Principles for Developing Comprehensive Network Visibility

Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, Nicholas Weaver
International Computer Science Institute

Abstract

We argue that for both defending against attacks and apprehending the scope of attacks after they are detected, there is great utility in attaining views of network activity that are *unified across time and space*. By this we mean enabling operators to apply particular analyses to both past and future activity in a coherent fashion, and applied across a wealth of information collected from a variety of monitoring points, including across administratively independent sites. We outline the core design goals necessary for building systems to develop such visibility in an operationally viable way.

1 Introduction

Increasingly, network operators, security analysts and system administrators are faced with wide-scale security issues that are not easily tackled with information from a single vantage point over a small window of time. Today's serious attacks manifest over extensive timescales and often involve disparate components of the targeted systems. However, monitoring efforts are often burdensome because operators lack a unified view of network activity that includes all components of their network, as well as a wide swath of time. These attacks leave analysts trying to ferret out answers to key questions: How did the attackers get in? What did they do once inside? Where did they come from? What patterns of activity serve as *markers* that reflect their presence? How do we prevent this attack in the future?

Analysts must tackle these questions using a multitude of monitors (e.g., NIDS, firewalls, NetFlow data, service logs) and so must first use custom, generally ad-hoc, analysis procedures tightly scoped to each source, and then gather these disparate bits of analysis together to form a high-level view of the scope of an attack. Pursued in this fashion, the process is tedious and error prone.

As a simplified example, consider a NIDS that develops an understanding that an *ssh* session originates from a host that previously scanned the network. From the NIDS' vantage point, since the session is encrypted the operator cannot tell if the attacker has successfully logged into the *ssh* server, nor, if so, which account has been compromised. Meanwhile, looking at the *sshd* log on the end system may reveal successful access to an account, but without the context of the attacker's previous scanning activity, which makes such access then suggestive of a compromise. An operator ideally needs to understand that (i) the peer host is an attacker, (ii) which, if any, account on the *ssh* server has been compromised, (iii) what additional host-level activity the attacker conducted on the *ssh* server and (iv) whether and which additional network services within the enterprise may have been accessed by the attacker. Gathering the full context of the activity requires drawing upon a multitude of logging information from a variety of sources, and then conducting nuanced analysis to form an understanding of the extent and implications of the attack. Further, detecting similar activity *in the future* requires additional effort that today for real-time monitoring often takes a significantly different form.

Much of the emphasis for today's intrusion detection systems (IDSs) focuses on a particular type of monitoring data (e.g., host logs or network packets) rather than synthesizing a broader view across multiple sources (e.g., keystrokes as seen on a host, past DNS lookups, honeypot data, internal NetFlow records, correlations between network activity and local desktop input or lack thereof). In addition, responding to attacks entails not only real-time detection but also, and just as importantly, post facto forensics. Operators can only answer crucial questions about the scope of an intrusion and the breadth of possible damage by drawing upon high-quality logs of past activity. Storing disparate forms of information in a unified fashion can not only save operators time in finding the answers to their questions, but also renders the process less error-prone.

Another critical situation that can greatly benefit from a unified system view is the problem of combating insider attacks. Dealing with insider threats has similarities to that of detecting intrusions, but also some salient differences. The scope of insider attacks is often difficult to apprehend at the time of detection because the full extent of an attack can involve a sequence of steps each of which the attacker had authorization to perform. Only the specific *combination* of steps—such as first accessing sensitive data, then transferring it to an intermediary machine, and finally from there exfiltrating it to a non-authorized third party—constitutes the malicious activity. When security officers eventually discover the attack, it becomes crucial to understand the full implications of the damage, which requires access to readily searchable logs (i) of a wide range of the computing and network environment's activity (ii) over a large window of time.

The benefits of developing some form of unified view of network activity across time and space are clear. However, developing a system that is both inclusive and useful, as well as operationally viable, is highly challenging. It is not enough to unify a given type of logging information (e.g., *syslog* servers); we need the means to form holistic views across different forms of monitoring (e.g., unifying NIDS results with inferences from system logs).

