# Reflections on the Engineering and Operation of a Large-Scale Embedded Device Vulnerability Scanner

Ang Cui and Salvatore J. Stolfo
{ang,sal}@cs.columbia.edu

Department of Computer Science
Columbia University
New York NY, 10027, USA

**Abstract.** We present important lessons learned from the engineering and operation of a large-scale embedded device vulnerability scanner infrastructure. Developed and refined over the period of one year, our vulnerability scanner monitored large portions of the internet and was able to identify over 1.1 million publicly accessible trivially vulnerable embedded devices. The data collected has helped us move beyond vague, anecdotal suspicions of embedded insecurity towards a realistic quantitative understanding of the current threat. In this paper, we describe our experimental methodology and reflect on key technical, organizational and social challenges encountered during our research. We also discuss several key technical design missteps and operational failures and their solutions.

## 1 Introduction

Over the past year, the Columbia University Intrusion Detection Systems Lab conducted a quantitative study on the distribution of trivially vulnerable embedded network devices openly accessible over the internet. Trivially vulnerable devices are those which have well-known default root-level credentials configured on publicly accessible administrative interfaces. At the time of writing, our vulnerability scanner has identified over 1.1 million such vulnerable embedded devices. Our latest data indicates that approximately **1 in 5** publicly accessible embedded device on the internet is configured with well-known default administrative credentials. In order to accurately establish a quantitative lower bound on the number of vulnerable devices on the internet, we engineered a parallelized scanning infrastructure capable of monitoring the entire IPv4 space with reasonable speed. Each automated scan takes approximately two weeks and produces a snapshot of all accessible HTTP and Telnet servers on the internet, along with a list of embedded devices which are confirmed to be trivially vulnerable. Table 1 shows several key metrics of our latest scan results. Figure 1 graphically illustrates the distribution of trivially vulnerable embedded devices across IPv4 space.

| Total IPs Scanned | Devices Targeted | Vulnerable Devices | Vulnerability Rate |
|---|---|---|---|
| 3,223,358,720 | 5,652,358 | 1,134,535 | 20.07% |

**Table 1.** Scale and Result of the Latest Global Default Credential Scan.

Each iteration of the global scan sweeps across over 3.2 billion IP addresses, cataloging every publicly accessible HTTP and Telnet server on the internet over a period of two weeks. The initial output of each responding server, various meta-data, as well as logs and results of vulnerability tests performed on every candidate embedded device is stored in a cluster of SQL databases. On average, each global scan records over 90 GB of data and involves over 10 billion database transactions. The remainder of this paper is a description and reflection on the successes and failures of our engineering process and will discuss:

- Major design choices of our scanner infrastructure.
- Technical, organizational and social challenges encountered throughout the course of the study.
- Description and reflection on engineering missteps and failures of our system and protocol.
- Lessons learned from this large-scale study.

The remainder of this paper is organized as follows. Section 2 briefly describes the experimental methodology of our study. For a more detailed description of our study and findings, please see [1]. Section 3 discusses major technical bottlenecks and missteps, the iterative redesign of our scanner infrastructure, as well as organizational challenges and social implications of our study. Section 4 summarizes key lessons learned from our experiences engineering and operating our scanner infrastructure. Lastly, we present our conclusion and acknowledgements in Sections 5 and 6. Appendix C contains logical and physical diagrams of our project's IT infrastructure.

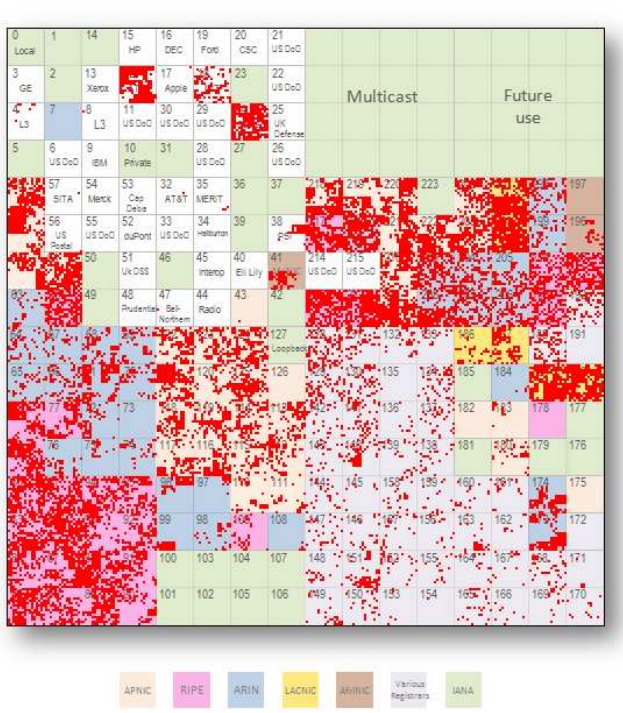



**Fig. 1.** Distribution of Vulnerable Embedded Devices in IPv4 Space.

## 2 Experimental Methodology

The default credential scan process is straightforward and can be broken down into three sequential phases: **recognizance**, **identification**, and **verification**.

