CANVuS: Context-Aware Network Vulnerability Scanning

Yunjing Xu, Michael Bailey, Eric Vander Weele, and Farnam Jahanian

Computer Science and Engineering, University of Michigan
2260 Hayward St., Ann Arbor, Michigan 48109, USA
{yunjing, mibailey, ericvw, farnam}@eecs.umich.edu

Abstract. Enterprise networks face a variety of threats including worms, viruses, and DDoS attacks. Development of effective defenses against these threats requires accurate inventories of network devices and the services they are running. Traditional vulnerability scanning systems meet these requirements by periodically probing target networks to discover hosts and the services they are running. This polling-based model of vulnerability scanning suffers from two problems that limit its effectiveness—wasted network resources and detection latency that leads to stale data. We argue that these limitations stem primarily from the use of time as the scanning decision variable. To mitigate these problems, we instead advocate for an event-driven approach that decides when to scan based on changes in the *network context*—an instantaneous view of the host and network state. In this paper, we propose an architecture for building network context for enterprise security applications by using existing passive data sources and common network formats. Using this architecture, we built CANVuS, a context-aware network vulnerability scanning system that triggers scanning operations based on changes indicated by network activities. Experimental results show that this approach outperforms the existing models in timeliness and consumes much fewer network resources.

1 Introduction

Users in modern enterprise networks are assailed with spyware that snoops on their confidential information, spam that floods their e-mail accounts, and phishing scams that steal their identities. Network operators, whose goal is to protect these users and the enterprise's resources, make use of intrusion detection/prevention systems [22, 23], firewalls, and antivirus software to defend against these attacks. To be effective, the deployment and configuration of these systems require accurate information about the devices in the network and the services they are running. While both passive network-based and host-based methods for building these inventories exist, the most prevalent method of assessment continues to be active network-based vulnerability scanning. In this model, a small number of scanners enumerate the potential hosts in a network by applying a variety of tests to determine what applications and versions are being run and whether these services are vulnerable. For very large networks, scanning can take a significant amount of time (e.g., several weeks) and consume a large amount of network resources (e.g., Mbps). As a result, network operators frequently choose to run these scans only periodically.

Unfortunately, the dynamics of a hosts' mobility, availability and service configurations exacerbate the problem of when vulnerability scanning should take place. We define the knowledge of these changes as the context of the network. A context insensitive model for vulnerability scanning suffers from wasted resources (e.g., time, bandwidth, etc.) and the observation of stale data. For example, often the network operators who are responsible for protecting the network do not have full control over the placement and availability of hosts in the network. Addresses may be allocated to departments within the organization who use the addresses in different ways, leaving the network operators with little insight into what addresses are allocated or unallocated. Furthermore, these departments themselves often have little control over how their users make use of these resources and even known, allocated IP addresses and hosts may exhibit availability patterns that are difficult to predict. As a result, network operators spend resources and time scanning IP addresses that have not been allocated or for hosts that are unavailable.

In addition, network operators have limited visibility into what services are being run on these hosts because they are typically managed by different administrators. Without the knowledge about the context, the accuracy of detecting these services and their configurations is bound by the frequency of scanning. As a result, any change that occurs since the last scan will obviously not be visible until the next scanning iteration. The rapid occurrence of new, active exploits, announced vulnerabilities, and available software patches, along with the dynamic nature of how users utilize the network, suggest that even small drifts in these inventories may result in a large security impact for the organization. Furthermore, the assumption that services remain relatively static over a short period of time is increasingly flawed. The emergence of peer-to-peer, voice-over-ip, messaging, and entertainment applications have led to a large number of dynamic services on these hosts. Periodically scanning, by its very nature, only captures a snapshot of those services that are active at an instant in time and it may miss many other important services.

To solve these problems, we introduce a context-aware architecture that provides a uniform view of network states and their changes. The architecture makes use of existing sources of host behavior across a wide variety of network levels including the link, network, transport, and application layers. Diverse data formats such as syslog, SNMP, and Netflow representing activities at these layers are used to generate abstract views that represent important network activities (e.g., a host connecting to the network, a new subnet allocated, a new binary in use). Instead of scanning all the hosts in the network at the same frequency, periodic scanning in our architecture selectively scans hosts based on their availability patterns. Moreover, these abstract views are used to create events about host configuration changes (e.g., users connecting to a new service, downloads from update sites, and reboots) to trigger active scanning. Thus, this approach is inherently interrupt driven and this event-based model, on top of the context-aware architecture, provides more timely and accurate results. In contrast, scanning periodically at a higher frequency would be the alternative, but would require substantially more resources.

To demonstrate the effectiveness of this architecture, a prototype system is constructed and deployed in a large academic network consisting of several thousand active hosts distributed across a /16 and a /17 network. Evaluation of this architecture over a 16-day period in March of 2010 illustrates that CANVuS outperforms existing techniques in detection latency and accuracy with a much fewer number of scans. The experimental results also reveal several problems of the current methodology including the lack of ground truth and the limited event types, both of which will be addressed in future work.

The rest of this paper is organized as follows: § 2 discusses research papers and commercial products that relate to enterprise network security, especially vulnerability assessment, and how our system differs from existing solutions. § 3 discusses our year-long evaluation of the university's scanning activities that lead to our current research. § 4 has an in-depth description of our context-aware architecture. Details of the CANVuS system implementation on this architecture is presented in § 5. § 6 describes the evaluation of CANVuS and the context-aware architecture. § 7 discusses the risks involved in this project and our mitigation efforts. The limitations and future work are explored in § 8. Finally, § 9 concludes the paper.

2 Related Work

A variety of security software solutions and appliances have been proposed to defend against the threats faced by enterprise networks. These fall roughly into those focused on real-time, reactive detection and prevention and those based on proactive risk identification and policy enforcement. Network-based, real-time detection and prevention solutions, such as intrusion detection systems [22, 23] are deployed at natural aggregation points in the network to detect or stop attacks buried in network packets by applying known signatures for malicious traffic, or by identifying abnormal network behaviors. Host-based antivirus software [18, 32] is meant to protect hosts from being infected by malicious programs before their binaries are executed and, like network-based approaches, may do so either through static signatures or anomaly detection.

