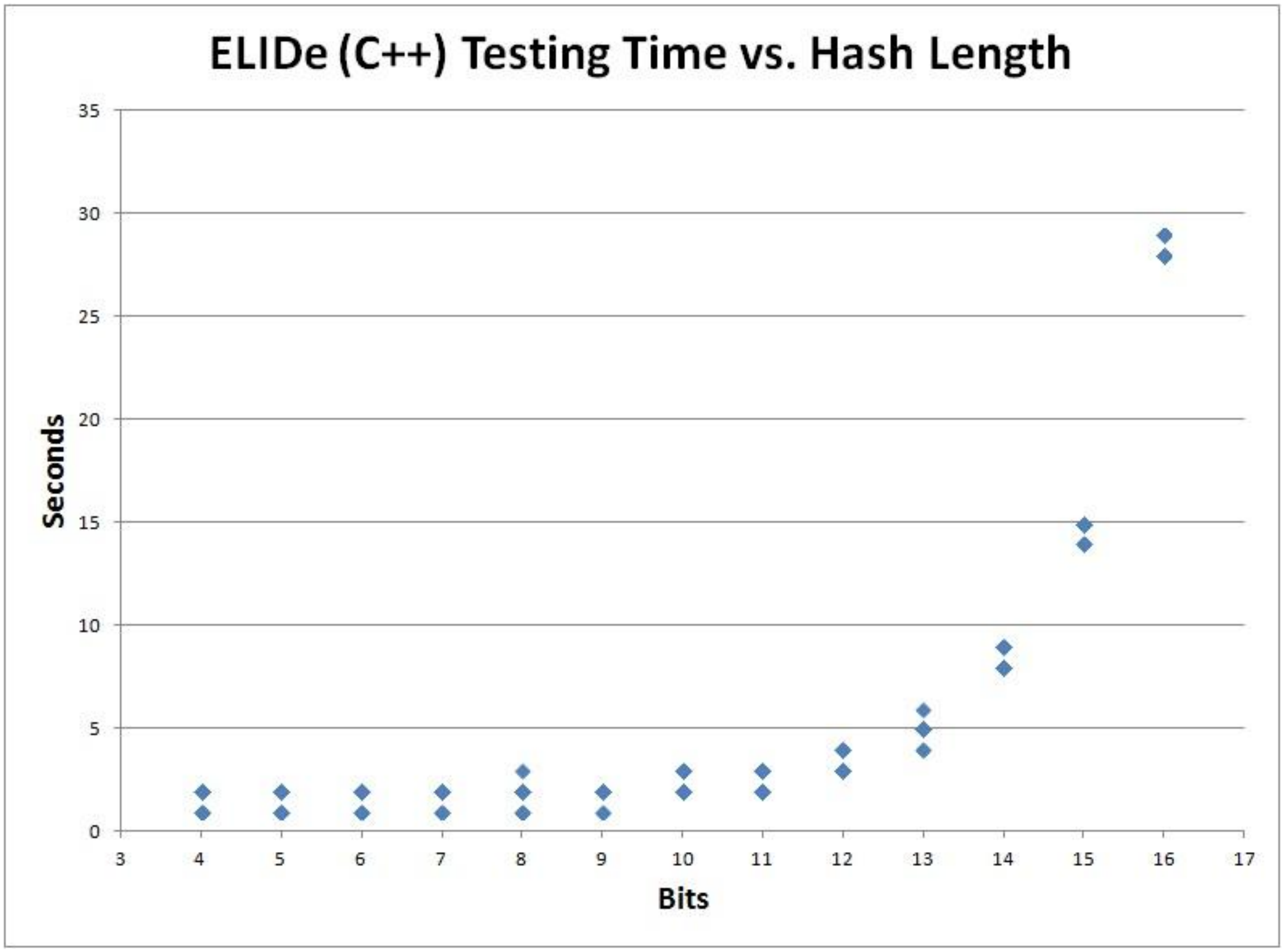

Figure 5. Time required by the C++ ELIDe implementation to classify 26 MB of packet data as a function of feature hash length.