**Recognizance:** First, *nmap* is used to scan large portions of the internet for open TCP ports 23 and 80. The results of the scan is stored in a SQL database.

**Identification:** Next, the device identification process connects to all listening Telnet and HTTP servers to retrieve the initial output of these servers[1]. The server output is stored in a SQL database then matched against a list of signatures to identify the manufacturer and model of the device in question.

**Verification:** The verification phase uses an automated script to determine whether it is possible to log into devices found in the identification phase. This script uses only well known default root credentials for the specific device model and does not engage in any form of brute force password guessing. We create a unique *device verification profile* for each type of embedded device we monitor. This profile contains all information necessary for the verification script to automatically negotiate the authentication process, using either the device's Telnet or HTTP administrative interface. Figure 2 shows two typical device verification profiles, one for the administrative Telnet interface for Cisco switches and routers, the other for the HTTP administrative interface for Linksys WRT routers using HTTP Basic Authentication. Each device verification profile contains information like the username and password prompt signatures, default credentials as well as authentication success and failure conditions for the particular embedded device type. Once the success or failure of the default credential is verified, the TCP session is terminated and the results are written to an encrypted flash drive for off-line analysis.

The device selection process is manual and iterative. We begin by analyzing data gathered by the recognizance phase of our scanner, which collects the initial output from active Telnet and HTTP servers found by NMAP. We maintain three sets of signatures: non-embedded devices; non-candidate embedded devices; and candidate embedded devices. Signatures of non-embedded devices include those of popular HTTP servers such as Apache and IIS as well as Telnet common authentication prompts of general purpose operating systems. Signatures of non-candidate embedded devices include those that do not ship with a well known default credential[2]. Signatures of candidate embedded devices include string patterns that positively identify the device as one that we are actively

[1] In case of HTTP, we issue the 'get /' request

[2] For example, the Polycom VSX 3000 video conferencing unit uses the device's serial number as the default password.

```
root:
    username_prompt: ['sername:']
    username: cisco
    askuser: true
    passstr: ['assword:']
    incorrect: [sername, assword]
    success: ['\#']
    passwords: ['cisco']
    deviceType: cisco
    isActive: 'true'
```

```
root:
    authType: basicAuth
    passwds: ['admin']
    authRealm: WRT54G
    username: ''
    deviceType: linksys-wrt
    loginURL: '/'
    isActive: 'true'
```

**Fig. 2.** Two typical device profiles, stored as YAML files. Left: Cisco telnet. Right: Linksys WRT HTTP.

monitoring. After the recognizance data is tagged using these three signature sets, we manually inspect and tag the remaining records, creating new signatures and device verification profiles. The process of identifying and incorporating new embedded devices into the scanner is arguably the most labor intensive part of the project. Once the scanner infrastructure was sufficiently developed, its maintenance and operation required little manual intervention. However, as the popularity of different embedded devices rise and fall, constant effort was required to update what is essentially an rule-based expert system.

The technical methodology of our experiment is simple and straight forward. However, the scale of the operation and the sensitivity of the data collected by our scanner posed several major challenges. The next section will discuss the technical, organizational and social challenges we encountered throughout our initial study.

## 3 Major Challenges

Using *nmap* to scan the internet and writing the initial script to verify trivial vulnerability of specific embedded devices is not technically difficult. However, ensuring that the entire scanner infrastructure is scalable, safe (for the scanned networks as well as our host network), secure, resilient and accurate gave rise to several interesting technical challenges. Section 3.1 describes the iterative improvements we made to the scanner infrastructure throughout its development.