In contrast, proactive approaches to network security seek to reason about risks before an attack event happens and to limit exposure to threats. To accomplish this form of proactive assessment and enforcement, these approaches require accurate views of the hosts, their locations, and the services running on them. One common way of determining this information is through the use of a network-based vulnerability scanner. Active network-based vulnerability scanners (e.g., Nessus [25], Retina [11]) operate by sending crafted packets to hosts to inventory the targets, providing fingerprints of the host operating systems and the host network services. Conversely, passive scanners [8, 17, 26, 31] fingerprint software versions by auditing their network traffic and matching them with the signature database. They can continuously monitor target networks and are less intrusive to the targets. However, their scope is limited by the traffic they have access to and, as a result, passive scanners are usually deployed alongside active scanners. In addition to these generic scanners, there has been a great deal of recent work in specialized scanners that evaluate the security of popular applications such as web applications [6, 15].

Once the accurate inventory and service data is acquired, it can be used for a variety of tasks. For example, firewalls [9] are available to both networks and end hosts to enforce administrator polices, to block unwanted services [1, 2], and to prioritize the patching of vulnerable services [7, 19] before they are exploited. Often this reasoning makes use of attack graph representations of this inventory and service data to make their placement and configuration services. An attack graph is a graphical representation of all possible ways to compromise hosts in a network by taking advantage of its vulnerabilities. Sheyner et al. did the early work of attack graph generation using a model checking approach [27]. Subsequently, several improvements [5, 13, 20, 21] have been proposed to solve the scalability problem of the original attack graph approach. Another improvement is the introduction of link analysis methods in attack graphs to automate the analysis process [16, 24].

CANVuS varies from this existing work in that it does not provide new active or passive tests to determine a host configuration, nor does it propose a new representation or application of the host and service inventory data. Rather, the proposed architecture seeks to provide more up-to-date data with fewer costs than existing approaches by leveraging network context. In this sense, our work is relevant to other work in utilizing context to improve the performance and accuracy of a variety of security techniques [29]. For example, Sinha et al. leveraged the characteristics of the network workload to improve the performance of IDSes [30] and showed that building honeypots that are consistent with the network could improve the resilience of honeypots to attacks and improve their visibility [28]. Notions of managing numerous remote probing devices through a middleware layer was explored, though only in the context of IDSes, in the Broccoli system [14]. Cooke et al. built the Dark Oracle [10] that closely resembles the work in this paper in terms of methodology by using context-aware information to provide a database of network state, but it addressed primarily allocation information. Allman et al. proposed a general framework that also uses a trigger-based approach to do reactive network measurement [4]. While this is similar to our work in terms of the high-level idea, it tries to solve a different problem, and it contains no implementation or evaluation to demonstrate the effectiveness of their approach. More generally, to address the problems of comprehensive network visibility, a set of guidelines were outlined in [3] for three broad categories — basic functionality, handling and storage of data, and crucial capabilities. To our knowledge, no work has fully addressed all of these guidelines, although some work has been attempted to address the storage and querying of this ubiquitous visibility over time and space [34]. Our work makes progress towards comprehensive network visibility with the goal of creating a flexible, yet efficient unified network visibility system for CANVuS.

3 Motivation

The motivation for this work derives directly from our interactions with the University of Michigan's office of Information and Infrastructure Assurance (IIA) [33]. This group is tasked with: "(i) Facilitating campus-wide incident response activities, (ii) Providing services such as security assessments and consultation, network scans, education and training, and (iii) Managing IT security issues at the university level." As part of these roles, this office engages in quarterly scans of seven /16 subnets belonging to the University of Michigan. As part of an effort to evaluate this process, we assisted the IIA staff in analyzing several quarterly scans of this space by using both Nessus [25] and

Retina [11]. The results of this analysis were kept private to assist the security operation staff, but we were struck by several poignant observations from the experience:

- The scans generally take one and a half to two weeks to complete.
- In an effort to reduce the amount of time spent scanning, a significant number of vulnerability signatures present in the tools were not used.
- With the exception of a handful of departments, the scans of the IP space proceeded without knowledge of sub allocations in each department, scanning large blocks of space in their entirety.
- Due to the impact of work day availability patterns, the operators schedule the scans to occur only during working hours (i.e., 8 AM to 5 PM, Monday through Friday).
- Only 85% of the IP addresses in each scan were shared, the other 15% were unique.
- Only 85% of the total unique vulnerabilities discovered were present in both scans, with 15% of each scan's vulnerabilities appearing only in that scan.
- Only 56% of the configurations between two scans were unchanged for those IPs in common between the scans.

While surprising to us, the IIA staff were keenly aware of the dynamic nature of their network and the overhead imposed by the scanning activities. Although they deployed several stop-gap measures to deal with the effect of this dynamic network context (e.g., scan during work hours), these operators simply lacked the platform with which to achieve network-wide visibility.

4 Architecture and Design

In this section, we describe a context-aware architecture that provides a uniform view of network states and their changes for security applications. The architecture consists of three major components. The first component is a set of network monitors that are distributed over many network devices. The list of network devices to monitor could include switches, routers, and servers, but the architecture allows for other similar devices as well. The second major architectural component is a Context Manager, which converts data from network monitors to a network state database. The third and final component is the network state database that provides a uniform model for context-aware vulnerability scanning. Other context-aware applications may be built upon this database as well. A high-level diagram with the major components of the architecture is illustrated in Figure 1.

