A new data set for evaluation of intrusion detection systems

Ronald Huizer (r.huizer@xs4all.nl)
Nova Southeastern University

Abstract

We present a model for the generation of NIDS evaluation data sets that aims to decouple the synthesis of benign and malicious application layer traffic from the underlying network topology used to transport this traffic. This separation allows us to more easily generate evaluation data sets that mix application layer traffic on different network topologies.

Introduction

Compiling data sets to measure the effectiveness of network intrusion detection systems, or NIDSes in short, is a large and complicated task. The metrics of greatest interest for NIDS evaluation are the rate of successful detection of threats, and the rate of false detection of threats, as the primary aim of an NIDS is to maximize the former, while minimizing the latter. A data set for the evaluation of NIDSes should ideally cover a large amount of identifiable threats, while at the same time allow them to be presented in the context of benign network traffic. Covering more threats allows the data set to be used to prime the NIDS to reduce the occurrence of false negatives, while covering benign traffic can be used to prime the NIDS to reduce the occurrence of false positives. Although NIDSes typically handle anomalies on many different protocol layers, in this paper we will focus exclusively on application layer traffic.

When considering the evaluation of NIDSes in the most ideal circumstances, we would run every existing program to abuse threats in as many different environments as possible, and evaluate how the NIDS behaves. Using the original programs rather than network data traces allows the dynamic behavior of such programs to be captured in the context of the network they are affecting, which models the dynamic character of the real world as much as possible. This approach however is also extremely impractical, as it is too time consuming to collect all of these programs, and apply them in a large variety of different network contexts. Moreover, this would rely on having access to a large variety of real networks, which will be hard to arrange. As such, we propose a model for such programs and contexts that can be effectively compiled and managed, while at the same time still covers our evaluation requirements.

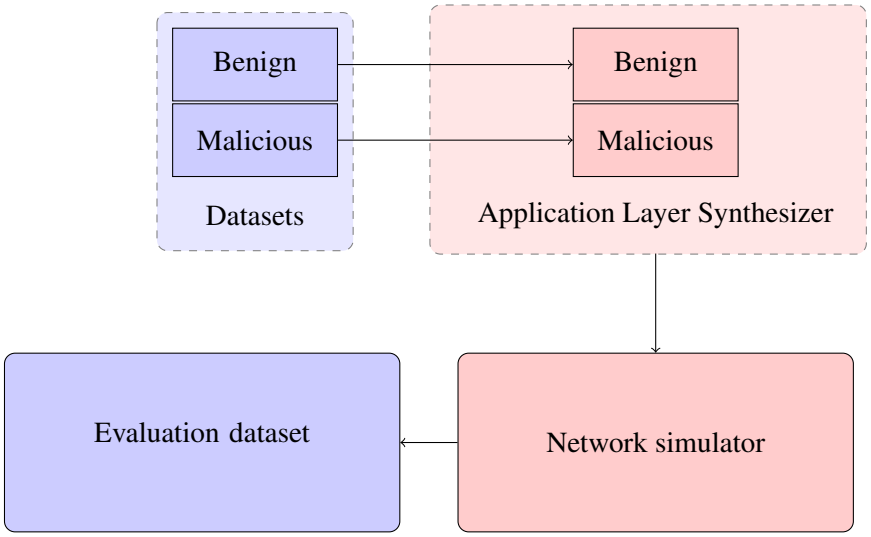

Figure 1. A model for data set generation.

Evaluation models

The most basic evaluation model consists of one or more data sets collected on a real network over a certain period in time, during which a large amount of normal traffic, and a small amount of threat based traffic is generated[1]. This will cover a real world scenario which can be used to prime the NIDS. The downside of this approach is that it is very rigid, and proper care should be taken to how evaluation results are handled. It is trivial to prime a NIDS to deal with the evaluation data set near perfectly, but doing so comes at little practical gain, as there is such a variety of different data sets that would not be optimized for. The model therefore can be considered a single instance of the problem space, and evaluators of the NIDS should be careful to not over-fit the IDS against this data set. The original MIT/DARPA data sets are an example of this, and McHugh (2000) discusses short-comings of the network context attacks are applied in, as well as potential targeted priming of NIDSes in section 4.3.

In the model we propose, we differentiate between the modeling of benign and malicious application layer data sets, and the modeling of the underlying network environment in which these attacks occur. Figure 1 illustrates this better, and we can see traces of benign and malicious traffic are used to synthesize application layer traffic, which are then projected onto a simulated network environment to produce an evaluation data set. Although this model is similar to other approaches, such as those proposed by Massicotte et al. (2006) and Sommers et al. (2005), it differs in such that it decouples the application layer from the network layer completely. In the rest of the paper will will focus on the part of this model that projects application layer traffic onto a simulated network.