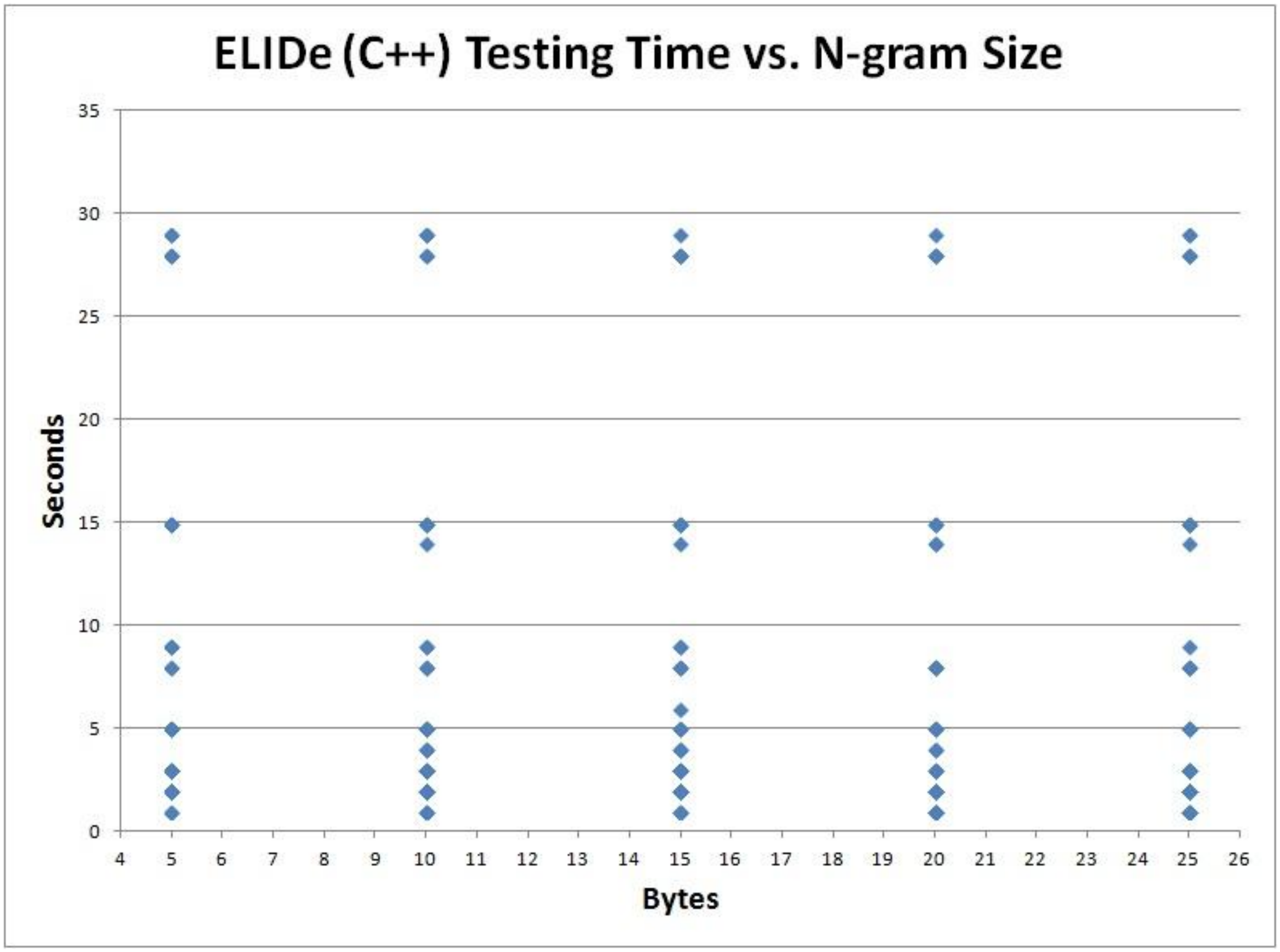

Figure 6. Time required by the C++ ELIDe implementation to classify 26 MB of packet data as a function of N-gram size.