The design of this context-aware architecture is informed strongly by the design principles outlined in Allman et al. [3], especially those basic guidelines of scope, incremental deployability, and operational realities. We aim for a system that built for an individual enterprise and utilizes existing sources of data collected from infrastructure and services already deployed in the network. We utilize the existing common data formats (e.g., syslog, SNMP, Netflow) and store and access this data through common, extensible mechanisms (e.g., databases, SQL). Where necessary, we support probe-based mechanisms for extracting similar data from network data streams in the event that existing hardware is overloaded or does not support data export. With respect to the outlined data-oriented goals, we opt to focus on exploring data breadth over

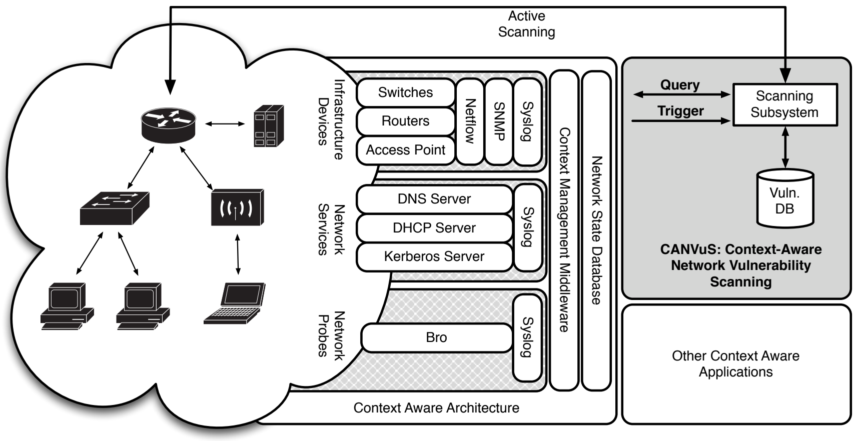

Fig. 1. Our context-aware network vulnerability scanning (CANVuS) architecture. The enterprise network is monitored by using data from existing physical infrastructure devices, network service appliances, and generic network probes. These heterogeneous sources of data are combined into a unified view of the network context which can be queried by context-aware applications or can have triggers automatically executed in response to certain contextual changes. In the case of CANVuS, contextual changes that indicate possible configuration changes are used to more efficiently scan network devices.

long term storage, smart storage management, graceful degradation, etc. The CANVuS application does not require extensive historical data, although we acknowledge that other context-aware applications will indeed require these functionalities and we look to leverage existing work in this direction for future versions of our architecture [34].

In the next three subsections, we first describe categories of monitoring points or data sources, from which creates a view of network context. Using this understanding, we then present the design of the Context Manager, which converts data from network monitors to the network state database in a uniform representation. Finally, we provide an example of what the network state database would look like for context-aware applications.

4.1 Sources of Data

Inferring network states and state changes is a challenging problem because the required information is distributed across many devices, network services, and applications. Thus, the key to capture the states and changes is to monitor the targets from a network perspective and approximate the context by aggregating network events from various data sources, which may lay in different layers of the network stack. To determine what data sources to use for event collection and integration, we first need to understand what types of network activities could be monitored and how they relate to changes in the network.

In this architecture, the monitors distributed across the network fall into three categories. The first category is monitors deployed in infrastructure devices, such as switches, routers and wireless access points. The advantage of having these types of monitors is that they provide detailed knowledge about the entire network, as well as the host information, with high resolution. This is a direct result of these devices providing the core hardware infrastructure for the network. For example, by querying the switches with SNMP, the system knows exactly when a particular physical machine is plugged into the network and when it leaves. However, because of the importance of these devices, tradeoffs are involved between the fidelity of data and the overhead (or difficulty) of the data collection.

The second category of monitors is for network services, which may include DNS, DHCP and even Kerberos services. The data observed by this second category of monitors provides additional detailed states about the hosts. Services in this class are the ones that are deployed in almost every enterprise and are critical to the operation of the network. For example, the data generated for DHCP service helps to distinguish between configuration changes for the same host and MAC-to-IP binding changes for different hosts. Depending on the types of the network, monitors in this category may require access to the syslog that are local to each service.

The final category consists of passive probes, which are deployed along with packet taps. These probes perform realtime analysis of network traffic that can be either generic TCP/UDP flows or application/OS specific to generate events. By monitoring TCP/UDP flows, the system gains the knowledge of which hosts are available and what services are running and being used by clients. In addition, using Deep Packet Inspecting (DPI) for an application protocol or OS fingerprint helps inventory the specific versions of applications (e.g., HTTP, SSH) or operating systems running in the network. Deviating version information for the same host directly indicates configuration changes for that host. For example, one could use Bro's protocol analysis to do connection reconstruction and application version fingerprinting [22] (which we used in our implementation of CANVuS). However, the visibility of passive probes is limited to the network traffic they have access to. In other words, silent applications/services will be missed, which is also the limitation of passive vulnerability scanners [8, 17, 26]. Thus, the passive probe is merely an additional type of monitor for collecting context data for observing a complete view of the network.

We acknowledge that host-based monitors exist for clients, but we decided that they are beyond the scope of this work. Intuitively, host-based monitors can provide the most accurate inventory information that effectively renders vulnerability scanning useless. Unfortunately, they are very difficult, if not impossible, to be enforced in large enterprise networks due to scalability and administrative reasons. However, as some traditional host-based programs, like antivirus software, are moved into the cloud [18], we may be able to build another type of monitors for potential in-cloud security services. The key advantage is that the in-cloud version of the services already have the visibility into the end hosts because they provide functionalities that used to be local to the hosts. Thus, the system will be able to obtain more fine-grained information about host activities without unacceptable modifications or performance penalties to the end hosts.

4.2 Context Manager

The Context Manager, a layer between data monitors and the network state database, infers network context (states and state changes) from aggregated data collected from various monitors and translates this to a uniform model. Specifically, the network states are a collection of simple facts about the target network, and they keep evolving as underlying hosts and services change. For example, these facts may include what hosts are available at the moment, what are the MAC addresses for a set of IPs in a certain period of time, or when is the first time a particular host connected to the network.

Existing libraries are used to read data from syslog, SNMP and Netflow that then get filtered and processed into a uniform representation of the network's state that is then inserted into the network state database. These modules can come from a programming languages standard library (e.g., Python's syslog module) or from third party libraries (e.g., PySNMP [12]). Additionally, the Context Manager supports a flexible framework for adding additional plugins to read input from new data monitoring sources and translate this data into a uniform model. These plugins can be thought of as data adapters that convert the source inputs canonical data format into a representation used in our architecture.

