Network Access Log Visualization & Sonification

Benjamin deButts
ben@debutts.com
Mentor: Lane Harrison

Abstract

As the worldwide web gets larger and larger, the variety and number of network incidents swells alongside it. Security analysts need to be able to detect, analyze, and respond to incidents faster than ever. However, access logs are both extensive and exhaustive in detail, visualization, a field that bases itself around making clunky, hard-to-decipher data manageable and understandable, is a promising method of aiding the detection/response process. By visually observing traffic activity (particularly which and how many attacks are being executed), analysts will be able orient themselves in a notification stream more quickly and efficiently than with other text-based tools. Additionally, by mapping distinct notification sounds to anomalous incidents, analysts will be able to monitor live network streams without constant visual attention. Upon the recognition of an "incident" tone, they will be able to interact with the application, and quickly get up to speed with network activity.

Introduction

Network security is a bustling field and new attacks occur by the second. Analysts tasked with monitoring these networks have quite an undertaking and would be vastly helped by having the filtering between harmless and possibly harmful traffic done for them. Security analysts live extremely busy lives. When asked what are the day-to-day activities as a security analyst are, Mike Adler of *Cisco* responded: "The core functions are centered around the Intelligence cycle: collection (monitoring sources and awareness), processing (researching and prioritizing), analysis (collaborating and validating), reporting, and closing the loop by adjusting the collection to the current activity. The day-to-day activities include handling the constant flow of information, data, and events and adjusting to the changes." (Adler). The takeaway: their lives are *industriously* busy. And running in parallel alongside their day-to-day work, is a tumultuous

Internet churning out new “vulnerabilities […] everyday” (Victor). A major problem derivative of constantly new vulnerabilities is that the security mechanism (or IDS, Intrusion Detection System) has to be updated as frequently as possible to catch them—which means, the number of false positives, or non-incidents accidentally triggered, go up as well.

The SANS institute writes that “the major problem that false positives create is that they can easily drown out legitimate IDS alerts.” (Owen). Essentially, unending amounts of alerts can hide an attacker amongst the crowd and waste analysts’ time and effort on harmless activity (Victor). Furthermore, studies examining the costs of interruption explain just how detrimental interruptions are to a programmer’s work: “When resuming work, developers experience increased time to perform the task, increased errors, increased loss of knowledge, and increased failure to remember to perform critical tasks.” (Parnin). Security analysts, perhaps more than your typical programmer, experience this incessantly, as their responsibilities include monitoring an alarm system, yet also many other tasks (as enumerated above). Herein lies the problem that this program attempts to solve: distract the analyst less frequently and allow passive monitoring with a combined visualization and sonification network activity tool.

To the Community/Applications

Combining and improving on typical IDS systems by adding sonification and visualization combats two problems: false positives and work interruption. By issuing distinct sounds for distinct incidents, analysts can passively monitor networks without constant visual attention. After hearing a recognized sound, a log-visualization allows the analyst to be updated on the missed traffic, isolate and observe attacks and furthermore obtain the desired information without having to parse through the exhaustive log: their status code, time stamp, and from what

IP address the attack originated (see Design Decisions for how these traits were determined as "desired"). A fully functional application with added functionality that lets an analyst include their own desired strings for incidents, would unquestionably streamline the access log review process and free up more of an analyst's time to devote to other important security necessities.

Design Decisions

Access logs, by their very nature, are narrative stories of what requests have been made on a network. These narratives run constantly, at all hours, and include accesses both harm*less* and harm*ful*. As a tool for security experts, this paper and the tool derivative of it will focus primarily on helping analysts identify harmful traffic efficiently.