Figures 7 and 8 visualize the data collected representing the true positive rate of the C++ implementation as a function of the feature hash length and N-gram size, respectively. These results verify that converting ELIDe into a C++ application did not adversely affect the accuracy of its classification.

The final stage of the implementation transitioned the ELIDe application onto a resource-constrained hardware platform more likely to be used in a mobile tactical network, and the Raspberry Pi was chosen as that representative platform. ELIDe was successfully tested on a Raspberry Pi, its throughput was benchmarked at approximately 8.3 megabits per second (using hashed N-gram features that were 10 bits in length) while retaining its functional characteristics and true positive rate.

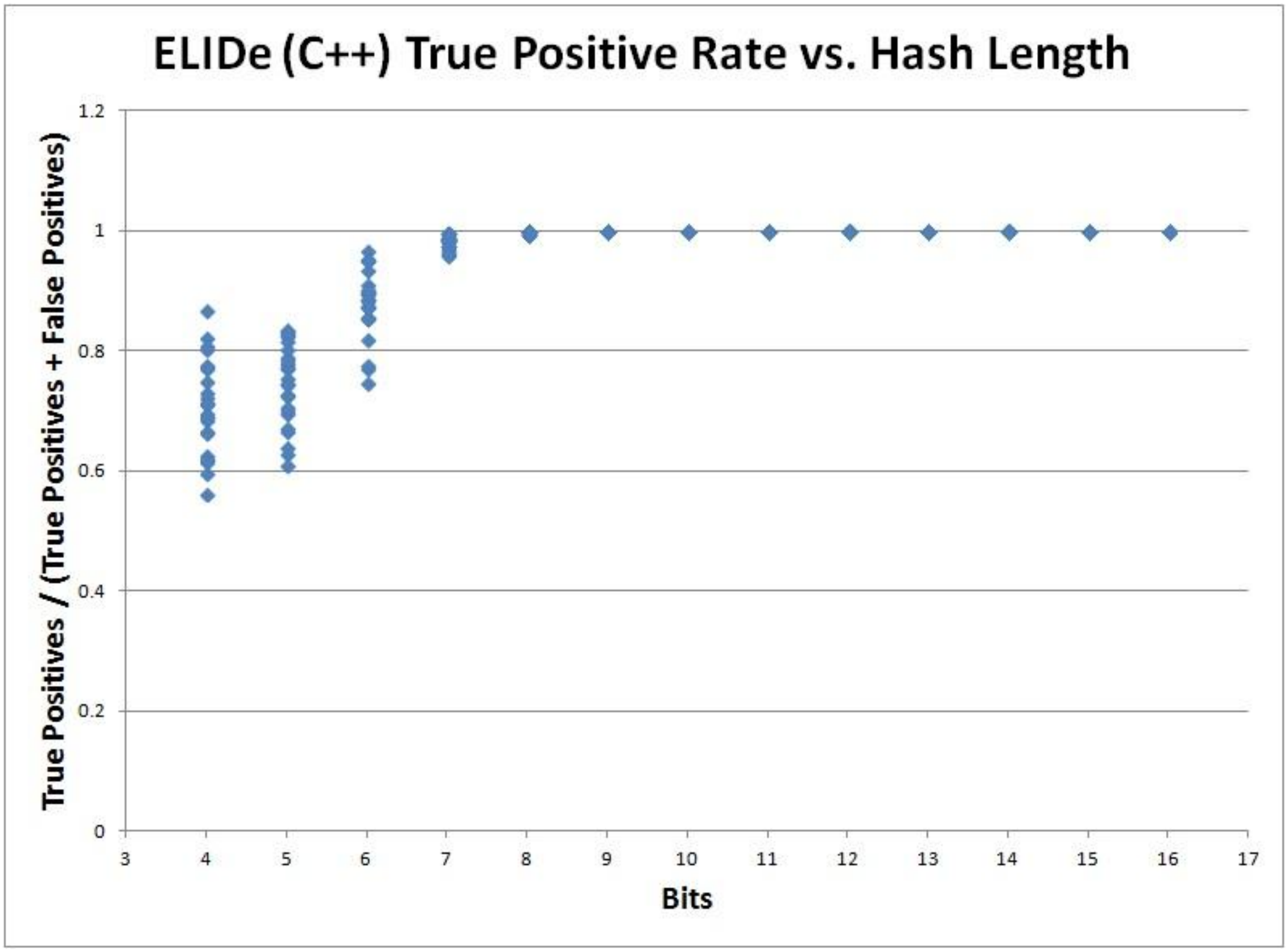

Figure 7. True positive classification rate of the C++ ELIDe implementation as a function of feature hash length.