4.3 Network State Database

The network state database provides underlying model for which context-aware applications are written upon. The applications use standard database triggers or the programmable querying interfaces to interact with the database. The types of data that may be represented can be found in Table 1. This table shows how for that the data is uniformly represented across typical network abstraction layers, sources, and their respective data formats.

Network Layer	Data Source	Data Format	Description
Link Layer	Switch	SNMP	Mapping between a MAC address and a Physical Switch Port
	Switch	SNMP	Mapping between a MAC address and an IP address
	Bro	Ethernet	Mapping between a MAC address and an IP address
Network Layer	Router	SNMP	Network allocation
Transport Layer	Router	Netflow	New connection established
	Bro	TCP/IP	New connection established
Application Layer	DNS Server	Syslog	Name resolution
	Bro	UDP/IP	Name resolution
	DHCP Server	Syslog	Mapping between a MAC address and an IP address
	Bro	UDP/IP	Mapping between a MAC address and an IP address
	Bro	TCP/IP	Application version fingerprint

Table 1. Example contents of the network state database

5 CANVuS

In this section, we describe the implementation of CANVuS, a vulnerability scanning system, based on our context-aware architecture. It was implemented in Python to connect the network state database and a vulnerability scanner. In our implementation, we used Nessus [25]. Ideally, if all host configuration changes produce network artifacts, the need for network vulnerability scanning of these events would be straightforward. However, not all host changes have network evidence that can be captured by at least one of the monitors. Thus, CANVuS uses both query and callback interfaces from the network state database to leverage the context information and to maintain a history of scanning results in its own vulnerability database.

During the initialization phase, CANVuS queries the database for all available hosts as scanning candidates. Due to the constraints in hardware and network resources, all hosts are not scanned concurrently. Instead, candidates are queued for pending scans, whose size depends on the network conditions, the configured policies, and the amount of available physical resources. A scheduling strategy is needed here to select the next candidate to scan. For example, each candidate could be weighted based on their triggering events and scheduled accordingly. In the current implementation, a simple FIFO strategy is applied. At the same time, CANVuS registers a callback function with database triggers so that a new candidate will be appended to the pending queue when a change happens, unless the same target is already in the queue. To conduct actual scanning operations, Nessus is used, yet is modified to change its target pool from a static configuration file to the queue discussed above.

Conversely, if a scanned host has no events fired after N seconds since its last scan, and there is no evidence (including both the context information and former scanning results) indicating that it becomes unavailable, it will be added to the queue for another scan. Thus, each host is effectively equipped with its own timer. Once it expires, CANVuS will query the network state database and its vulnerability database to determine if further scanning is necessary. Clearly, the value of the timer is a configurable policy up to the decision of the operators. In addition, when registering callback functions, instead of simply subscribing all possible changes in the database, CANVuS defines a set of event-specific policies to filter the ones that are less relevant to host configuration changes.

The choices of polices involve tradeoffs and depend on the objectives of the administrators who manage this system. The purpose of our current implementation is not to provide a reference system for production use. Instead, we aim to figure out what events are more effective in detecting changes and what policies are more appropriate with the given network conditions and administrative requirements. Therefore, the policies used for our experiment were set to be as aggressive as possible so that an optimal solution could be determined by filtering unnecessary scans after the experiment was complete. More specifically, the default policy is that every single events triggers a scan. The only exception is the TCP event, since there are too many of them for each host, an active timeout is enforced to prevent them from overwhelming the system. On the other hand, if scans are being constantly triggered by inbound connections to ports that scanners fail to discover, a negative timeout is also enforced to suppress this type of event being fired over and over again. Further details regarding the revision of our trigger implementation and policy decisions are presented in the evaluation.

The vulnerability database is the central storage of vulnerability data for all of the hosts in the network. It keeps track of the result for every scan conducted against each host. In addition to the raw results generated by our modified Nessus scanner, each scan record contains following information:

– The time when this scan is triggered.
– The time when the backend scanner starts and finishes the scan.
– A list of open ports and the vulnerabilities on each port.
– A map from open ports to services.
– Operating system fingerprint (optional).
– The type of triggering event.

As more results are inserted into the vulnerability database, the information can be used in policy evaluation for further scans. Additionally, this information may be queried by administrators at any time for risk assessment or used by other security applications.

6 Evaluation

In this section, we evaluate CANVuS in a large academic network. The basic metrics used throughout this section is the number of scans conducted, which represents the resource consumption or overhead, and the latency of detecting configuration changes, which represents the system efficacy. Ideally, CANVuS should outperform periodic scanning with fewer number of scans by implicitly avoiding examining unallocated IP addresses and unavailable hosts. Moreover, CANVuS should also achieve lower detection latency as many host configuration changes create network evidence that trigger scans timely in our architecture.

We begin by discussing our experimental methodology. Then we show how CANVuS outperforms existing models in terms of the number of scans required and the detection latency. Next, we explore the impact of timeout values to the CANVuS system. The following section evaluates the contribution of various data sources to CANVuS and their correlations with observed changes on the hosts. We conclude the evaluation by discussing the scalability requirements of the context-aware architecture.

6.1 Experimental Methodology

The target network for the experiment is a college-level academic network with one /16 and one /17 subnet. There are two measurement points for the experiments. One is the core router for the entire college network. Because it has the access to the traffic between the Internet and the college network, there were two monitors built on it:

– TCP connection monitor: records the creation of new TCP connections.
– Application version monitor: records the version strings in protocol headers.

The second measurement point is a departmental network within the college that has the visibility into both the inbound/outbound traffic and more fine-grained inter-switch traffic. As result, the following monitors were deployed:

– ARP monitor: records the ARP messages probing for newly assigned IPs.
– DHCP monitor: records DHCP acknowledgment events.

- DNS monitor: records queries to certain software update sites.
- TCP connection monitor: as described above.
- Application version monitor: as described above.

This choice of measurement points and event monitors enables CANVuS to cover the network stack from the link layer to the application layer. Moreover, it also allows us to analyze the effectiveness of individual monitors with different visibility.