Finally, we can gain great utility by broadening the spatial range of network visibility beyond individual sites. There has been considerable activity, in terms of both research and operational systems, oriented around sharing information across sites about network attacks and attackers. The operational systems range from collecting alerts from intrusion detection systems and firewalls [3] to publishing information about vulnerabilities and exploits [2, 12] to live feeds of purported malicious activity such as sourcing of unsolicited email [10, 1], phishing [9], and botnet command-and-control [5]. These efforts focus heavily on sharing information at a *global* scope, where the producers and consumers of the information generally have no established relationship with one another. Consequently, concerns of privacy limit the types and granularity of information that sites are willing to contribute to the data sources, and concerns of trust limit the degree to which information from the sources can prove "actionable" in terms of incorporating it into automated response, or deciding to dedicate always-scarce analyst time to investigate the implications of the information.

We argue that we can gain both greater power and greater utility by considering information-sharing systems with a more *restricted* scope. Rather than operating globally, the "sweet spot" for such information sharing occurs for exchanging information between a set of sites with similar threat models that have explicitly decided to work with each other. Such sites needn't fully trust one another, but can afford to automate many of the steps involved in coordinated attack analysis because they presume that *usually* the other sites will act in a responsible manner—and, critically, they have administrative recourse (can complain to bosses) when that fails. Thus, we seek the means to provide coherent forms of analysis not only within sites but across cooperating sites.

In this paper we endeavor to frame the core notions necessary for developing unified monitoring of network system activity, drawn from our own efforts at developing such an architecture. We present our thinking in terms of a set of design principles that we argue such a system needs to possess. Our goal in this work is to both influence current efforts to build broad monitoring infrastructures, as well as solicit comments on possible omissions or modifications to the guidelines. We present our set of design principles in § 2. We then discuss extensions and additional uses of a system built around our guidelines in § 3. We conclude in § 4.

2 Design Guidelines

We now turn to developing the essential guidelines on which we believe a unified network monitoring infrastructure should be built. We divide the guidelines into three categories: (i) overarching basic notions, (ii) the data in the system, and (iii) critical capabilities.

2.1 Basic Guidelines

We first sketch several base guidelines that we believe define the basic structure of a unified network visibility system in ways that make the infrastructure viable for operational deployment and use.

B.1: Scope. The fundamental scope of a unified network activity monitor will be an administrative domain (e.g., an enterprise or a department). This scoping keeps such a system operationally viable (see $B.3$ below) for several reasons: (i) the issues of data sensitivity within this scope are well known and already dealt with, (ii) the logistics of collecting and archiving the data are manageable (e.g., information volume), and (iii) within this scope operators have solid notions of how the network is run, which can be crucial when setting up monitoring and analyzing the resulting collection of data (e.g., key services to instrument and watch).

While we argue that the basic scope of a network activity monitor should correspond to an administrative domain, we do not preclude such systems from being integrated together to form a larger monitoring infrastructure (as discussed above). To the contrary, we believe such larger infrastructures have benefits and encourage exploration into such structures. We explore this idea in more detail in § 3.2.

B.2: Incremental Deployability. An important aspect of gaining a unified vantage point on network activity is that such a system must not rely on interfacing to *everything*. The system should accept information from whatever data sources will provide data. Further, analyses on the collected data should not fail because of a lack of full data—and in fact should explicitly indicate that the lack of data has an impact on the analysis where possible. For instance, consider an operator hunting for a particular requested URL and an analysis process that can understand that a particular host runs an HTTP server (inferred from NetFlow summaries, say) but does not submit HTTP logs to the unified data collector. The analysis process can then suggest "data not available" for the given (presumed) web server, rather than the less informative "no matching URLs found".