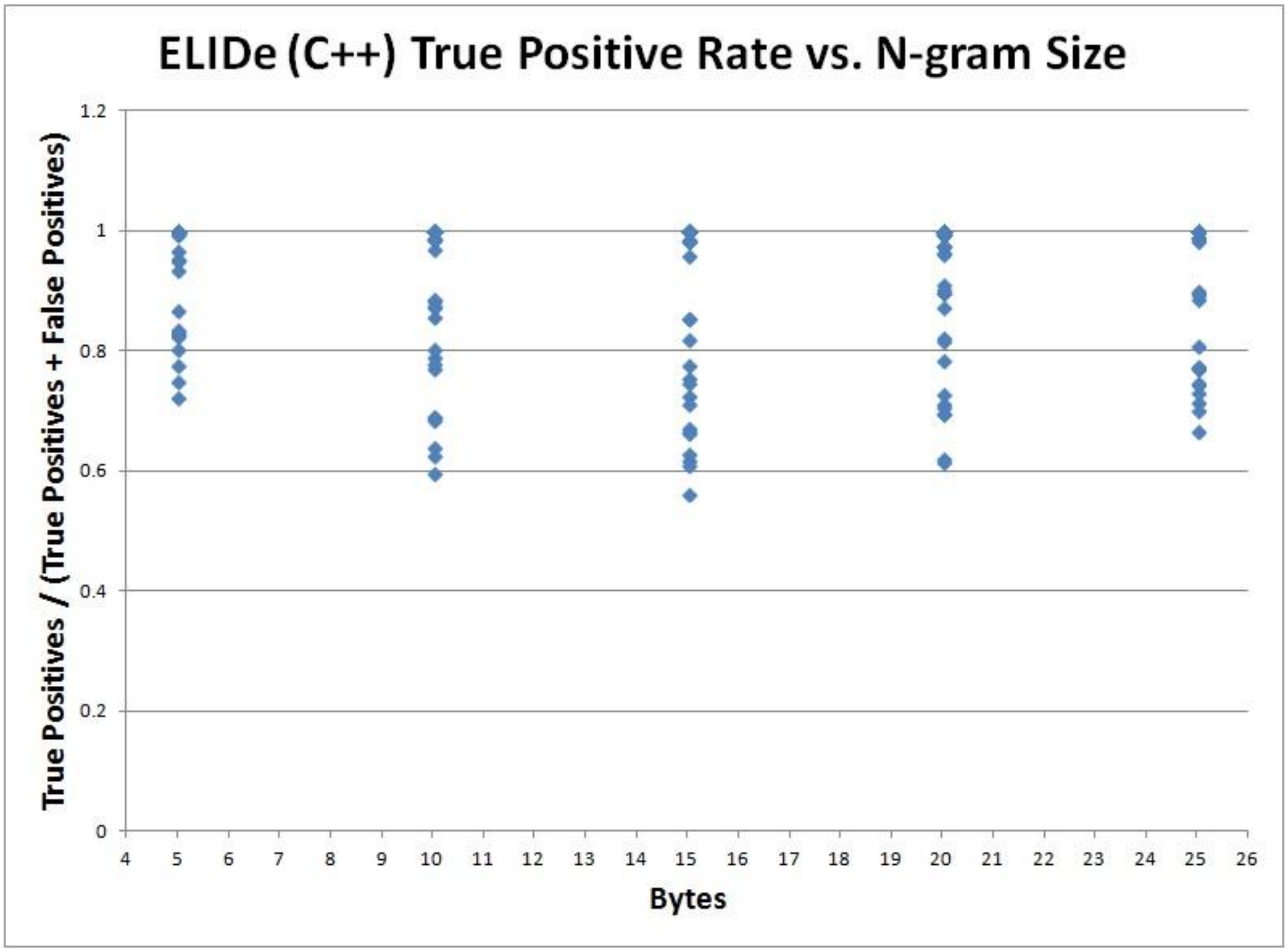

Figure 8. True positive classification rate of the C++ ELIDe implementation as a function of N-gram size.

## 5. Conclusion

ELIDe is the proof-of-concept for approximating the functionality of a robust network intrusion detection tool such as Snort for use in a mobile tactical network. The spartan processing and memory requirements make it ideal for dense coverage of every single node in a dynamic topology, and the lightweight constructs used to update ELIDe's signature baseline are obfuscated and suitable for transmission to devices for which physical loss is a possibility.

## 6. References

1. Snort intrusion prevention and detection system, http://www.snort.org
2. Snort rules, http://www.snort.org/snort-rules
3. Valgrind instrumentation framework. http://www.valgrind.org
4. Raspberry Pi single-chip computer. http://www.raspberrypi.org/
5. Kachirski, O.; Guha, R. Intrusion Detection Using Mobile Agents in Wireless Ad Hoc Networks. *Proceedings of IEEE Knowledge Media Networking Conference*, KMN’02, July 2002.
6. Schwartz, D. G.; Stoecklin, S.; Yilmaz, E. A Case-Based Approach to Network Intrusion Detection. *Fifth International Conference on Information Fusion*, IF'02, Annapolis, MD, July 7–11, 2002, pp. 1084–1089.
7. Guha, R.; Kachirski, O.; Schwartz, D. G.; Stoecklin, S. A. Case-Based Agents for Packet-Level Intrusion Detection in Ad Hoc Networks. *Seventeenth International Symposium on Computer and Information Sciences*, Orlando, FL, October 28-30, 2002.