Based on these event monitors, CANVuS was deployed to scan the college network using a 12-hour active/inactive timeout and 1-hour negative timeout. In addition, another instance of a Nessus scanner was deployed for comparison purposes. It was configured to constantly enumerate the entire college network (a.k.a. the loop scanner). Both scanners were restricted to allow a maximum of 256 concurrent scans. The experiment lasted for 16 days in March, 2010, during which time the loop scanner completed 46 iterations. And it was interrupted by a power outage for a couple of hours at the end of the first week. Since we expect the system to be running long term, occasional interrupts would not have a major impact to the experiment results.

We performed a direct comparison between CANVuS and the loop scanner across both dimensions of resource consumption and detection latency. During the current exceptionally aggressive experiment, the loop scanner took less then 9 hours to finish one iteration. In realistic deployments, we envision using larger values and scanning less aggressively. Thus, the result of the loop scanner is only considered to represent the best performance that traditional periodic scanning systems can achieve in detection latency. More realistic values are represented below by sampling multiple 9-hour periods.

Ground truth for the experiments was established by identifying the period in which an observable network change occurred. Specifically, the scanning records from both scanners are first grouped together based on the MAC (if available) or IP address of each target host, and then each group of records are sorted by the time when each scan started. In each sorted group, a change is defined as two consecutive scan records (scans of unavailable hosts are ignored) with different sets of open ports. In addition, we assume that the change always happens immediately after the former scan finishes. Subsequent discussions that require the knowledge about ground truth are all based on this model unless otherwise noted. We approximated the ground truth in this way because it is infeasible to gain the local access to a large number of hosts in the target network to collect the real ground truth. As a result, we will not be able to analyze the true positive rate of CANVuS, and the average latencies for both scanners represent the upper bound, or the worst case.

6.2 CANVuS Evaluation

Table 2 lists the number of scans conducted by CANVuS with a break down by event types and the total for the loop scanner. The loop scanner has an order of magnitude more scans than CANVuS because only about 20% of the IP addresses in the target network are known to be allocated, and at any instant of time, the number of available hosts are even less than that. However, the loop scanner has to exhaust the entire IP blocks unless the address allocation and host availability information can be statically encoded, which is rarely the case in enterprise networks [10].

	CANVuS	Loop Scanner
Total	534,717	4,355,624
ARP	1.55%	N/A
DHCP	17.37%	
TCP	38.33%	
DNS	10.28%	
App. Protocol	0.03%	
Timeout	32.44%	

Table 2. The numbers of scans conducted by CANVuS and the loop scanner

In addition, the average detection latencies for changes discovered by CANVuS is 22,868.751 seconds versus 26,124.391 seconds for the loop scanner. Please recall that our assumption says the evidence of configuration changes will trigger scans instantaneously. However, the latency for CANVuS here is far from zero. This anomaly is caused by the fact that we used the combined scanning results to approximate the ground truth and timeout-based scanning was still applied in some situations when no network network changes occurred.

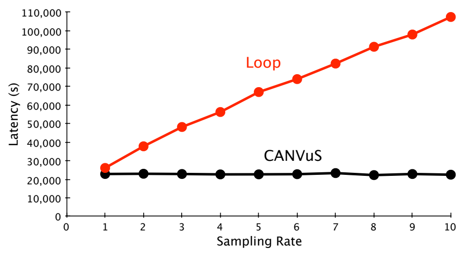

Fig. 2. A comparison of the detection latency with sampled results for the loop scanner.

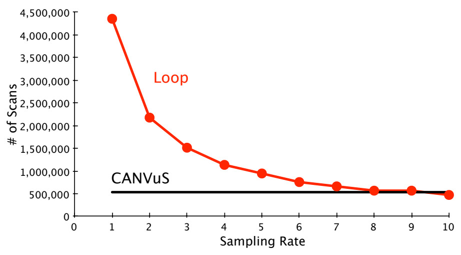

Fig. 3. A comparison of the number of scans with sampled results for the loop scanner.

Moreover, the latency for CANVuS is not significantly better than that of the loop scanner. This is because the configuration for the loop scanner is already very aggressive and represents the best performance that traditional periodic scanning systems can achieve in detection latency. To make the loop scans less aggressive and to demonstrate the tradeoff in scanning costs, the data set is sampled with a rate from 1 to 10 to include both the original case and the case that both scanners have a comparable number of scans. Figure 2 illustrates the result. The curve for the loop scanner goes up almost linearly because of the linear sampling, while the curve for CANVuS goes slightly up and down because the approximated ground truth has been changed after sampling. In addition, Figure 3 shows the corresponding changes for the number of scans.

6.3 Timeout-Based Scanning In CANVuS

As discussed previously, a timeout-based scanning approach is used along with the trigger-based scanning as many configuration changes are not observable through network events. However, unlike traditional periodic scanning, which randomly picks scanning targets in a large pool with fixed cycles, the timeout mechanism in CANVuS is still context-aware. Specifically, the system uses the context data to infer IP allocation information and host availability patterns so that only the hosts that are believed to be connected to the network will be scanned. These timeouts are assigned per-host timer and are based on the network state, scanning history, and administrators' policy decisions. As a result of these approach, fewer scans are "wasted" on hosts that don't exists or are unavailable. Taken another way, given the same number of scans, CANVuS is more likely to examine a larger number of active hosts and detect host changes with lower latency than the periodic scanning system.

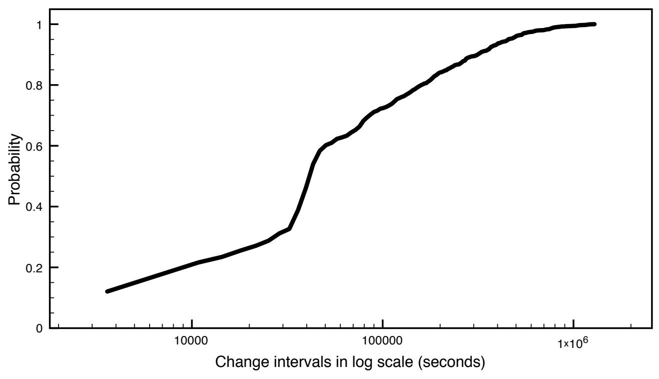

Fig. 4. The CDF for the intervals of detected changes.

