DEMO: Akatosh: Automated Cyber Incident Verification and Impact Analysis

Jared M. Smith
Oak Ridge National Laboratory
University of Tennessee, Knoxville
smithjm@ornl.gov

Elliot Greenlee
Oak Ridge National Laboratory
University of Tennessee, Knoxville
greenleeed@ornl.gov

Aaron Ferber
Oak Ridge National Laboratory
ferberae@ornl.gov

ABSTRACT

Akatosh, a U.S. Department of Homeland Security Transition to Practice Program (TTP) project developed by Oak Ridge National Laboratory with industry and academic partnership, enables automated, real-time forensic analysis of endpoints after malware-attacks and other cyber security incidents by automatically maintaining detailed snapshots of host-level activity on endpoints over time. It achieves this by integrating intrusion detection systems (IDS) with forensic tools. The combination allows Akatosh to collect vast amounts of endpoint data and assists in verifying, tracking, and analyzing endpoints in real time. This provides operations personnel and analysts as well as managers and executives with continuous feedback on the impact of malicious software and other security incidents on endpoints in their network.

CCS CONCEPTS

•Security and privacy →Malware and its mitigation; Intrusion detection systems; Operating systems security;

KEYWORDS

Incident Response; Forensic Analysis; Endpoint Security; Breach Remediation

1 INTRODUCTION

While Intrusion Detection Systems (IDS) can help prevent attacks on a system, they also incur a higher than desired false positive rate. When a cyber attack happens to break through an IDS or other defensive system, the effort of performing an analysis of the affected systems and the recovery from any potential infections is costly and time-consuming. Developed at Oak Ridge National Laboratory (ORNL), Akatosh is a highly configurable system based on the integration of the capabilities of one or more IDSfis and automated configuration and system verification.

With this integration, it is possible to analyze systems in near real-time and provide operations and forensic analyst personnel with continuous feedback on the impact of software, malware, and active users on deployed systems. By providing an interface between any number of IDSs and the Akatosh client, Akatosh is able to intelligently fisnapshotfi affected systems based on cues from the IDS. Before incidents, Akatosh takes regularly scheduled snapshots to account for states of the system over time. With these snapshots, it can automatically provide a succinct report on the Akatosh server by differentiating these previous known states and post-infection states based on the timing and metadata associated with IDS notifications to the client interface. With the differentiated states from any point in the history of the machine, Akatosh helps point out whether a true infection occurred, and if so, what was impacted, thus lowering the false positive rate of modern IDS products. With this data, it is also possible to analyze trends over time in attacks and come closer to conclusions on why a system was attacked.

Akatosh is a U.S. Department of Homeland Security (DHS) Transition to Practice program (TTP) project, and is one of eight technologies in the DHS TTP 2017 class being developed and led by industry and academic institutions such as MIT Lincoln Laboratory, MITRE, Worcester Polytechnic University, and Pacific Northwest National Laboratory. Through the DHS TTP program, we are working with industry partners to pilot the this technology, demonstrate it's capabilities at industry "demo days" in major international hubs, and ultimately license the research technology to partners with the goal of being integrated into production systems serving real users.

2 BACKGROUND

In practice, forensic analysts and other operations personnel face two distinct and important problems. In the realm of computer security defense mechanisms, IDSs consume information like network packets, endpoint statistics, and other metrics that the IDS uses to pick out anomalous behavior, which potentially represent cyber attacks. Unfortunately, IDSs have high false alert rates and the sheer number of alerts over time can overwhelm security operations personnel, which makes correctly identifying actual attacks difficult. Another problem faced by enterprises can be seen in a 2016 study by IBM and the Ponemon Institute [11], which found that among 383 companies, the cost of incident response and mitigation for a successful cyber attack accounted for 4 million USD on average per incident. over a quarter of the total cost was due to forensic activities associated with the breach. This cost largely comes from having to verify endpoint state and conduct forensic analysis after alerts from endpoints indicate that they were potentially impacted by an attack or related security incident.