8. Antrosiom, J. V.; Fulp, E. W. Malware Defense Using Network Security Authentication. In *Information Assurance, 2005. Proceedings*. Third IEEE International Workshop on, pp. 43–54. IEEE, 2005.
9. Iland, D.; Pucher, A.; Schäuble, T. Detecting Android Malware on Network Level, 2011.
10. Android operating system. http://www.android.com/
11. Wireshark packet analyzer. http://www.wireshark.org/
12. Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Masinter, L.; Leach, P.; Berneres-Lee, T. Internet Engineering Task Force RFC 2616. http://tools.ietf.org/html/rfc2616
13. Mockapetris, P. Internet Engineering Task Force RFC 1035. http://tools.ietf.org/html/rfc1035
14. Python programming language. http://www.python.org/
15. NumPy. http://www.numpy.org/
16. Rivest, R. Internet Engineering Task Force RFC 1321. http://tools.ietf.org/html/rfc1321
17. Appleby, A. Murmurhash. https://sites.google.com/site/murmurhash/

18. Antonatos, Spyros; Anagnostakis, Kostas G.; Markatos, Evangelos P. Generating Realistic Work-Loads for Network Intrusion Detection Systems. In *WOSP*, pages 207–215, 2004

19. Gould, Stephen, et al. Apparatus and Method for Memory Rfficient, Programmable, Pattern Matching Finite State Machine Hardware. U.S. Patent No. 7,082,044. 25 Jul. 2006.

20. Smith, Randy, et al. Evaluating GPUs for Network Packet Signature Matching. Performance Analysis of Systems and Software, 2009. *ISPASS 2009. IEEE International Symposium on. IEEE*, 2009.