Furthermore, conducting an *active* study involving the entire internet and handling and safe guarding a database of over a million vulnerable embedded devices complicated our efforts beyond mere technical hurdles. Scalability and security concerns were compounded by social and organizational implications of operating within an university environment. Safeguards were instituted within our experimental protocol and internal IT environment to prevent intentional and accidental exfiltration of sensitive data. For example, an isolated and fortified network environment had to be created for this project to ensure that the rest of the university and even other members of our research group could not access our production scanners and databases (See Appendix C). Furthermore, an identical development environment was created to allow new project students to contribute to our code base without granting them access to the actual vulnerability database. Section 3.2 discusses the measures taken to ensure that our study is conducted ethically and that sensitive vulnerability data is safe-guarded at every phase of our study.

The public-facing nature of our global vulnerability assessment further complicates our efforts for at least three reasons. We intentionally carried out the scans publicly without employing any stealthy or anonymizing techniques. This is done primarily to ensure that no harm is done by our scanners, and that network operators can easily communicate with our research group. With the help of our university NOC and network security group, we formed an informal response team in order to handle the anticipated responses to our public scanning activities. In retrospect, the volume of responses received regarding our scans was far smaller than first anticipated. This may be an indication that most networks no longer bother with scan detection, or they may ignore scanning since it is so commonplace.

Lastly, the dissemination of our vulnerability data to the proper parties can help network operators secure large numbers of vulnerable embedded devices. However, vulnerability disclosure must be done carefully through the proper channels. While we recognized the need to alert operators of large populations of vulnerable devices, we lacked the experience and means to carry this out properly. Section 3.3 describes how we partnered with Team Cymru to disseminate our research findings in order to reduce the vulnerable device population.

### 3.1 Technical Challenges and Missteps

The scanner infrastructure underwent three major redesigns over a period of one year. Each redesign iteration involved a partial rewrite of our code base while the network topology of our infrastructure remained unchanged.

After the technical goals of our quantitative study were first outlined, a working scanner was hastily written using a collection of bash and python scripts which invoked *nmap* and processed its XML output. This proof of concept scanner was able to quickly identify thousands of trivially vulnerable embedded device on our own University's network.

Encouraged by our initial success, the research group set out to make several obvious improvements to our infrastructure:

**Automation:** The initial scanner implementation produced encouraging results. However, its operation required almost constant manual intervention. We implemented all three phases of the scanner workflow as individual command-line tools. Other scripts were written to map out sections of the internet for the scanner to target. Yet other scripts were written to tabulate and analyze the raw output of the scanner to produce the final vulnerability report. Automation and coordination between each individual component of the scanner was clearly needed. To achieve this, we migrated all standalone tools into Apache/mod_py and created a master job scheduler. Using mod_py allowed us to expose the scanner's API over HTTP. This greatly reduced the complexity of the job scheduler process, which simply made periodic HTTP GET requests to Apache. Leveraging Apache's infrastructure also gave us improved stability, parallelism and logging at an acceptable overhead.

**Data Wrangling:** The initial scanner stored all intermediate outputs as plain text files, which quickly ran into scalability issues when we scanned even modest sized networks. To solve this problem, the scanner was modified to store all intermediate outputs in a MySQL database. This improved the performance of the scanner and the analytics code which produced the final vulnerability report and greatly simplified our data generation code which delegated all caching, concurrency and transactional storage logic to the MySQL server.

Furthermore, the use of SQL standardized the schema of the various record types used by the scanner and allowed us to easily export intermediate data to other applications via ODBC. A dedicated MySQL server was created to service all scanner nodes to centralize data storage. A mirrored disk volume was used to store the MySQL database to prevent data loss due to disk failure. A set of *cron* scripts were also put in place to backup the database contents on an hourly basis to a separate disk volume.

**Scalability:** The initial scanner design was limited to a single host. The migration of the scanner API over to Apache/mod_py and MySQL allowed us to easily parallelize scanning activity across an array of identical scanner nodes. During the first redesign of the scanner infrastructure, the single server implementation was scaled out to a scanner which utilized 8 scanner nodes and a single centralized database.

The second iteration of the scanner design automated the entire workflow, improved performance and reliability and simplified the codebase by leveraging exiting platforms like MySQL and Apache. However, we quickly noticed several major bottlenecks within the new design.

Most noticeably, the central database became overloaded during periods of high utilization. Since the job scheduler and all scanner nodes depended on the central MySQL server, the entire system frequently thrashed and halted. To solve this problem, the job scheduler was improved to frequently monitor the load of all scanner nodes to prevent over-allocation of scan jobs to nodes which are already heavily utilized. Node utilization polling was done using SNMP once per second. Scanner nodes were removed from the allocatable queue when their system load value exceeded a configurable threshold, or when the maximum allowable running *nmap* instances is exceeded.