Apache logs are where analysts learn what requests are being made, as well as the details of that request. These details include source IP address, a timestamp, the specific GET or POST request, and a status code describing whether the request was good or not. This data can inform an analyst on what incident was carried out and when, in addition to from which IP address this request was made. A typical line in an access log looks like the example string below:

```
46.28.206.150 - - [25/Aug/2014:21:23:01 +0200] "GET /cgi-bin/php HTTP/1.1"
404 177 "-" "-"
```

In this example, the source IP is 46.28.206.150, the time stamp August 25th 2014 at 21:23:01, the request was for "/cgi-bin/php HTTP/1.1" and the status code was "404". Analysts surf through thousands and thousands of lines just like these and have to peg which ones are suspicious and possibly malicious in nature. Most analysts, or at least ones that work for companies that can afford one, have IDS to help them filter out the good or harmless requests. "Harmful" traffic is a relatively ambiguous term. More often than not, distinguishing "bad" traffic from "good" is the entire game that analysts are playing as they monitor networks.

However, there are unique and distinct requests and accesses that can without question be labeled as “attacks” or “incidents”. For these select accesses—and more as more types of attacks are uncovered—there will be a method of alerting and orienting the security analyst.

However, even beyond intrusion detection systems (which are often flawed in their own ways), an analyst still has to notice every last alert and not necessarily in a particularly easy manner. Visualization would present this “easier manner” and sonification alongside it with a smarter triggering system would combat these problems.

Why implement audio flags in addition to visual? The reasons are fundamentally two: first, and most obviously, audio flags can be passively monitored—much as many people listen to music while working, analysts would be passively aware that a certain sound uttered from the program means a certain request has been made. Two, and less obvious, is that audio data is processed differently than visual. Contrary to visual, humans process audio linearly through time. This means that where a visual schematic could be “read” many different ways by the viewer, (certain flags being noticed at different times), audio can only be understood one way, and furthermore, in *one* sequence (Skau). Applied to our access logs, this minimizes the delay of visually searching a file into a trivial: heard sound vs. didn’t hear sound. Upon hearing the sound, *then* arises the advantage of visualization (Figure 2) as it can quickly orient the analyst in the narrative that has been unrolling beneath the non-harmful accesses.

When tackling the question of, “how often does an analyst need to be alerted to incidents”, two central measures emerged: the severity of the incident, and its typical frequency of occurrence. These two measures fundamentally decided the priority of any one incident. The reason an alert system can be overly exhaustive in alerts is because one: the incident in question happens *often*, and two: the system alerts for incidents that wouldn’t even necessarily need a notification. Therefore, a system interested in lowering the demands on an analyst would need to address these two central reasons for information inundation.

In the scope of this project, and with the help of a security expert and professor at Tufts University, Ming Chow, eight incidents were selected. In order of severity, Professor Chow listed: shell code, requests for "phpMyAdmin", requests for "wp-admin", any time the string "admin" is spotted outside of the previous two incidents, directory traversal (i.e /etc/ requests), cross-site scripting tags ("<script>" for example), "nmap", and finally http errors as the most harmless. In later versions of this program, more incident detection could be included in the log parsing, visualization, and sonification. Ideally, the application would be synced with a pre-existing IDS or the analyst would personally write what strings a parser should look for. Depending on the application, internal network monitoring versus external, or the company in question, an analyst might want to prioritize certain requests over others.

Next, each incidents' frequency was determined by running a ruby program on Professor Chow's honeypot and printing the attack type, source IP address, time stamp, and status code