Figure 4 depicts the distribution of intervals between *all* detected changes the values ranging between hours and days. The bias demonstrated in the middle of the graph is the result of our experimental methodology and the choice of 12 hours for our active/inactive timeout.

Thus, in practice, we determining the timeout value based on operators' administrative requirements. For example, using a timeout value at the order of a week, which covers the changes on the tail of the curve in Figure 4, would be reasonable. An alternative strategy is to halve the timeout value if a change is detected accordingly, but otherwise double it adaptively modifying a host's timeout value. In either case, there is clearly a tradeoff between the number of scans and the detection latency, which is also the case for normal periodic scanning.

6.4 Exploring the Impact of Various Data Sources and Triggers

In this subsection, we study the relationship between triggering events and the captured configuration changes. Specifically, the study is focused on their temporal correlations instead of causalities, which requires detailed control over the target hosts.

To do this, two problems need to be addressed. First, since most subnets in the target network use DHCP to assign IP addresses (despite whether the address mapping is dynamic or static), the changes witnessed may actually be mapping changes instead of changes in configuration. To eliminate the negative impact of dynamic address assignment, which would obscure the analysis results, the discussion in this subsection is confined to the 535 hosts that were assigned exactly one unique IP address during the experiment period. They are extracted from the 1,691 scanned IP addresses in the nine /24 departmental network subnets for which we have complete DHCP message logs.

Among these 535 hosts, 1,985 changes were detected by CANVuS during the experiment. After merging with the results from the loop scanner, the number of detected changes becomes 2,145, where the increase is an artifact of the method used to generate the approximated ground truth. However, many changes are considered to be ephemeral or short-lived, which is the second problem that must be handled. In other words, certain ports appear for a short while and then disappear without exhibiting real configuration changes. Many client programs may exhibit this behavior. For example, the client side of Apple's iTunes uses port 5353 to communicate with each other for sharing. P2P download software is another example. This type of changes provide little value in revealing the temporal correlations between changes and triggering events, due to their short lifetime.

Triggering Event	Count	With network evidence
ARP	1	1
DHCP	15	10
TCP	2	2
DNS	3	1
Timeout	4	0

Table 3. Permanent changes categorization

As a result, we only consider those long-term or permanent changes to study their temporal correlations with triggering events. However, the word 'permanent' is not well defined, given the limited experiment period. Thus, for the simplicity of analysis, we only examine hosts that had exactly one change during the entire 16-day period because these changes are most likely to be permanent unless they were detected at the very end of the experiment. Among the 535 hosts, 25 of them fall into this category. Though this is not statistically significant, they still provide important clues for us to find appropriate policy setting for the target network. Recall that our conjecture is that most permanent changes have network evidence that can be witnessed and used for creating triggers to achieve timely detection. By manually going through all these changes and analyzing all logged events that happened around the time when the changes were detected, we find

56% of them have network evidence that has strong temporal correlation with changes, which means they either have triggered or could have triggered scans to detect the changes.

Table 3 is a summary of the analysis results in detail. Several things should be noted here. First, all the permanent changes we studied that were captured by timeout exhibited no network evidence at all. This is a limitation of our system using pure network events. Unless some level of host information is monitored, this timeout-based method cannot be completely replaced with the trigger-based approach. In addition, a significant portion of changes were detected via DHCP and ARP events, which corresponded to hosts reboots. This is reasonable because many configuration changes may not take effect until the host is restarted. Finally, the rest of the changes corresponded to activities on the service or process level.

Moreover, we argue that although ephemeral changes may not be helpful in studying the temporal correlations, they are still relevant to risk assessment. Despite the possibility of being exploited by sophisticated attackers, many short-lived but well-known ports may be used to tunnel malicious traffic for cutting through firewalls. For example, the TCP event monitor captured some occasional events through port 80 for certain hosts, while the application event monitor failed to fingerprint any version information for them, which means the traffic did not follow the HTTP protocol. Thus, in both cases, it is valuable to detect these short-lived changes, but there is no guarantee for the loop scanner to achieve this goal. In fact, the loop scanner tends to miss most of these changes once its scanning period greatly increases (e.g., in the order of weeks). With the help of TCP events, CANVuS can fire scans immediately after there is traffic going through these ports. And there are 35 changes exclusively captured by CANVuS that fall in this category. Conversely, if there is no traffic ever going through the short-lived ports, while CANVuS may also miss them, the resulting risk is much lower because attackers have no chance to leverage them either.

6.5 Scalability Requirements of the Context-aware Architecture

Figures 5 and 6 constitute a summary for the scale of the data from the departmental monitors done in hour intervals. The number of raw packets or flows per hour is counted in Figure 5. This raw data was observed at the various network monitors and probes before being converted by the Context Manager into the network state database. We note that the first three days worth of data are missing in our graph due to the misconfiguration of the monitoring infrastructure. For the duration of our experiment, we observed flows on the order of 16 million per hour at its peak and on average around 4 million per hour. Other noticeable observations include a typical average of 1 million DNS packets per hour and about 12 thousand ARP packets per hour. These four graphs in Figure 5 show that this system must handle adequately large volumes of traffic due to its distributed nature.

Figures 6 shows the number of events per hour that triggered scans after being converted by the Context Manager. Compared to the graphs in Figure 5, the volume of events generated from the raw data was greatly reduced. To highlight the number of flows and DNS packets went from the order of millions to the low hundreds. ARP packets when from the order of thousands to tens. This shows that our Context Manager

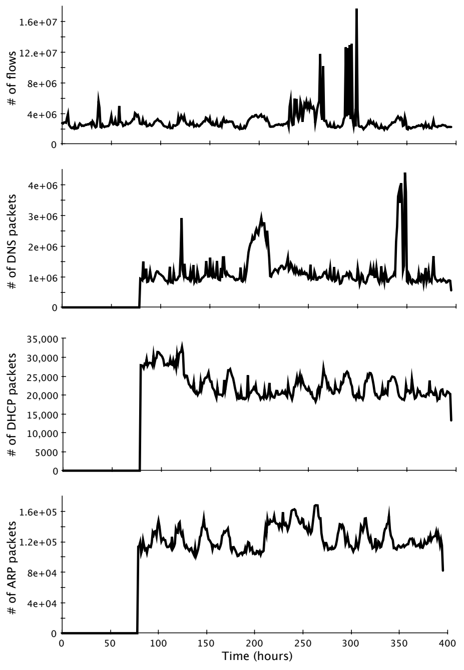

Fig. 5. The scale of the raw data.