[1] Although large amounts of threat based traffic are common in distributed denial of service attacks, we do not consider such traffic worthwhile for evaluating NIDSes, as the nature of this threat is that it gives itself away when successfully used.

Application layer model

Generating both benign and malicious application layer traffic by itself is a complicated problem, and a lot of research has been done in this area. Although this part of the model it outside of the scope of our work, we will briefly summarize the work previously done in this area. Most of these approaches have been coupled with the network model, such that creating data sets in different environments is a time-consuming task.

Sommers, Yegneswaran, & Barford (2004) present MACE, which is able to compose malicious traffic based on a set of rules that describe the attack sequence, how the attack can be obfuscated, how victims are chosen, and background network traffic. However, the resulting malicious data sets were tied to a physical network by running MACE on this network using a propagation delay emulator.

Sommers et al. (2005) expand on previous work, and combine MACE with a benign traffic generator that statistically mimics the behavior of lower-layer network protocols. Their approach however does not decouple the benign and malicious traffic generators from the underlying network model.

Shiravi et al. (2012) use a specific language for describing attacks, called α-profiles, to generate malicious traffic. Benign traffic is synthesized based on statistical profiles, so called β-profiles, constructed from large sets of benign internet data traffic. These profiles have been applied in the context of a physical network setup. Although the authors mention the use of network simulators for generating these data sets, they disregard them due to the lack of required functionality. With our work, we hope to show it is practical to use network simulators to assist in the generation of data sets for NIDS evaluation in different network topologies.

Network models

NIDS systems are primarily interested in monitoring application layer data for potential threats, as the largest set of attacks use the application layer as a vector rather than the underlying protocol layers. It may therefore seem irrelevant to evaluate the NIDS in different network contexts, and just evaluate its performance on application layer data. However, when applied in a real network environment, a NIDS system is not only responsible for evaluating application layer data, but, as the NIDS takes link-layer packets on the network as input, also for reconstructing said application layer data from lower protocol layers. This includes properly dealing with fragmentation, packet loss, time-outs, retransmits et cetera. As such, it is important to evaluate the NIDS system on different networks. Further discussion of why replaying captured traffic is not a suitable method for NIDS evaluation can be found in Allen & Marin (2003).

In order to model the underlying network architecture, we can consider using a network simulator to generate either synthetic protocol data, or replay existing application layer data in a simulated network. Synthesizing network data has been discussed previously by Sommers, Kim, & Barford (2004) for IP, TCP, and UDP data, as well as Cao et al. (2004) for HTTP specific data. However, it should be noted that this project aims at synthesizing network packet data that stochastically behaves like real data on simulated networks, but makes no attempt to synthesize application layer content that is to be evaluated by the NIDS.

Methodology

The approach we have taken for our network model is based on the work of Weigle et al. (2006), who describe a method for replaying application level data on a simulated network model, and is closer to what we would like to accomplish. Their approach distills application layer data and its ordering dependencies from captured network data. This information is stored in application data units, or ADUs, which contains enough information to replay an application layer stream between a server and a client based on timing information and the ordering of data, independent on what is happening on lower networking layers. Their software demonstrates the mixing together of separate TCP streams, but it does not deal with the content of these TCP streams, nor is their network topology suitable for generating NIDS evaluation data sets. Furthermore, it is only possible to start new connections based on an absolute time, rather than one that is relatively ordered to other ADUs. This prevents us from capturing the ordering dependency between new connections and ADUs cleanly.

Using the NS-3 network simulator – see Riley & Henderson (2010) – we have implemented a topology that models the internet, in which one or more attackers reside, a local area network, in which the target machines reside, and a core router between them. Each node on the internet is modeled as a peer-to-peer connection to the router, with an arbitrary delay, throughput and loss-rate. This captures part of the behavior of the internet in the real world, although improvements can still be made with regard to asymmetric routing, and packet reordering.

We then define a mapping between ADUs from different TCP streams, and nodes on our network. This mapping also performs address translation between the end-points in the TCP stream and the ones used in the simulated topology. A simulated run of the network will replay the application layer ADUs we have generated between nodes on the network, and a trace point on the local area network interface of the router can be used to record a capture file of all data exchanged between the internet and nodes on the local network. As the router is in a similar place to where a NIDS system would reside on the local area network, this provides us with a fairly representative trace of mixed benign and malicious traffic in this particular network topology. Because this approach decouples the application layer component of benign and malicious traffic from the underlying network, it is fairly easy to adapt our approach to a different network topology.