B.3: Operational Realities. A system that provides network visibility must respect the operational constraints of a given situation. The system should leave operators in control of what data is accumulated, whether the data is sanitized, how that is accomplished, and with what retention policies. The system should not be a "black box" that administrators do not understand but are expected to simply trust.

B.4: Restricting and Logging Access. Finally, we must ensure that access to the information is properly controlled and logged. The sensitivity of the information such a system could contain is clearly high and therefore the data store may be a juicy target for attackers. Ideally, logging about the repository would be to a distinct service such that nefarious use could be detected without a malicious user being able to readily cover their tracks.

2.2 Data-Oriented Guidelines

Next we focus on guidelines that involve the data required to gain a unified view of network activity.

D.1: Data Breadth. The power in a unified infrastructure derives from its being able to interface with a multitude of activity monitoring systems, ranging from application logs to observations made by intrusion detection systems, firewalls and routers. Additionally, while in $B.1$ we scope basic operation to a single organization, the design of the infrastructure should accommodate data from sources outside the organization (see § 3.2). Along with data variety, the system also needs to scale to accommodate a large number of data providers.

D.2: Large Measurement Window. Serious breaches often manifest over long time intervals, and therefore a network activity monitor should be designed to track as much past activity as possible. Host compromise may happen well before activity revealing the compromise. Being able to understand all activity undertaken by a compromised machine has clear benefit in terms of both assessing further compromise within the organization and also understanding any missed signs that would have pointed to the compromise earlier. Both of these illustrate the clear need for a rich set of information covering as much of the past as feasible. Ideally, the window into the past would be unlimited. However, pragmatically this will clearly not be the case and so the next two guidelines are developed to cope with limited resources for data storage.

D.3: Smart Storage. As noted above, logistics will ultimately creep into a network activity monitor, and we will have to sacrifice existing data to make room for the never-ending torrent of new data. As a first pragmatic step towards keeping the window into the past as large as possible, a monitoring system should provide hooks for operators to setup smart data retention policies.

As an example, consider the Time Machine discussed in [7]. This system can buffer several days of high-volume packet-level network traffic using commodity hardware. It relies on the simple but crucial observation that, due to the heavy-tailed nature of network traffic [8], the system can record most connections in their entirety, yet skip the bulk of the total volume by only storing up to a cutoff limit of bytes per connection. Using this heuristic proves to be an enormous gain in terms of the length of the monitoring window.

In abstract terms, the existing Time Machine's operation relies on three concepts: *(i)* a high-volume stream of input that we want to store; *(ii)* a set of rules identifying how to separate more important input (e.g., the head of a session) from less important input (e.g., the heavy tail), which are used to trade off between comprehensive storage and available resources; and *(iii)* an aging mechanism (e.g., when the storage space fills up, the system expires the oldest packets from the archive).

We observe that these three notions are in fact quite general and independent of the type of data being managed. A generic network activity monitor should provide filtering via operator-defined hooks to establish policies to be setup that can lengthen the overall monitoring window by sacrificing unlikely-to-be-useful data.

D.4: Graceful Degradation. Aging mechanisms needn't simply discard data. Rather, they can employ *graceful degradation*, reducing data into more compact but still representative forms. To continue with the Time Machine example from above, before discarding old packets the system could instead distill these discarded packets into NetFlow-like connection records. A later aging step might further aggregate these connection summaries into traffic matrices of which hosts communicated with which others and the total volume of data exchanged over a given time interval. Degradation could also involve random subsamples, allowing some individual records to still be characterized. When aging in this fashion, the history loses granularity over time, yet still retains a degree of information that could prove useful later.

D.5: Uniform Data Model. To deal with heterogeneous data sources, a unified network activity monitor needs a flexible data model that can represent a variety of information types and is amenable to propagation over the network. In this context, asynchronous *event-based* communication has received particular attention in the distributed systems community. In this model, communicating entities *publish* the availability of information they can provide, and *subscribe* to information published by peers according to local interest [4].