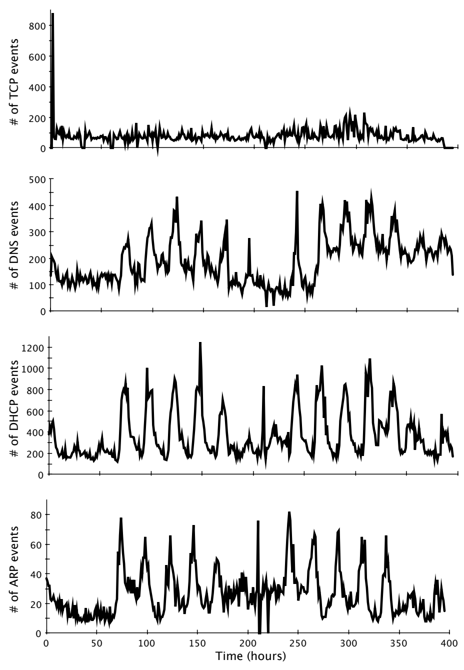

Fig. 6. The scale of the events.

is able to greatly reduce large volumes of data to something manageable for our event-based vulnerability scanning.

In addition, the cumulative number of unique MAC addresses in the departmental network is shown in Figure 7, which quantifies the scale of the physical boxes within the department (only for the second measurement point) that should be audited. We observed that slight bumps indicate new observances of MAC addresses over the course of a day while plateaus occurred over the weekends. We speculate that the observance of new, unique MAC addresses will level off if given a longer period of time to run our experiments. This graph also gives insight in bounding to the amount of raw and event-generated traffic that would be observed by the detection of fewer and fewer unique MAC addresses.

7 Risk Mitigation and Analysis

In this section, we examine our efforts in minimizing the harm to users, services, hosts, and network infrastructure while performing our experiments. We understand that active probing involves the use of network resources and interaction with product services. In consultation with the Computer Science and Engineering Departmental Computing Office, the College of Engineering Computer Added Engineering Network, and the University of Michigan office of Information and Infrastructure Assurance, we developed

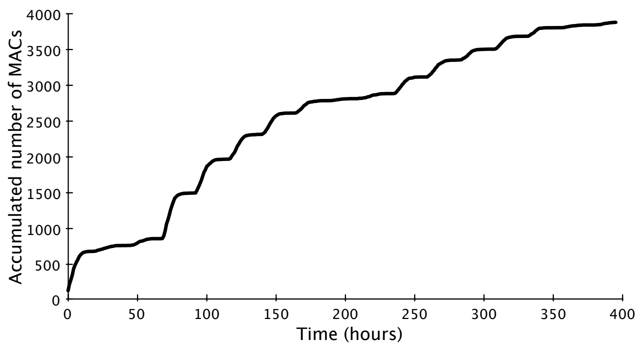

Fig. 7. The accumulated number of unique MAC addresses in the departmental network.

the following research plan to mitigate the impact on hosts, services, and network infrastructure: (i) to minimize the effect of network scanning, we limited the bandwidth available to our scanning devices, (ii) We implemented a whitelisting feature to our scanning, and the engineering computer organization broadcasted an opt-out message to all departmental organizations prior to our experiment (along with the complete research plan), (iii) We applied only those polices consistent with the Nessus "Safe Check Only" label.

Acknowledging that a network's security context is considered sensitive information and data such as MAC addresses and IP addresses have been viewed as personally identifiable information in several contexts, we took steps to assure that the "research records, data and/or specimens be protected against inappropriate use or disclosure, or malicious or accidental loss or destruction" according to our IRB guidelines. This includes, but is not limited to the following official categories: Locked office, Restricted access, Restrictions on copying study-related materials, Access rights terminated when authorized users leave the project or unit, Individual ID plus password protection, Encryption of digital data, Network restrictions, No non-UM devices are used to access project data, Security software (firewall, anti-virus, anti-intrusion) is installed and regularly updated on all servers, workstations, laptops, and other devices used in the project. Due to the technical nature of the work, we did not seek IRB approval for the project as we did not feel they were prepared to understand the risks of this work. The proposed research plan was instead approved through the College of Engineering Dean of Research, and additionally approved by the departmental, college, and university computing organizations specified above.

8 Limitations and Future Work

While our initial evaluation demonstrates the promise of a context-aware approach to vulnerability scanning, it does highlight several limitations which form the foundation for future work in this area. First, the accuracy of our evaluation is hampered by the use of network vulnerability scanning results as the sole ground truth for measuring host configuration changes. In addition to the previously discussed limitation that a network-based scanner provides only an approximate view of a host changes, this approach also limited the granularity of our measurements to the polling frequency of the network scanner. To overcome this issue, we plan on developing a host agent that is capable of collecting fine-grained information on local changes and deploying it on a network with a large number of different hosts (e.g., end hosts vs. application servers). A second rich area for future work is the exploration of new triggers (either new events or combinations of these events) for host configuration changes. Currently, the most effective events were generated by the DHCP monitor and corresponded to host reboots. In the future, we plan to increase the diversity of trigger events and explore other types of network evidence for host changes.

9 Conclusion

In this paper, we proposed a context-aware architecture that provides information about the network states and their changes for enterprise security applications. We described how this architecture converts network data from infrastructure devices, network services, and passive probes to a uniform representation stored in the network state database. Then we introduced CANVuS, a context-aware vulnerability scanning system built upon this architecture that triggers scanning operations based on changes indicated by network activities. We experimentally evaluated the system by deploying it in a college-level academic network and comparing CANVuS against an existing system. We found that this approach outperforms existing models in low detection latency, while consuming fewer network resources.