21. Song, Haoyu; Lockwood, John W. Efficient Packet Classification for Network Intrusion Detection Using FPGA. *Proceedings of the 2005 ACM/SIGDA 13th International Symposium on Field-Programmable Gate Arrays. ACM*, 2005.

22. Sourdis, Ioannis, et al. Packet Pre-Filtering for Network Intrusion Detection. Architecture for Networking and Communications Systems, 2006. ANCS 2006. *ACM/IEEE Symposium on. IEEE*, 2006.

23. Sung, Jung-Sik, et al. A Multi-Gigabit Rate Deep Packet Inspection Algorithm Using TCAM. *Global Telecommunications Conference*, 2005. GLOBECOM'05. IEEE. Vol. 1. IEEE, 2005.

24. Shi, Qinfeng, et al. Hash Kernels. *International Conference on Artificial Intelligence and Statistics*. 2009.

25. Shi, Qinfeng, et al. Hash Kernels for Structured Data. *The Journal of Machine Learning Research* **2009**, *10*, 2615–2637.

26. Yu, Fang, et al. SSA: A Power and Memory Efficient Scheme to Multi-Match Packet Classification. *Proceedings of the 2005 ACM Symposium on Architecture for Networking and Communications Systems*. ACM, 2005.

| | |
|---|---|
| 1 | ADMNSTR |
| (PDF) | DEFNS TECHL INFO CTR |
| | ATTN DTIC OCP |
| | |
| 1 | GOVT PRNTG OFC |
| (PDF) | ATTN A MALHOTRA |
| | |
| 1 | US ARMY CYBER COMMAND |
| (PDF) | ATTN 24 C PRESSLEY |
| | |
| 9 | US ARMY RDECOM CERDEC |
| (PDFS) | ATTN RDER IWI G BERTOLI |
| | ATTN RDER IWI K BOYLE |
| | ATTN RDER IWI SP P ROBB |
| | ATTN RDER IWI SP S BLAIR |
| | ATTN RDER STI IS J SANTOS |
| | ATTN RDER STI S LUCAS |
| | ATTN RDER STI TN G ZIGLICH |
| | ATTN RDER STI TN M MAGENHEIMER |
| | ATTN RDER STI TN S SNYDER |
| | |
| 7 | US ARMY RSRCH LAB |
| (PDFS) | ATTN RDRL CIN D B RESCHLY |
| | ATTN RDRL CIN D C ELLIS |
| | ATTN RDRL CIN D D KELLY |
| | ATTN RDRL CIN D J COLE |
| | ATTN RDRL CIN D T PARKER |
| | ATTN RDRL CIN S C SMITH |
| | ATTN RDRL CIN D L M MARVEL |
| | |
| 2 | US ARMY RSRCH LAB |
| (PDFS) | ATTN RDRL SLE I A REVILLA |
| | ATTN RDRL SLE I D LANDIN |
| | |
| 17 | US ARMY RSRCH LAB |
| (PDFS) | ATTN RDRL CIN A KOTT |
| | ATTN RDRL CIN D G PAYER |
| | ATTN RDRL CIN D H CAM |
| | ATTN RDRL CIN D J EDWARDS |
| | ATTN RDRL CIN D J WITTKAMPER |
| | ATTN RDRL CIN D M SHEVENELL |
| | ATTN RDRL CIN D O AKPAN |
| | ATTN RDRL CIN D P GUARINO |
| | ATTN RDRL CIN D R ASTROM |
| | ATTN RDRL CIN D R ERBACHER |
| | ATTN RDRL CIN D R HARANG |
| | ATTN RDRL CIN D R PINO |
| | ATTN RDRL CIN D S HUTCHINSON |
| | ATTN RDRL CIN D W GLODEK |
| | ATTN RDRL CIN S C ARNOLD |
| | ATTN IMAL HRA MAIL & RECORDS MGMT |
| | ATTN RDRL CIO LL TECHL LIB |

INTENTIONALLY LEFT BLANK.