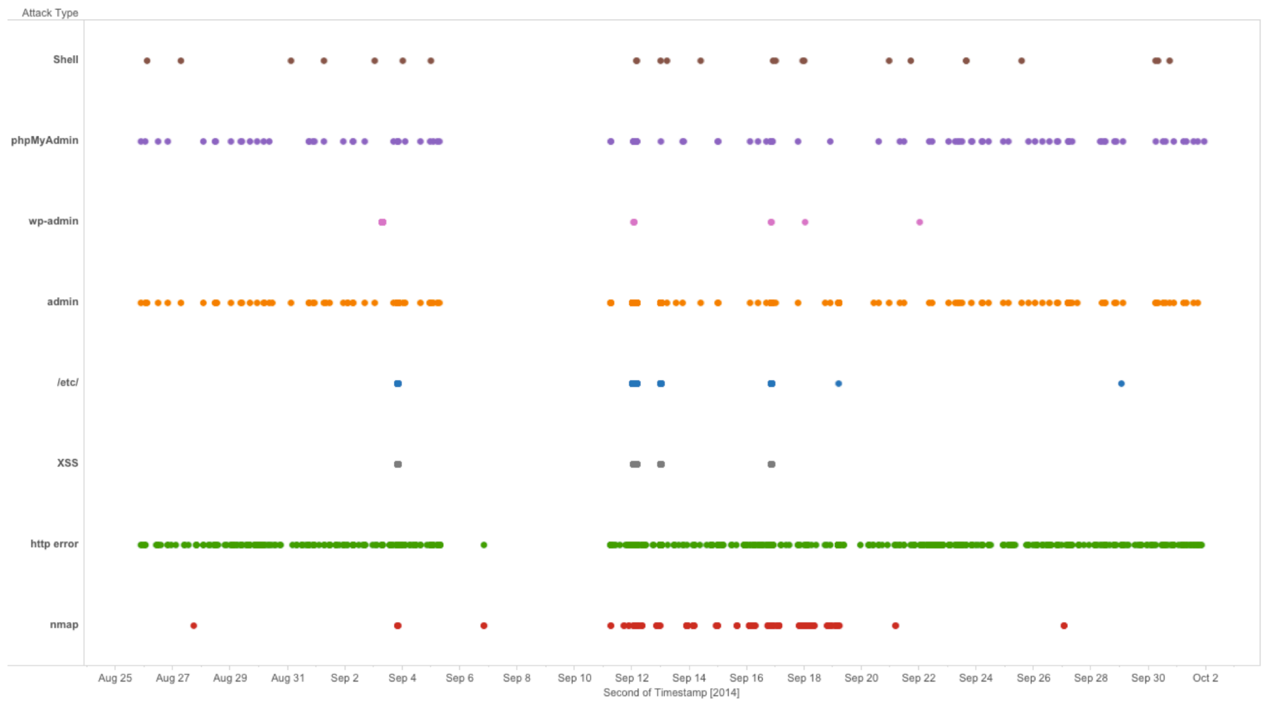


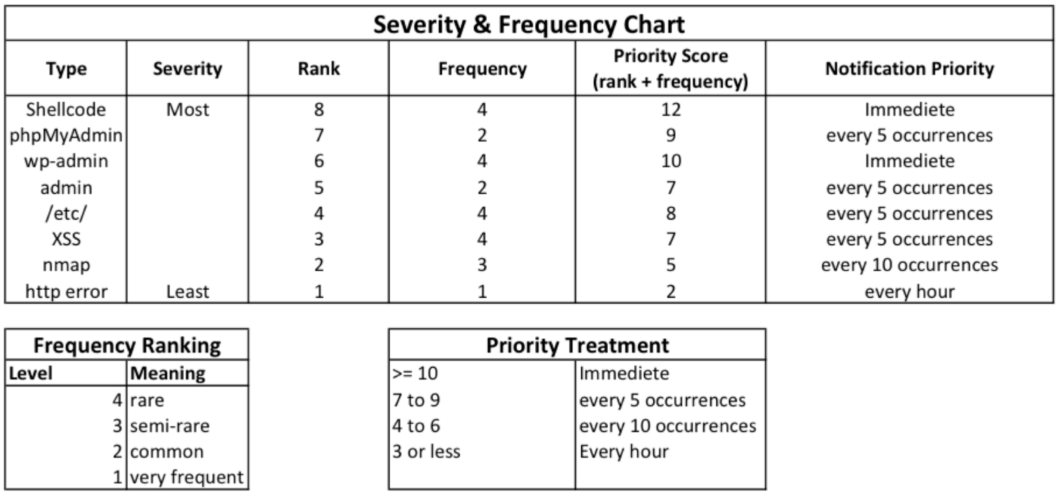
Figure 1: to determine the smarter triggering method, we analyzed the frequency of particular attacks over the course of honeypot log. We interviewed a expert to determine the severity of each of these attacks. Based on frequency and severity, we create a set of rules to support passive monitoring: Table 1.