Each scanner node is designed to execute up to 32 *nmap* scans and potentially thousands of vulnerability verifications simultaneously[3]. Our code did not restrict the total number of simultaneous MySQL connections created by each node. This quickly caused the MySQL server to reject incoming connections as multiple scanner nodes were brought online. To solve this problem, we incorporated a connection pool controller into our scanner node using the pysqlpool[4] package. Furthermore, we increased the default *max_connections* value in MySQL, which was set to a conservative 151 by default.

Inefficient SQL queries within the analytics and report generation code frequently overloaded the central database, causing the entire scanner system to halt. We reasoned that delegating read and write operations to separate physical servers would improve performance by isolating the resource drain of the report generation code from the scanner nodes. While this was certainly true, it also led us to our first major technical misstep.

[3] We arrived at this number through experimentation on our own hardware. Various kernel options can be tweaked to improve *nmap* performance. For example, see http://seclists.org/nmap-dev/2008/q1/354

[4] http://code.google.com/p/pysqlpool/

**Technical Misstep #1: Database Cluster Fiasco.** The central database became a prominent performance bottleneck after we implemented the first version of the parallelized scanner infrastructure. To improve performance, we experimented with using the Community Edition of MySQL Cluster. In theory, this would give us a single high performance logical MySQL instance which is scaled out to several physical machines. A three node MySQL cluster was deployed, along with MySQLProxy[5], which allowed us to control how read and write queries are dispatched within the database cluster.

This solution delivered the performance boost we needed, but came at the cost of unreliability and ultimately, total data loss. The physical cluster nodes were configured according to available best practice guides and connected to a single low-latency gigabit switch (Cisco 4948). However, we noticed that various database nodes sporadically dropped out of the cluster during periods of high utilization. This eventually led to database inconsistency, long periods of re-synchronization, and after a two week trial-run period, total data loss on several large tables.

We still plan to revisit the MySQL Cluster solution in the future, as a HA MySQL cluster is preferable to our current solution. In the interest of time, we abandoned the clustered approach in favor of replicating the central database sever. Instead of a single scanner infrastructure, we essentially created three identical deployments and divided the IP space to be scanned equally between each scanner cluster.

This deployment also provided an unexpected advantage. Since *nmap* and our vulnerability verification code are both sensitive to network latency and packet loss, we divided the target IP space geographically among the three scanner clusters. This allowed us to customize latency and packet loss related parameters to best suite the part of the world a particular scanner cluster was targeting.

**Technical Misstep #2: Data At Rest, Forever.** The second technical fiasco struck shortly after the first. Recognizing the sensitivity of our vulnerability report, which contained the IP, device type, username and password of every vulnerable embedded device discovered by our scanners, we put forth a policy of encrypting any vulnerability-related data leaving our production environment. Communication with Team Cymru, who mediated data dissemination to vulnerable counter-parties was done over PGP. To further reduce the possibility of unauthorized exfiltration of sensitive data, once analysis is performed on a scan iteration, all sensitive data is purged from our production databases. One copy of the final report is stored on an IronKey Encrypted USB drive for posterity. We chose the IronKey for its onboard AES-CBC encryption engine, its self-destruct functionality, and the ability to configure it with no possibility of password recovery.

Suffice it to say, several months into the vulnerability study, the only member of the research team entrusted with the IronKey passphrase forgot the passphrase. Said passphrase was never written down or told to anyone else. Thus, several months of vulnerability data was lost as the IronKey quietly self-destructed in front of most of the research group. Despite this setback, we still currently employ the same data at rest protocol as before.

**Performance Monitoring.** Detailed performance monitoring was the latest major addition of the scanner infrastructure. Despite our efforts to improve the job scheduler implementation, scanner nodes and database servers inevitably thrashed or became over or under utilized. At the time of implementation, the scanner infrastructure had grown from a single standalone host to a three-cluster deployment, each composed of one database server and six scanner nodes. A read-only analysis database server was also created. In total, 22 machines were operating 24x7. Monitoring the performance of these machines again required significant manual intervention. As a response, we augmented the standard linux SNMPD with performance metrics specific to our scanner infrastructure. Such metrics includes the number of active verification jobs, the number of incomplete transactions from all phases of the workflow, the current number of running *nmap* instances, etc. We then created a simple performance monitor dashboard which polled SNMP values and graphed them using RRDTool. TIme permitting, we would like to incorporate these performance metrics into our existing Nagios deployment.