When abstracting network activity, an event model provides an excellent match. Example events would include "packet observed", "TCP connection started", and "URL U fetched from web server S by client C". A key benefit is that events—which already represent an abstraction of activity—can be further elevated to successively higher semantic levels by correlation and aggregation. For instance, a number of unsuccessful connection attempts (already abstracted from the actual observed packets) may be rolled into a summary along the lines of "N attempts in the last T seconds," which can then escalate into an alarm if meeting some threshold. In terms of our data model, such correlation and aggregation of activity is vital for reducing the volume of overall information, per the discussion of data aggregation in terms of guideline *D.4* above.

D.6: Policy Neutral Data. It is desirable that events represent policy-neutral activity so that event subscribers are not restricted in their analysis by receiving an event stream that contains pre-conceived judgments about activity. This decoupling of the data and the analysis allows for a rich set of data crunching that may not have been envisioned when the data was collected. For instance, flow summaries can provide a wealth of insights about network behavior (e.g., for detecting scanning [6]), even though they might come from a tool like NetFlow with an original purpose of simply observing general network activity.

This guideline could limit the integration of existing NIDS systems. For example, the popular open-source Snort NIDS [11] only reports *alerts*: each time a signature matches, the system raises an alert. But by definition any set of signatures implies policy. While we could convert signature alerts into a stream of events carrying information about the match, the utility of such an instrumentation is limited because of the degree to which it presupposes what activity is of interest.

D.7: Low Overhead. Moving data to (and possibly from) the network activity monitor should not be a burdensome activity. Clearly, the torrent of activity in even modest networks is high and therefore fine-grained monitoring may not be feasible. However, the system design should be such that the overhead of collecting a unified view is as low as possible. The publish/subscribe model sketched in guideline *D.5* usefully aids this concern by only transmitting data the receiver has interest in. In addition, using events to represent network activity helps reduce the volume by naturally abstracting network activity.

D.8: Data Sanitization. The ability to sanitize sensitive information in the data should be designed into any network activity system. Clearly such features are needed if the data is to be exported to another organization (see § 3.2). However, even internal to a given organization sanitization has a role. Sanitization can be used to prevent operators from accidentally tripping across information they did not need to possess.[1] In addition, policy may deem some information too sensitive to store in the infrastructure. Such policy will often revolve around an organization's risk profile. For example, short term data retention may have high benefit in understanding intrusions, but long term data retention may entail risks through data leakage (including through legal liability or vulnerability to subpoenas). Once the risk outweighs the benefit, data will need to be sanitized.

Two different forms of sanitation may be necessary: local and global. Local sanitation requires policies that remove individual fields or events from the archive. Global sanitation is a trickier issue, as individual data may not be sensitive but the combination of elements creates a sensitivity issue; thus, it may be necessary to sanitize the results of analyses and not just the data.

We also note that the aggregation notions sketched above provide a way for clearing sensitive information: the correlation logic can alter, or completely remove, sensitive information; and aggregation limits the degree to which a remote observer can trace back final results to the lower-level activities they reflect.

2.3 Capabilities

Finally, we sketch key capabilities that a system for developing a comprehensive view of network activity will possess.

C.1: Common Analysis Procedure. A system for gathering a comprehensive view of network activity should allow for a common procedure to be used for all analysis. In other words, the same analysis that retrospectively crunches the information store should also be able to be installed as prospective queries over future information the system collects. Coupled with the unified data model sketched in $D.5$, such a capability allows operators to code an analysis one time, rather than the cumbersome current method of developing ad-hoc scripts to analyze past activity (for each data source) and then turn-

[1]This can be useful even if it is only a matter of presentation and operators can go back to the non-sanitized data, as needed.

ing around and converting those to work within ongoing monitoring systems such as a NIDS or a firewall.