(four attributes Prof. Chow explained a security analyst would care about). This extensive CSV file was charted on Tableau (a graphical-visual tool) (Figure 1).

As you can see, there are *many* events and many of them are happening very often. Particularly the green line-almost on the bottom of the chart are http errors which occur almost constantly. However, other incidents, like shell code notifications for example, are more seldom seen. If these incidents were each to intonate their own tone *every* time they occurred, what would result is a constant (and altogether unproductive) cacophony of sounds. The proposed solution to this is to issue a formula for dictating how often and with what urgency each incident should trigger a sound.

"Severity ranking" was informed by Professor Chow, and "Frequency ranking" was informed by the above Tableau graphic—events that appeared to happen very often, were given lower scores as they would contribute to an overwhelming sonification of an access log. These two attributes, frequency and severity were then summed to a "Priority Score", from which they were mapped to a notification priority for the application. Thus, "http errors", which seemed to operate as more of a flag for incoming activity and are less harmful in of themselves, are only chimed every hour if they do occur.

Table 1: Using inferred frequencies and severities, we produced a rule-based notification system for passive monitoring.



Severity & Frequency Chart

Type	Severity	Rank	Frequency	Priority Score (rank + frequency)	Notification Priority
Shellcode	Most	8	4	12	Immediete
phpMyAdmin		7	2	9	every 5 occurrences
wp-admin		6	4	10	Immediete
admin		5	2	7	every 5 occurrences
/etc/		4	4	8	every 5 occurrences
XSS		3	4	7	every 5 occurrences
nmap		2	3	5	every 10 occurrences
http error	Least	1	1	2	every hour

Frequency Ranking

Level	Meaning
4	rare
3	semi-rare
2	common
1	very frequent

Priority Treatment

>= 10	Immediete
7 to 9	every 5 occurrences
4 to 6	every 10 occurrences
3 or less	Every hour