Separation of benign and malicious traffic

When the synthesis of benign and malicious traffic involves using real traces, it is necessary to determine what part of the real traces are benign and what parts are malicious. Solving the problem of differentiating between benign and malicious traffic sets is outside of the scope of this paper, but is discussed by Sommers et al. (2005). However, it is important to observe that this separation is not weakened by our proposed decoupling, as mixing benign and malicious traffic together on our virtual network topology is a process that can easily be traced. We have a-priori knowledge of what benign and what malicious traces will be initiated by what nodes in the topology at what times, and this knowledge can easily be used to cross-reference the generated evaluation data set. In case more granular knowledge is necessary, it would be possible to taint all packets belonging to malicious traces, and propagate this taint flag for all derived packets on the simulated network topology. This

allows us to differentiate between benign and malicious data in the generated data set in a packet granular fashion.

Database design

The generated evaluation data set has been stored in a database that captures network data from the IP layer, can be queried based on protocol properties, and is annotated by specific attack information[2]. The ER diagrams for our database can be found in the appendix. Currently IP, UDP and TCP packets are supported (although both IP and TCP options have been ignored), and the full packet is stored in the database, as well as any packet headers, such that the database can be easily queried based on these headers. A raw byte form of the packet is also stored in the database, as this can be used to easily reconstruct the packet capture trace itself.

Because our approach does not taint network packets at this time, the *AttackPackets* table is currently unused, although it has been included for the sake of completion. Currently, malicious traces can be located in the evaluation file by looking at the data and IP addresses in the evaluation database based on the timestamp in *Attack*.

Improvements

Causal ordering

As discussed previously, ADUs capture the causal relationship between application layer requests and responses in protocols, which allows for variations in transport layer behavior that does not violate the consistency of the application layer exchanges. As an example of this, it is not possible for a response to be received before a request has been made. We have ensured that ADUs to create new connections are ordered in the same way. However, ADUs that belong to different connections are currently not synchronized, and if there is a causal dependency between them, this dependency may be lost. A simple example of this would be a proxy server, with a high and a low throughput link. As in our model the high throughput link would be independent from the low throughput link, it is possible for the proxy to send data to an endpoint without having received it first. As such, the resulting data set may not be a truly valid representation of the exchange, but NIDS systems should also avoid relying on this inter-connection dependency as much as possible, as overly constrained assumptions will limit the effectiveness of the NIDS.

References

Allen, W. H., & Marin, G. A. (2003). On the self-similarity of synthetic traffic for the evaluation of intrusion detection systems. In *Proceedings of the 2003 symposium on applications and the internet* (pp. 242–). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dl.acm.org/citation.cfm?id=827273.829235`

[2]We have not implemented taint propagation as discussed previously, so currently this annotation captures the start of the attack, the type, and the end-points involved.

Cao, J., Cleveland, W. S., Gao, Y., Jeffay, K., Smith, F. D., & Weigle, M. C. (2004). Stochastic models for generating synthetic http source traffic. In *Infocom*. Retrieved from http://dblp.uni-trier.de/db/conf/infocom/infocom2004.html#CaoCGJSW04

Massicotte, F., Gagnon, F., Labiche, Y., Briand, L., & Couture, M. (2006). Automatic evaluation of intrusion detection systems. In *Proceedings of the 22nd annual computer security applications conference* (pp. 361–370). Washington, DC, USA: IEEE Computer Society. Retrieved from http://dx.doi.org/10.1109/ACSAC.2006.15 doi: 10.1109/ACSAC.2006.15

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Trans. Inf. Syst. Secur.*, *3*(4), 262-294. Retrieved from http://dblp.uni-trier.de/db/journals/tissec/tissec3.html#McHugh00

Riley, G. F., & Henderson, T. R. (2010). The ns-3 network simulator. In K. Wehrle, M. GÃŒnes, & J. Gross (Eds.), *Modeling and tools for network simulation* (p. 15-34). Springer. Retrieved from http://dblp.uni-trier.de/db/books/collections/Wehrle2010.html#RileyH10

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, *31*(3), 357-374. Retrieved from http://dblp.uni-trier.de/db/journals/compsec/compsec31.html#ShiraviSTG12

Sommers, J., Kim, H., & Barford, P. (2004, June). Harpoon: a flow-level traffic generator for router and network tests. *SIGMETRICS Perform. Eval. Rev.*, *32*(1), 392–392. Retrieved from http://doi.acm.org/10.1145/1012888.1005733 doi: 10.1145/1012888.1005733

Sommers, J., Yegneswaran, V., & Barford, P. (2004). A framework for malicious workload generation. In *Proceedings of the 4th acm sigcomm conference on internet measurement* (pp. 82–87). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1028788.1028799 doi: 10.1145/1028788.1028799

Sommers, J., Yegneswaren, V., & Barford, P. (2005). Toward comprehensive traffic generation for online ids evaluation..