3 SYSTEM DESIGN

System Architecture Akatosh starts by reducing the impact of false positives and the cost of incident response by enabling automated, real-time forensic analysis of endpoints when prompted

CCS'17, October 30–November 3, 2017, Dallas, TX, USA
 ISBN 978-1-4503-4946-8/17/10.
DOI: http://dx.doi.org/10.1145/3133956.3138854

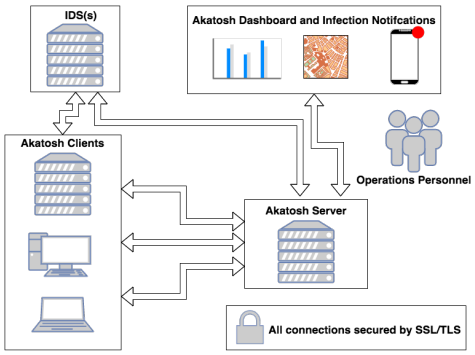

Figure 1: High-level architecture diagram for Akatosh.

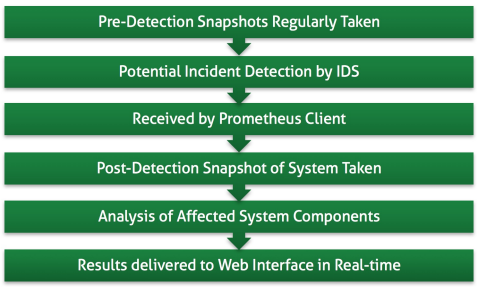

Figure 2: Step-by-step explanation of the Akatosh endpoint snapshot and real-time analysis process.

by IDS alerts. By doing these this, Akatosh helps operations personnel verify that an alert on an endpoint corresponds to a true cyber-attack. The system is comprised of small Akatosh client or agent, the Akatosh server, and the Akatosh dashboard, as depicted in Figure 1. The Akatosh clients live on network endpoints and take regularly scheduled baseline snapshots on configurable time intervals to record endpoint state over time.

These snapshots capture specific data about the endpoint, including processes, loaded drivers, registry entries, network connections, and other data. When an IDS detects anomalous behavior it alerts the Akatosh system. Depending on the nature of the alert (configured by the operators), the Akatosh client immediately takes a snapshot of the endpoint that generated the alert and sends the snapshot to the Akatosh server. The Akatosh server automatically produces a succinct incident report differentiating the post-alert snapshot from the most recent baseline snapshot. The Akatosh dashboard displays all endpoints being tracked, their status, the snapshot data being collected as the system receives IDS alerts, and the incident reports.

Figure 2 summarizes the underlying process described above. Akatosh automatically analyses the differences between pre-alert and post-alert snapshots in real-time and displays the results on the dashboard, showing the specific endpoint components affected by the anomalous behavior.

Time-Series State Differentiation Akatosh analyzes the parallel historical timeline of memory images for each client machine in order to provide insight into machine state differences. The majority of these images will reflect daily activity without a malicious presence, but this timeline also captures the critical period following an IDS alert. Consider the memory image taken before and just after such an alert; the apparent differences shed light on new files, processes, and a multitude of other system changes caused by possible malicious activity. In order to analyze the images, Akatosh integrates with existing memory forensics tools Volatility and Rekall [6, 8]. These two frameworks combine individual plugins for extracting various well-known operating system state data to provide a comprehensive view of machine memory. An example of a plugin would be the processes running on the machine, or the active and recently ended network connections. Akatosh then performs state differentiation across each plugin in order to pinpoint differences across images. By displaying these differences to forensic analysts, our system provides detailed context and a frame of reference which intends to speed up the process of recovery after an attack.