Further, having a common framework for creating analysis also aids cross-organizational sharing. Even if organizations do not share data, operators from different organizations would be able to readily share code that describes what they have observed on their networks in concrete terms. This contrasts with the ad-hoc, natural language descriptions currently used across organizations, which are often inherently imprecise.

C.2: What-If Analysis. The temporal dimension of network activity analysis should support automated exploration of how analyses intended for future activity would have behaved if applied in the past. For example, understanding that an analysis procedure will trigger numerous false-positives can be vital to understand before installing that analysis for future events. Another form of what-if analysis might be to explore the implications of different possible values for missing data.

3 Extensions

In this section we discuss two additional capabilities that could be enabled by a unified monitoring infrastructure following the above design principles.

3.1 Troubleshooting

Troubleshooting network problems shares many of the same basic demands as coping with attacks. The layered, modular structure of many network and protocol designs can make it particularly difficult to pin-point the sources of errors when the system fails to operate correctly. Thus, when operators chase down problems they can find it highly beneficial to draw upon as much logged information from as many angles as possible.

While the need to understand past activity to find the source of problems is clear, an infrastructure that also allows operators to easily set up analogous analysis and/or monitoring for future activity can offer great additional power. For example, suppose an operator finds that DHCP leases were failing because of clock drift on one of the DHCP servers. If, in addition to correcting the clock to fix the immediate problem, they can also readily *codify* the corresponding troubleshooting process, then they can add it to a repertory of analysis procedures that they can draw upon in the future to troubleshoot similar instances of the problem. (They might even use the codification to install systematic monitoring of the clocks on each of their DHCP servers, to obtain automated warning of the problem's recurrence.)

Another example is that even conceptually simple services like "a web page" are today actually comprised of numerous components—both on the client and server side—that must interact. A web page may be served from a farm of servers that includes a load balancer at the front to manage the workload assigned to each individual web server. Furthermore, the web servers in turn may require information contained in one or more backend database servers. While understanding each of these components is perhaps straightforward, obtaining an over-arching view of how the full system is functioning and troubleshooting the system when it is not working properly becomes increasingly difficult as the scope and complexity of the aggregate system increases.

3.2 Cross-Organization Data Sharing

Finally, we examine the issues that arise when extending a network activity monitor beyond a single organization, as scoped in guideline $B.1$ above. Even when expanding the scope of a particular system, $B.1$ continues to serve as the *base* scope for the system. We frame enabling cross-organizational sharing as building on these local systems.

The operational reality of current cross-institutional information sharing and response is primitive. From our experiences working with security officers at a number of sites (Lawrence Berkeley National Laboratory, National Energy Research Scientific Computing Center, UC Berkeley, TU Munich), site operators set up ad-hoc information exchanges via email, online messaging, or the telephone. This state of affairs, while wrought with problems in terms of overhead and response times, also has some distinct advantages: operators (i) are explicitly involved in the decision of from whom to accept information and how much trust to place in that information, (ii) decide with whom to share information, (iii) retain the ability to apply local policies as appropriate for their organization, and (iv) need not rely on "black boxes" they may not fully understand.

A network activity monitoring system that follows the guidelines given in § 2 will be well-positioned to bridge the gap between ad-hoc telephone calls and fully-automated "black boxes". At a minimum, by providing a single repository for local retrospective analysis the system we outline above makes it easier for analysts and operators to find needed information when (say) talking to a colleague from another organization on the phone. An obvious next step is enabling organizations to easily swap analysis code. This turns key parts of the exchange between operators into formal definitions rather than informal descriptions.

More speculatively, operators could run queries against the data archive of other organizations. We envision such cross-organizational data sharing as tenable between sites that have a high degree of trust in one another to act in a responsible manner; we do not envision a system for global sharing of information. Even in such a restricted setting, an operator would likely still have to manually green-light a query across their past activity, and also for the results being returned to the peer. As with recording of local information, we see such a semi-automated approach as working from a "dirty slate", to *evolve* the current state of affairs towards a place where advanced, efficient, and coordinated sharing is reality—even if the scale is initially more modest than we might like. An organization's own unified activity monitor will be well-positioned to deal with requests from outside organizations and to interact with operators, as necessary. In addition, by following the guidelines given in § 2 the system will include hooks for aggregating and sanitizing data before transmitting it to another organization.