The visibility that the performance dashboard gave us allowed us to quickly fix several previously unknown performance issues. The graphical dashboard also allowed us to easily verify that the scanner was operating with reasonable speed. This greatly reduced the time to resolution of most routine problems the scanner encountered.

### 3.2 Organizational Challenges

When the study was first planned, we inquired with our University's Institutional Review Board. As it turned out, no IRB approval of our experimental protocol was necessary as it did not involve human subjects. Nonetheless, conducting large-scale, active vulnerability assessments within an University environment gave rise to at least two organizational challenges.

[5] http://forge.mysql.com/wiki/MySQL_Proxy

First, our responsibility as a research University to involve undergraduate and masters students in our research can conflict with our ethical responsibility of prudently limiting access to sensitive information to a small group of core researchers. The bulk of the engineering was done by PhD students and long time members of the research group. Towards the end of the development process, outside students who showed interest in our work were invited to participate as single-semester research project students. This is perhaps an unique feature of working within an University environment. While we had no reason to distrust project students, we also did not feel it was appropriate to grant them access to sensitive vulnerability data without proper training in our experimental protocol. To address this, another scanner cluster was cloned, wiped of sensitive data, and moved onto an isolated development network. There, the project students acclimated themselves to the codebase and workflow. After a month of preparation, students were individually granted access to the production environment once they have demonstrated proficiency and a full understanding of our policies and experimental protocol.

Second, an *active* vulnerability assessment can only be responsibly conducted with proper support and mitigation staff. Should our scanner misbehave or cause any complications for the target networks we are scanning, such problems must be acknowledged and addressed in a timely manner. To keep the lines of communication open between our research group and counter-parties who are interested in our activities, we publicized a mailing list on our official project page which forwarded all incoming emails to the entire research group as well as our University's network security operations group. With the gracious help of the University security group, we were able to quickly respond to, and in almost all cases, address opt-out requests and requests for information within 24 hours. Section 3.3 discusses the surprisingly low volume of responses we received regarding our experiment.

### 3.3 Social Implications of Large-Scale Vulnerability Research

**Proper Disclosure and Dissemination of Data:** Perhaps the biggest challenge of our research project is the responsible dissemination of vulnerability data to the appropriate organization. As we pointed out in our recent publication [1], several ISPs operating in Asia, particularly South Korea, operated the majority of vulnerable embedded devices discovered by our scanner. This was in fact welcomed news. Hundreds of thousands of vulnerable embedded devices are owned and centrally managed by three private businesses. Thus, a significant portion of all trivially vulnerable embedded devices can be fixed with the cooperation of possibly a handful of people. However, reaching and convincing the right group of people proved quite difficult. Our research group had neither the experience nor the social network to manage the dissemination of vulnerability data to counter-parties throughout the world. Instead, we partnered with Team Cymru[6] to accomplish this task. Each time a major nexus of vulnerable devices is discovered, a member of our research group reaches out to Team Cymru with the appropriate vulnerability report. We are currently attempting to reach several major organizations regarding our findings. We are hopeful that our longitudinal study will reveal measurable impacts on the global vulnerable device population as a result of our collaborative efforts.

**Responses to our Scanning Activities:** As shown in our recent publication [1], we received far fewer responses from the internet regarding our study than first anticipated. Over the last seven months, our public project page served 329 unique visitors, most of which were obvious automated scans for vulnerable web services. At the time of writing, we have handled approximately 70 conversations with various network operators regarding our activities, half of which were requests for vulnerability data on their own networks. When we first began our scanning activity, we anticipated a large initial volume of inquires and opt-out requests followed by an eventual decline. In reality, we still see approximately the same volume of inquires during each iteration of the scan one year after our initial scan. This may be an indication that most networks no longer bother with scan detection, or they may ignore scanning since it is so commonplace.

In order to handle the occasional opt-out requests from network operators, we created an automated console which allowed any member of the research group to remove arbitrary network ranges from all scanner nodes. We also implemented an emergency shutdown plan which the Columbia network operators can invoke as an option of last resort. This plan essentially involved powering down the scanner nodes, then disconnecting our production network gateway router from the campus network. We have not yet had to invoke this plan.