Classification of State Differences Using state differentiated images before and after a period of time, a classification can be made between detecting or not detecting malicious activity. Rather than examining the images themselves or other behavioral information, our system analyzes two kinds of state transitions: clean to clean and clean to infected. As forensic experts investigate alerts using the Akatosh system, a determination is made between real and false. This expert knowledge can be captured through classification algorithms. In order to produce a sample data set equivalent to this real knowledge, we capture machine state while infecting systems with malware. This process is performed programatically using Cuckoo Sandbox [3], an open source malware analysis system, to inject malware from online repositories and ORNL resources into virtual environments running software to approximate human behaviors like opening and closing software, navigating the internet, and sending emails [19]. This programmatic collection of machine images before and after software and malware has run on a system gives us a working dataset of images with which to feed through Akatosh.

Using this dataset we can perform feature extraction and a classification method survey. Hand-coded feature extraction per plugin is possible but knowledge and time intensive. This approach allows for pre-analysis using linear discriminant analysis to determine the plugins that contribute most to overall classification accuracy, guiding initial ordering of plugin results on the Akatosh system [12]. More generally, standard natural language pre-processing tools like bag of words and n-gram extraction can be co-opted to fit this type of document [20]. Our method survey covers standards in document classification such as naive bayes, expectation maximization, support vector machines, and decision trees [1] [18] [15]. The results of these are combined using classifier fusion to produce a binary recommendation with a specific confidence [16]. After the model crosses a pre-determined confidence threshold, Akatosh begins to make recommendations to analysts in two areas. Overall, the algorithm recommends high-confidence real alert predictions higher in the queue of new IDS alerts. For individual clients, the algorithm presents high-confidence real alert plugin results higher so that analysts can more quickly check these indicators. Through

these recommendations, the overall time to recovery of an impacted machine is reduced.

Implementation Akatosh is implemented in Python [5], which allows the system to run on Windows, Linux, and Mac-based OSs. As stated earlier, Akatosh utilizes Volatility and Rekall [6, 8] for extracting machine state data from images. To capture images from machines, Akatosh uses the Rekall Memory Forensic Suite of imaging tools, which are used in other frameworks such as Google's GRR [7]. To store data on the server, Akatosh uses a combination of the battle-tested relational database, PostgreSQL, and a static file storage system, Minio, based on Amazon S3 [9, 13].

4 EVALUATION

Can We Scalably Collect Full State Captures from Hundreds of Endpoints? Akatosh can collect memory images of nearly any practical size (tested up to 64 GB) in less than 30 seconds for 16 GB images. Akatosh stores no data on client machines, as it transfers the images as they are captured. In the varying network conditions tested (between 10-1000 Mbps upload speeds), the transfer speed is bounded only by the speed at which memory can be captured. When images are stored on the server, images are encrypted with a 2048-bit key using the AES algorithm. Additionally, up to 60% on average of the original memory size can be compressed due to the nature of the image format, thus reducing 16 GB image captures to less than 8 GB when stored in Minio.

With respect to client performance overhead incurred due to imaging, no noticeable slowdowns can be seen from the client, and in our testing we saw no more than 10 to 30% CPU usage to image a machine. Finally, Akatosh currently scales up to 150+ machines and can load balance image capturing and state differentiation analysis effectively for a variety of clients, including Windows, Linux, and Mac.

Can We Surface Deeper Context to Existing Alerts with State Differences? The work described in the prior section on classification of memory diffs is still under active development; however, Akatosh has been tested against several pieces of historically significant malware, and has identified all of the components the tested malware was known to affect on client machines. The components known to be affected were pulled from a variety of published write-ups on the malware. The malware tested consisted of the DarkComet Trojan, the NJRat Trojan and Reverse Shell, and Stuxnet.

These results indicate that when Akatosh is alerted to an infection on a system where malware has infected the host in question, our system can identify the affected components and bring them to the attention of the incident response personnel. Future work remains to be done to test Akatosh against vast amounts of other malware and software, though early results are promising.