In general, we identify the following high-level guidelines as desirable for any approach that allows operators to share information with peers with whom they have a reasonable amount of trust:

S.1: The approach does not presume any particular analysis policies, nor require participants to agree on common policies (as similarly developed in $D.6$ above).

S.2: The scheme provides each site with control over the information that it imports and exports. Specifically, each site independently *(i)* selects information to provide to peers, *(ii)* scrubs sensitive content from information it shares, and *(iii)* chooses which externally received information to incorporate into its local analysis. In addition, different policies can be applied to each peer.

S.3: The approach *augments* the local security apparatus. Hosts, NIDS, service daemons, and the like do not require knowledge of the site's sharing policies to carry out their functions.

S.4: The setup should be robust against failures (including those induced by attacks). When a failure occurs, the site's existing security components must continue working as they would have had there been no additional sharing infrastructure in place.

S.5: The approach calls for sharing only moderate volumes of data. The goal should be to exchange semantic abstractions of activity (expressed by queries over event logs); therefore, we favor robustness over efficiency.

We believe these guidelines—in conjunction with organization-level systems that are consistent with the guidelines developed in the previous section—hold the possibility of facilitating significantly greater sharing and easier communications among the operator community.

4 Summary

In this paper we lay the systematic design groundwork for systems that provide unified network monitoring. We believe that following the guidelines sketched in this paper will yield systems that provide enough flexibility to be highly useful in a wide range of situations. While some of the guidelines might seem mundane, and even perhaps low on vision to the research community, it is our position that starting from the current operational reality is the best way to make progress on these sorts of information collection systems. The principles we outline are fundamentally rooted in this "dirty slate" approach. We would not, however, preclude additional layers and extensions from being built on top of systems that follow the principles outlined in this paper (e.g., to do automated distributed intrusion detection). In some sense, gaining a unified view of network activity is not only an operational win, but will likely prove to be a research win in terms of both increasing researchers' understanding and providing a platform that will ultimately enable new research.

5 Acknowledgments

This work was supported by the National Science Foundation under grants CNS-0716640, ITR/ANI-0205519, and NSF-0722035, for which we are grateful. The opinions expressed in this work are solely those of the authors and should not necessarily be considered to be the opinions of the funding agency.

References

[1] BLACKLIST, D. P. http://www.dnsbl.org.

[2] CENTER, C. C. http://www.cert.org.

[3] Distributed Intrusion Detection System *DShield.org*. http://www.dshield.org.

[4] EUGSTER, P., FELBER, P., GUERRAOUI, R., AND KERMARREC, A. The Many Faces of Publish/Subscribe. *ACM Computing Surveys 35*, 2 (2003), 114–131.

[5] FOUNDATION, T. S. http://shadowserver.org.

[6] JUNG, J., PAXSON, V., BERGER, A. W., AND BALAKRISHNAN, H. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy* (2004).

[7] MAIER, G., SOMMER, R., DREGER, H., FELDMANN, A., PAXSON, V., AND SCHNEIDER, F. Enriching Network Security Analysis with Time Travel. In *ACM SIGCOMM* (2008).

[8] PAXSON, V., AND FLOYD, S. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking 3*, 3 (1995), 226–224.

[9] PHISHTANK. http://www.phishtank.com.

[10] PROJECT, T. S. http://www.spamhaus.org.

[11] ROESCH, M. Snort: Lightweight Intrusion Detection for Networks. In *Proc. Systems Administration Conference* (1999).

[12] SECURITYFOCUS. Bugtraq mailing list. http://www.securityfocus.com/archive/1.