## 4 Lessons Learned

### 4.1 Measure Three Times, Cut Once.

**Proper sizing** of the experimental endeavor, followed by **sufficient investment** in its infrastructure is a message which has been clearly highlighted by our various technical missteps and failures. In retrospect, the creation of a

[6] http://www.team-cymru.org/

clean, scalable, efficient and highly available codebase could have saved large amounts of time reimplementing bottlenecks and hunting bugs. It is tempting to hastily produce a proof of concept system to show that an idea can stand on its own legs. However, when the commitment is made to scale an experiment up to a global level, reliability and scalability should be carefully factored into the design requirements of the project infrastructure. We found that aggressively attacking bottlenecks and fixing them the ”right way” often involved bigger initial investments, but quickly paid off as the experiment went on.

### 4.2 Do Not Reinvent the Wheel.

Leveraging existing code such as Apache, MySQL and pysqlpool increased the performance and reliability of our scanner design. Building on top of standard software platforms and protocols also greatly simplified the complexity of our codebase. Inevitably, some software packages will be buggy, unstable and inappropriate for the production environment. However, we benefitted greatly from incorporating our code into existing software platforms.

### 4.3 Infrastructure Counts.

Ultimately, the capability, reliability and security of our technical infrastructure allowed us to engineer and operate our vulnerability scanner in a responsible fashion. It is trivial to use *nmap* to find potentially vulnerable routers on the internet. The existence of a fortified production network which can only be accessed via IPSec VPN, the existence of a non-sensitive development environment and our ability to track and protect sensitive data as it moves through the experimental process has enabled us to carry out our research with reasonable assurance that our servers are locked down and monitored, and that sensitive data can not be exfiltrated due to a failure in our experimental protocol or negligence. A large amount of effort was invested in building a fortified, high performance research environment. This effort created the bedrock on which the vulnerability assessment and several other sensitive ongoing projects are built. Without this solid foundation, our vulnerability assessment would not have been possible. Appendix C illustrates our network and computing infrastructure.

### 4.4 (Human) Infrastructure Counts.

Without the cooperation and help of Columbia University IT, our Network Security group and the IT administrators of our CS department, the vulnerability assessment most likely would not have been conducted. Thus, relationships with related groups who serve to ensure proper management and proper adherence to policy is of paramount importance, especially when the project is regarded as sensitive.

## 5 Conclusion

In this paper, we described the engineering and operations of a large-scale embedded device vulnerability scanner infrastructure. Using this system, we monitored large portions of the internet, and were able to identify over 1.1 million publicly accessible embedded devices as trivially vulnerable, or devices which have well-known root passwords configured in their administrative interface. The scanner infrastructure was developed over a period of one year and underwent several major design iterations. We highlighted key performance bottlenecks, design missteps, technical failures and lessons learned from our experiences. We also discussed important non-technical challenges of conducting sensitive security research within an open University environment as well as the social implications of large scale *active* vulnerability assessment efforts. We believe that large-scale quantitative analysis of real-world vulnerability data is crucial to understanding and mitigating serious emerging threats on the internet. While such research activities can yield pivotal insights into the reality of internet insecurity, they can also have serious real-world impact if not conducted properly. Therefore, it is crucial that researchers size the effort properly, build the proper infrastructure (both organizational and technical) and clearly define experimental protocols with security and safety in mind before engaging in research activities which may adversely affect external parties.

## 6 Acknowledgment

We would like to sincerely thank Joel Rosenblatt and the Columbia Network Security group for their help and support. We would also like to thank Team Cymru for making the dissemination of our vulnerability data to the proper parties possible. We would also like to thank all the past project students whose hard work made our study possible. They are: Fengwei Zhang, Keng He, Pratap Prahbu, Yuan Jochen Kang and Baibhav Pendse. This work is supported by The Office of Naval Research under grant N000140910757 and the Defense University Research Instrumentation Program (DURIP) equipment grant FA9550-09-1-0389.