5 RELATED WORK

Akatosh is the first of its kind system to integrate automated forensic analysis with IDSs. Through this integration, Akatosh can perform a detailed analysis of the affected endpoints at the exact time of the incident, unlike current incident response systems, which are less reactive to immediate changes in endpoint state, at least at the level of detail that Akatosh provides.

Additionally, the Akatosh dashboard automatically provides reports showing a high-level overview of affected endpoint components that operations personnel and analysts as well as managers and upper-level executives can understand and dig into. Reports are generated in real-time without shutting down endpoints to perform the tedious task of imaging the machine and analyzing the image on a separate machine. Similar products in the space do not provide differentiated endpoints states to operations personnel [2, 10], and may also require manual analysis of endpoints causing analysts to shut down machines before examining their state [4].

While products exist to perform endpoint history analysis for non-security related domains, such as infrastructure monitoring [14, 17], these products do not transition well to verifying, tracking, and analyzing the impact of cyber attacks. By focusing on affected endpoint components, Akatosh assists in verifying incidents and automatically tracking and analyzing propagation over the components.

6 CONCLUSION AND FUTURE WORK

In this work we have presented a novel system developed to enhance context around existing alerts in modern security defense systems, while allowing the scalability to potentially thousands of machines and reducing the cost of mitigating breaches when they inevitably occur. In the coming months, Akatosh will be undergoing pilots at the U.S. Department of Energy HQ and MITRE, as well as undergoing active development to finish the full classification abilities of the system as well as continue to scale out to additional clients.

REFERENCES

[1] Sonal Salve Swati Vamney Bhawna Nigam, Poorvi Ahirwal. 2011. Document Classification Using Expectation Maximization with Semi Supervised Learning. (2011). https://arxiv.org/abs/1112.2028
[2] CarbonBlack. 1999. (1999). http://www.carbonblack.com
[3] Cuckoo. 2017. Cuckoo. (2017). https://cuckoosandbox.org/
[4] EndCase. 2017. EndCase. (2017). https://www.guidancesoftware.com/encase-forensic
[5] Python Software Foundation. 2017. Python. (2017). https://www.python.org/
[6] Volatility Foundation. 2017. Volatility. (2017). http://www.volatilityfoundation.org/
[7] Google. 2017. GRR. (2017). https://github.com/google/grr
[8] Google. 2017. Rekall. (2017). http://www.rekall-forensic.com/
[9] The PostgreSQL Global Development Group. 2017. PostgreSQL. (2017). https://www.postgresql.org/
[10] Tanium Inc. 1999. Endpoint Security and Systems. (1999). http://www.tanium.com
[11] Ponemon Institute and IBM. 2017. Cost of Data Breach Study. (2017). https://www.ibm.com/security/data-breach/
[12] et al. Mika, Sebastian. 1999. Fisher discriminant analysis with kernels. (1999). http://ieeexplore.ieee.org/abstract/document/788121/
[13] Minio. 2017. Minio. (2017). https://minio.io/
[14] PrometheusIO. 2017. PrometheusIO. (2017). https://prometheus.io/
[15] J.R. Quinlan. 1993. C4.5: Programs for Machine Learning. (1993). http://dl.acm.org/citation.cfm?id=152181
[16] D. Ruta and B. Gabrys. 2000. An Overview of Classifier Fusion Methods. (2000). http://eprints.bournemouth.ac.uk/9649/
[17] Splunk. 2017. Splunk. (2017). https://www.splunk.com/
[18] J.A.K. Suykens and J. Vandewalle. 1999. Least Squares Support Vector Machine Classifiers. (1999). https://link.springer.com/article/10.1023%2FA%3A1018628609742?LI=true
[19] TheZoo. 2017. TheZoo. (2017). https://github.com/ytisf/theZoo
[20] Hanna M. Wallach. 2006. Topic modeling: beyond bag-of-words. (2006). http://dl.acm.org/citation.cfm?id=1143967