Weigle, M. C., Adurthi, P., Hernández-Campos, F., Jeffay, K., & Smith, F. D. (2006, July). Tmix: a tool for generating realistic tcp application workloads in ns-2. *SIGCOMM Comput. Commun. Rev.*, *36*(3), 65–76. Retrieved from http://doi.acm.org/10.1145/1140086.1140094 doi: 10.1145/1140086.1140094

Appendix
ER Diagrams

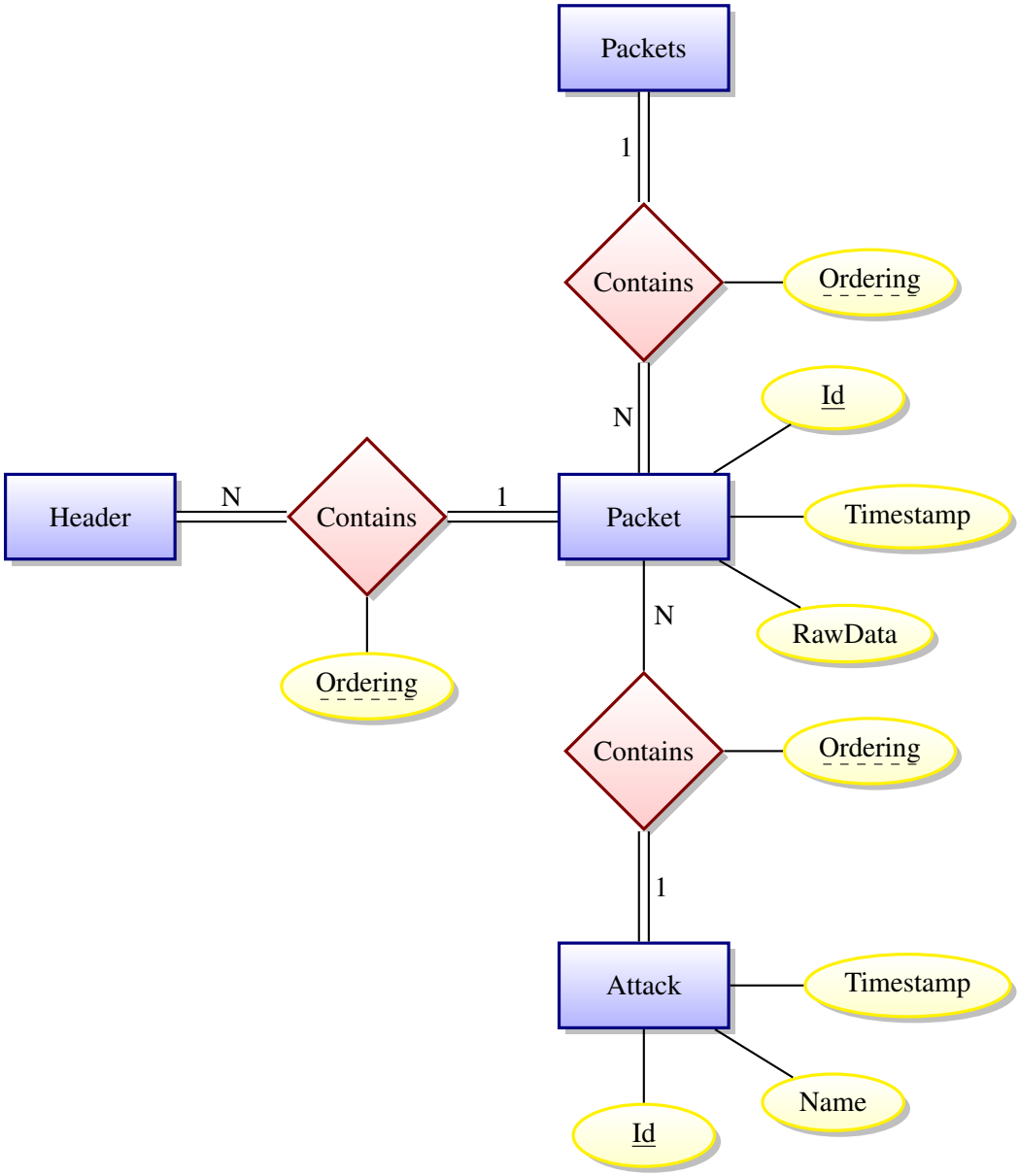

Figure A1. ER diagram of packet headers

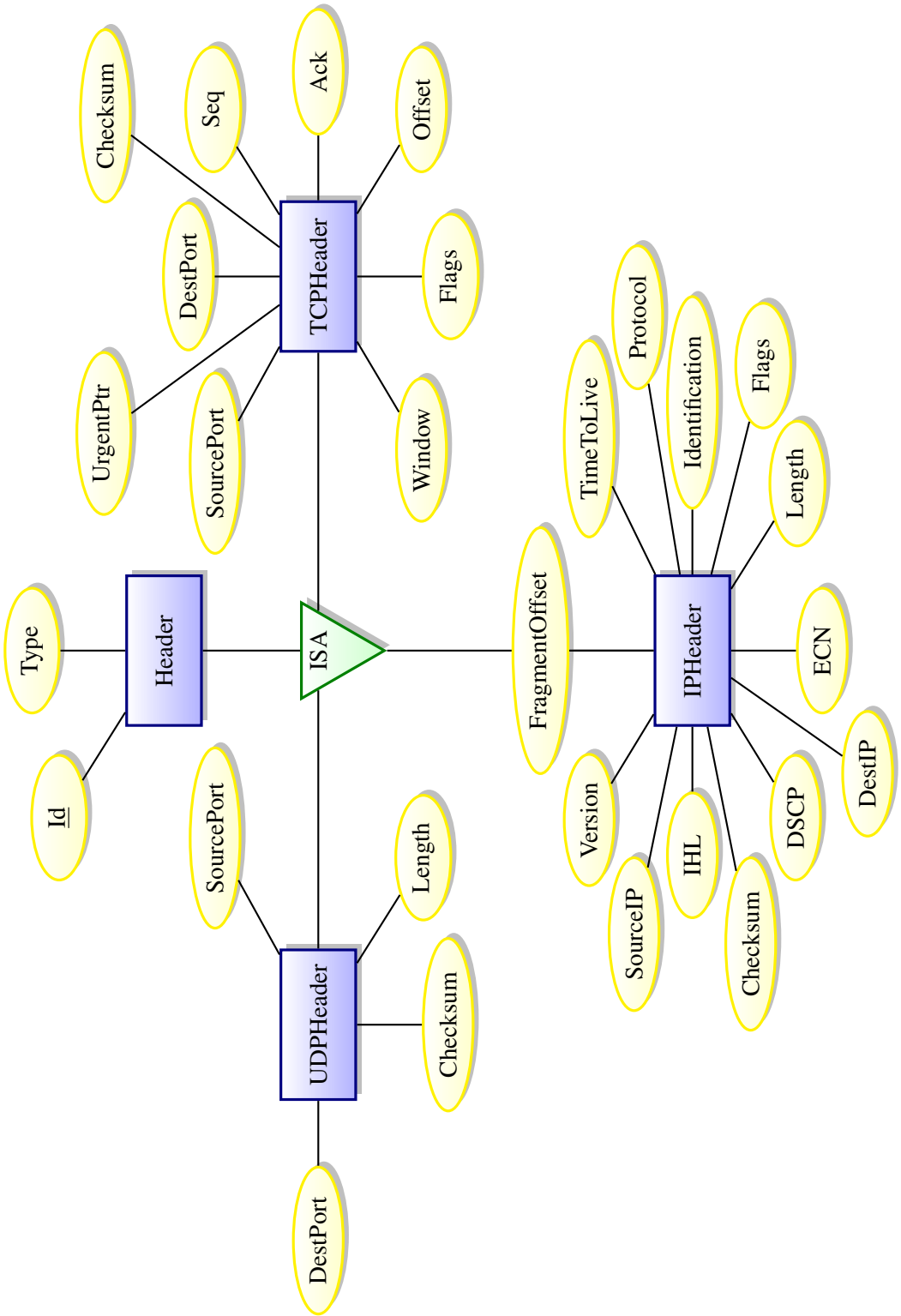

Figure A2. ER diagram of packet headers