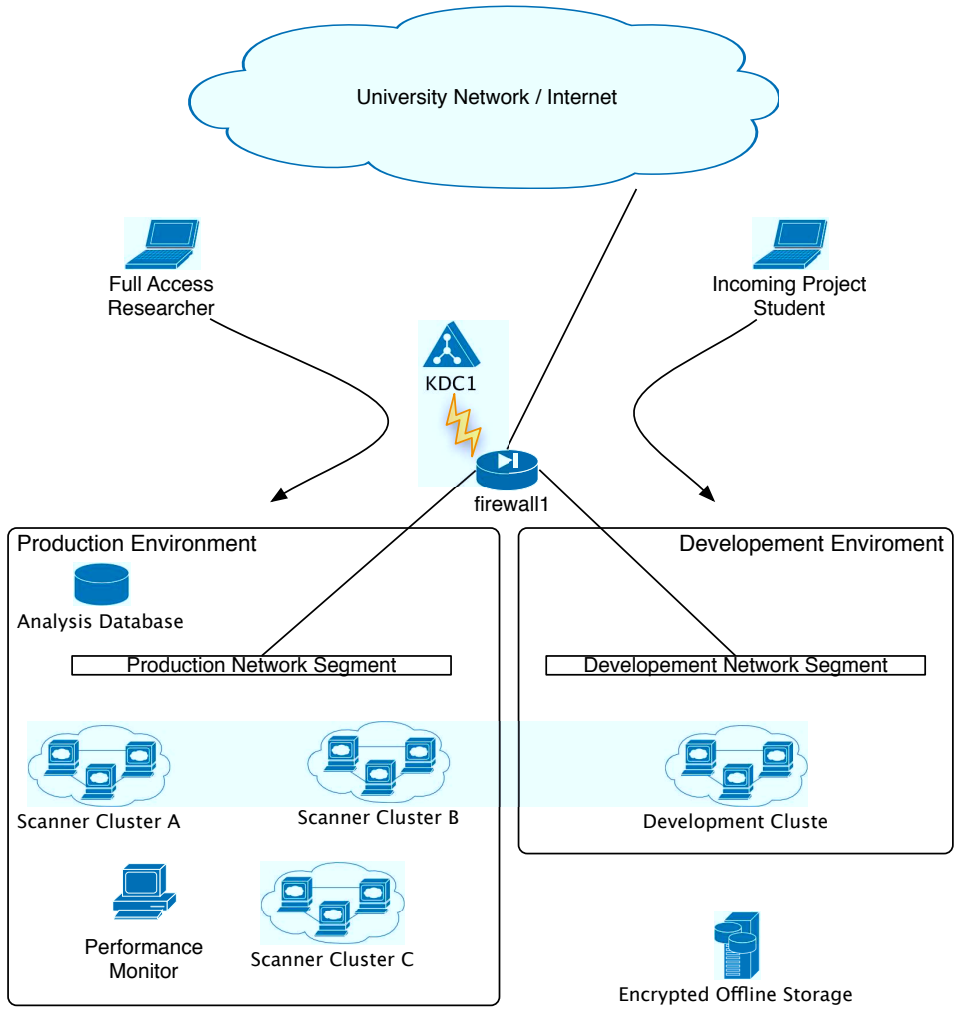
## References

1. Ang Cui and Salvatore J. Stolfo. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In Carrie Gates, Michael Franz, and John P. McDermott, editors, *ACSAC*, pages 97–106. ACM, 2010.

.