Acknowledgments

The authors wish to gratefully acknowledge the following colleagues at the University of Michigan for their assistance in performing this work: Paul Howell, Kirk Soluk, Dawn Isabel, Dan Maletta, Kevin Cheek, and Donald Winsor. This work was supported in part by the Department of Homeland Security (DHS) under contract numbers NBCHC080037, NBCHC060090, and FA8750-08-2-0147, the National Science Foundation (NSF) under contract numbers CNS 091639, CNS 08311174, CNS 0627445, and CNS 0751116, and the Department of the Navy under contract N000.14-09-1-1042.

References

1. Muhammad Abedin, Syeda Nessa, Ehab Al-Shaer, and Latifur Khan. Vulnerability analysis for evaluating quality of protection of security policies. In *Proceedings of the 2nd ACM workshop on Quality of protection (QoP'06)*, Alexandria VA, October 2006.
2. Mohammad Salim Ahmed, Ehab Al-Shaer, and Latifur Khan. Towards autonomic risk-aware security configuration. In *Proceedings of the 11th IEEE/IFIP Network Operations and Management Symposium (NOMS'08)*, Salvador, Bahia, Brazil, April 2008.

3. Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, and Nicholas Weaver. Principles for developing comprehensive network visibility. In Niels Provos, editor, *3rd USENIX Workshop on Hot Topics in Security, July 29, 2008, San Jose, CA, USA, Proceedings*. USENIX Association, 2008.
4. Mark Allman and Vern Paxson. A reactive measurement framework. In *Proceedings of the ninth Passive and Active Measurement conference (PAM'2008)*, Cleveland, Ohio, April 2008.
5. Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, Washington DC, November 2002.
6. Jason Bau, Elie Bursztein, Divij Gupta, and John Mitchell. State of the art: Automated black-box web application vulnerability testin. In *Proceedings of the 31st IEEE Symposium on Security & Privacy (S&P'10)*, Oakland, CA, May 2010.
7. Steve Beattie, Seth Arnold, Crispin Cowan, Perry Wagle, Chris Wright, and Adam Shostack. Timing the application of security patches for optimal uptime. In *Proceedings of the 16th Annual LISA System Administration Conference*, Philadelphia, PA, USA, November 2002.
8. Edward Bjarte. Prads - passive real-time asset detection system. `http://gamelinux.github.com/prads`.
9. W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security; Repelling the Wily Hacker*. Addison Wesley, Reading, MA, 1994.
10. Evan Cooke, Michael Bailey, Farnam Jahanian, and Richard Mortier. The dark oracle: Perspective-aware unused and unreachable address discovery. In *Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI '06)*, May 2006.
11. eEye Digital Security. Retina - network security scanner. `http://www.eeye.com/Products/Retina.aspx`.
12. Ilya Etingof. Pysnmp. `http://pysnmp.sourceforge.net/`.
13. Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06)*, Miami Beach, FL, December 2006.
14. Christian Kreibich and Robin Sommer. Policy-controlled event management for distributed intrusion detection. In *ICDCS Workshops*, pages 385–391. IEEE Computer Society, 2005.
15. Sean McAllister, Engin Kirda, and Christopher Kruegel. Leveraging user interactions for in-depth testing of web applications. In *Proceedings of the 11th International Symposium on Recent Advances In Intrusion Detection (RAID'08)*, Boston, MA, September 2008.
16. Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. Ranking attack graphs. In *Proceedings of the 9th International Symposium On Recent Advances In Intrusion Detection (RAID'06)*, Hamburg, Germany, September 2006.
17. Microsoft. Watcher - web security testing tool and passive. `http://websecuritytool.codeplex.com`.
18. Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proceedings of the 17th USENIX Security Symposium (Security'08)*, San Jose, CA, July 2008.
19. Jon Oberheide, Evan Cooke, and Farnam Jahanian. If It Ain't Broke, Don't Fix It: Challenges and New Directions for Inferring the Impact of Software Patches. In *12th Workshop on Hot Topics in Operating Systems (HotOS XII)*, Monte Verita, Switzerland, May 2009.
20. Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS'06)*, Alexandria, VA, October 2006.

21. Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. Mulval: A logic-based network security analyzer. In *Proceedings of the 14th USENIX Security Symposium (USENIX Security'05)*, Baltimore, MD, August 2005.
22. Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.
23. M. Roesch. Snort: Lightweight intrusion detection for networksx. In *Proceedings of the 13th Systems Administration Conference (LISA)*, pages 229–238, 1999.
24. Reginald E. Sawilla and Xinming Ou. Identifying critical attack assets in dependency attack graphs. In *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS'08)*, Malaga, Spain, October 2008.
25. Tenable Network Security. Nessus - vulnerability scanner. `http://www.nessus.org`.
26. Tenable Network Security. Nessus passive vulnerability scanner. `http://www.nessus.org/products/pvs/`.
27. Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of 2002 IEEE Symposium on Security and Privacy (S&P'02)*, Oakland, CA, May 2002.
28. Sushant Sinha, Michael Bailey, and Farnam Jahanian. Shedding light on the configuration of dark addresses. In *Proceedings of Network and Distributed System Security Symposium (NDSS '07)*, February 2007.
29. Sushant Sinha, Michael D. Bailey, and Farnam Jahanian. One Size Does Not Fit All: 10 Years of Applying Context Aware Security. In *Proceedings of the 2009 IEEE International Conference on Technologies for Homeland Security (HST '09)*, Waltham, Massachusetts, USA, May 2009.
30. Sushant Sinha, Farnam Jahanian, and Jignesh M. Patel. Wind: Workload-aware intrusion detection. In *Symposium on Recent Advances in Intrusion Detection (RAID'06)*, Hamburg, Germany, September 2006.
31. Sourcefire. Sourcefire rna - real-time network awareness. `http://www.sourcefire.com/products/3D/rna`.
32. Sourcefire, Inc. Clamav antivirus. `http://www.clamav.net/`, 2008.
33. University of Michigan. University of Michigan — ITS — Safe Computing — IT Security Services Office. `http://safecomputing.umich.edu/about/`, April 2010.
34. Matthias Vallentin. VAST: Network Visibility Across Space and Time. Master's thesis, Technische Universitat Munchen, Jan 2009.