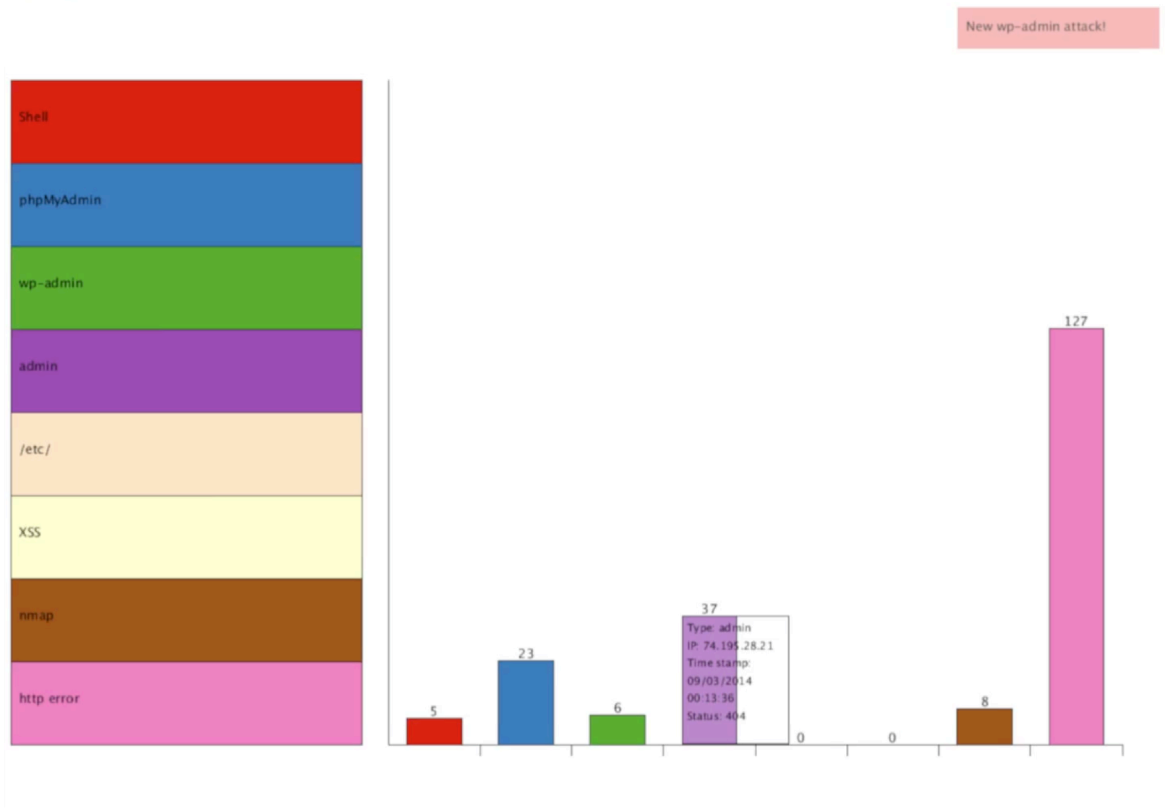


Figure 2: the resulting visualization, displaying active/inactive attack types on the left, and accumulating incidents as a bar chart seen on the right. Hovering over a bar displays attack type, IP address, time stamp, and attack status code of the most recent incident.

In The Future

Future iterations of this application would absolutely expand to include more attack types. Moreover, a functionality would be added to let the analyst include their *own* strings to look for and trigger upon seeing in the access log. This additional string would be ranked on frequency and severity by the analyst, therefore giving the incident trigger attributes to act on in the visualization and sonification. Additionally, this application could be synced with pre-existing intrusion detection systems to make their outputs, alert system, and triggering decisions more efficient and less burdensome on the analysts.

Conclusion

This program aims to expedite and facilitate the network monitoring process by applying visualization and auditory techniques. In this first implementation, visualization serves to place the analyst in the contextual narrative of what requests have been made on the network. Alongside it, the sonification of the program capitalizes on our natural ability to filter out and notice specific sounds out of the many, ultimately allowing passive monitoring of network traffic. Smarter triggering methods, according to the severity-frequency formula, pre-filters incidents so that analysts can ignore the more harmless incidents and be alerted to the more suspicious. Moreover, this triggering method also works against the high false positivity rate endemic to security work and lessens the number of distractions analysts as programmers experience on their day-to-day jobs.

Works Cited

Adler, M. (2012, April 16). What is it Like to be a Cisco Security Analyst? Retrieved December 8, 2014, from http://blogs.cisco.com/security/what-is-it-like-to-be-a-cisco-security-analyst

Chow, M. [Personal interview]. (n.d.).

Hadhazy, A. (2014, April 30). Heavenly Sounds: Hearing Astronomical Data Can Lead to Scientific Insights. *Scientific American*.

Humphries, C., Prigent, N., Bidan, C., & Majorczyk, F. (2013, October). ELVIS: Extensible Log VISualization. In *Proceedings of the Tenth Workshop on Visualization for Cyber Security* (pp. 9-16). ACM.

Owen, D. (n.d.). What is a false positive and why are false positives a problem? Retrieved December 10, 2014, from http://www.sans.org/security-resources/idfaq/false_positive.php

Parnin, C., & DeLine, R. (2010, April). Evaluating cues for resuming interrupted programming tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 93-102). ACM.

Phan, D., Paepcke, A., & Winograd, T. (2007, May). Progressive multiples for communication-minded visualization. In *Proceedings of Graphics Interface 2007*(pp. 225-232). ACM.

Skau, D. (2014, October 14). How Audio Can Help Communicate Time Data. Retrieved October 27, 2014.

Victor, G. (n.d.). False Positives in Intrusion Detection Systems. Retrieved December 10, 2014, from http://www.academia.edu/1431396/False_Positives_in_Intrusion_Detection_Systems