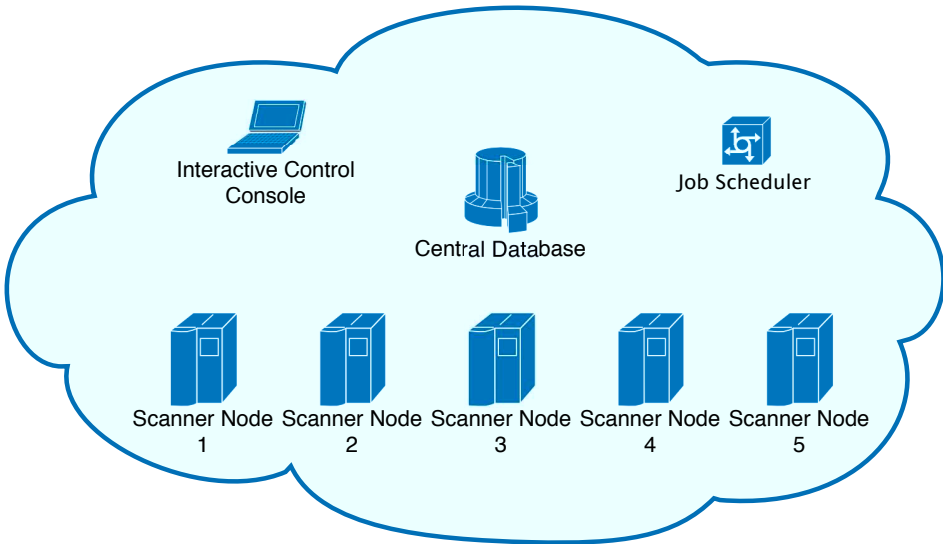
**APPENDIX**

## A Network Topology of Scanner Infrastructure




**Hacktory.cs.columbia.edu**, our research infrastructure is isolated from the University network and internet by a Cisco ASA5550 firewall. Access to either the production or development environment is only possible through IPSec VPN. Users are authenticated through the VPN by the firewall via TACACS to our central kerberos server, kdc1. A *full access researcher* can access both the production and development networks while a *incoming project student* can only access the development environment. The two networks are isolated from each other. No traffic is permitted between the two segments. The production environment houses three active scanner clusters (see Appendix B, the monitoring station, and an analysis database where all sensitive data is stored while vulnerability reports are being generated. The development environment contains a single standalone host containing all components of the scanner cluster. No sensitive data is stored on this host. Lastly, vulnerability data is periodically purged from our database and stored on an IronKey.

## B Components of a Single Scanner Cluster



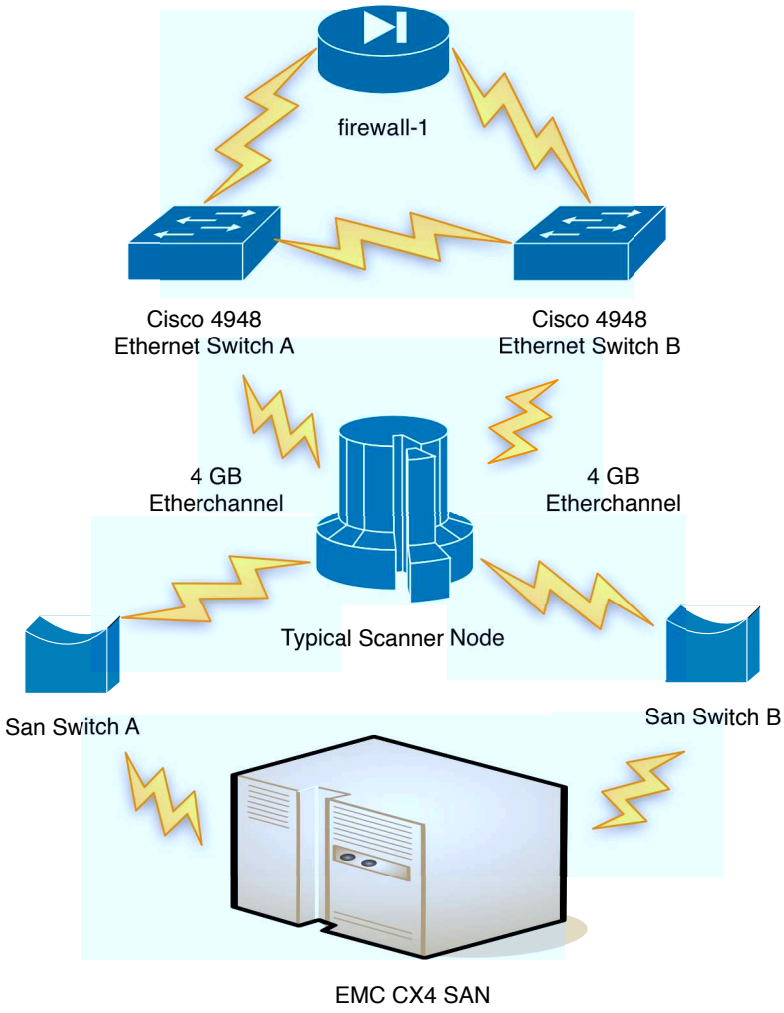

A Single Scanner Cluster

A single scanner cluster consists of:

1. A central MySQL database.
2. 5 scanner nodes.
3. 1 job scheduler node.
4. 1 host running the interactive control console.

The job scheduler node periodically monitors the system utilization of each of the five scanner nodes, allocating scan jobs accordingly. Each scanner node is configured to run no more than 32 *nmap* instances concurrently. All scan and vulnerability information is stored within the central database. Once a scan iteration is complete, the central databases of all three scanner clusters are merged onto the main analysis database, where reports are generated.

## C Physical Topology of a Single Scanner Node




The hacktory infrastructure was designed with performance and availability in mind. Scanner nodes are VM instances living on one of 10 servers running VMWare vSphere. Each vSphere node is dually connected to a pair of Cisco 4948 switches via 4GB ether-channel. Each node is also dually connected to a pair of 8GB SAN switches, which connects the server to an EMC CX4 SAN chassis containing approximately 15TB of redundant storage. For IP connectivity, the Cisco 4948 switches are then connected to our Cisco ASA5550 gateway firewall.