REDACTED VERSION

Security Analysis of Georgia's ImageCast X Ballot Marking Devices

Expert Report Submitted on Behalf of Plaintiffs Donna Curling, et al.
Curling v. Raffensperger, Civil Action No. 1:17-CV-2989-AT
U.S. District Court for the Northern District of Georgia, Atlanta Division

Prof. J. Alex Halderman, Ph.D.

With the assistance of Prof. Drew Springall, Ph.D.

July 1, 2021

Contents

1 Overview 4
1.1 Principal Findings 4
1.2 Main Conclusions 6
1.3 Organization of this Report 8
2 Georgia's Voting Equipment 9
2.1 Certification and Testing History 9
2.2 ImageCast X Hardware and Software 9
2.3 ImageCast Precinct Hardware and Software 11
3 Threats to Georgia Elections 12
3.1 Threat Actors 12
3.2 BMD Ballot Manipulation Attacks 13
4 Methodology and Testing Process 17
4.1 Testing Methodology 17
4.2 Materials Examined 18
4.3 Testing Process 18
4.4 Proof-of-Concept Attacks 19
5 Manipulating Ballots via the ICX Printer 20
5.1 Decoding Ballot QR Codes 20
5.2 Defeating QR Code Authentication 21
5.3 Demonstration Hardware-Based Attack 23
6 Attacks Against ICX Smart Cards 26
6.1 Extracting Election Secrets from Poll Worker Cards 28
6.2 Forging Technician Cards to Install Malware on any ICX 29
6.3 Creating "Infinite" Voter Cards 30
7 Constructing ICX Malware 32
7.1 Overview of the Approach 32
7.2 Obtaining the Real APK 33
7.3 Decompiling and Reverse-Engineering 33
7.4 Modifying the ICX App to Change Votes 34
7.5 Defeating Applicable Defenses 35
7.6 Conclusions 38
8 Installing Malware Locally 39
8.1 Attaching USB Devices to the ICX 39
8.2 "Escaping" the ICX App 41
8.3 Accessing a Root Shell via the Built-In Terminal App 43
8.4 Manual Malware Installation Process 43
8.5 Automating Malware Installation 44
8.6 Local Malware Installation using a Forged Technician Card 45
8.7 Local Malware Installation via Android Safe Mode 46
9 Installing Malware Remotely 48
9.1 ICX Election Definitions 48

9.2 Directory Traversal Vulnerability 50
9.3 Arbitrary Code Execution as Root 50
9.4 Installing Malware from the Election Definition File 51
9.5 Defeating Security Precautions More Easily 52
9.6 Conclusions 53
10 Manipulating Logs and Protective Counters 54
10.1 Vulnerable Storage Design 54
10.2 Manual and Automated Modification 55
11 Weaknesses in the ICP Scanner 56
11.1 The ICP Accepts Photocopied Ballots 56
11.2 A Dishonest Poll Worker with Access to the ICP Memory Card can Deanonymize All Voted Ballots 56
11.3 Installed Tamper-Evident Seal could be Bypassed or Defeated . . . 57
Expert Qualifications 60
References 61
Exhibit A: October 2020 Software Update Instructions 67
Exhibit B: Georgia Logic and Accuracy Procedures 78
Exhibit C: Pro V&V Field Audit Report 90

REDACTED VERSION

1 Overview

In 2020, Georgia replaced its insecure, decades-old DRE voting machines with new ballot scanners and ballot marking devices (BMDs) manufactured by Dominion Voting Systems. Although the same BMDs are used for accessibility in parts of approximately 15 other states, Georgia is unique in using them statewide as the primary method of in-person voting [89]. This unusual arrangement places potentially malicious computers between Georgia voters and their paper ballots. In contrast, in most of the United States, voters mark paper ballots directly by hand, and BMDs are reserved for those who need or request them [87]. Georgians who vote at a polling place generally have no choice but to use the BMDs.

All voting systems face cybersecurity risks. As the National Academies of Sciences, Engineering, and Medicine recently concluded "[t]here is no realistic mechanism to fully secure vote casting and tabulation computer systems from cyber threats" [58]. However, not all voting systems are equally vulnerable. Curling Plaintiffs contend that Georgia's universal-use BMD voting system is *so insecure* that it violates voters' constitutional rights.

To assist the Court in understanding the risks that the system creates, Curling Plaintiffs asked me to conduct a security analysis of the ImageCast X (ICX) BMD and associated equipment used in Georgia elections. Using an ICX provided by Fulton County, I played the role of an attacker and attempted to discover ways to compromise the system and change votes. I, along with my assistant, spent a total of approximately twelve person-weeks studying the machines, testing for vulnerabilities, and developing proof-of-concept attacks. Many of the attacks I successfully implemented could be effectuated by malicious actors with very limited time and access to the machines, as little as mere minutes. This report documents my findings and conclusions.[1]

1.1 Principal Findings

I show that the ICX suffers from critical vulnerabilities that can be exploited to subvert all of its security mechanisms, including: user authentication, data integrity protection, access control, privilege separation, audit logs, protective counters, hash validation, and external firmware validation. I demonstrate that these vulnerabilities provide multiple routes by which attackers can install malicious software on Georgia's BMDs, either with temporary physical access or remotely from election management systems (EMSs). I explain how such malware can alter voters' votes while subverting all of the procedural protections practiced by the State, including acceptance testing, hash validation, logic and accuracy testing, external firmware validation, and risk-limiting audits (RLAs).

The most serious vulnerabilities I discovered include the following:

1. Attackers can alter the QR codes on printed ballots to modify voters' selections. Critically, voters have no practical way to confirm that the QR codes

[1] I hereby incorporate my previous declarations as if fully stated herein. I have personal knowledge of the facts in this report and, if called to testify as a witness, I would testify under oath to these facts.

match their intent, but they are the only part of the ballot that the scanners count. I demonstrate how the QR codes can be modified by compromising the BMD printer (Section 5) or by installing malware on the BMD (Section 7).

2. The software update that Georgia installed in October 2020 left Georgia's BMDs in a state where anyone can install malware with only brief physical access to the machines. I show that this problem can potentially be exploited in the polling place even by non-technical voters (Section 8).
3. Attackers can forge or manipulate the smart cards that the ICX uses to authenticate technicians, poll workers, and voters. Without needing any secret information, I created a counterfeit technician card that can unlock any ICX in Georgia, allowing anyone with physical access to install malware (Section 6).
4. I demonstrate that attackers can execute arbitrary code with root (supervisory) privileges by altering the election definition file that county workers copy to every BMD before each election. Attackers could exploit this to spread malware to all BMDs across a county or the entire state (Section 9).
5. The ICX contains numerous unnecessary Android applications, including a Terminal Emulator that provides a "root shell" (a supervisory command interface that overrides access controls). An attacker can alter the BMD's audit logs simply by opening them in the on-screen Text Editor application (Section 10).
6. In a given election, all BMDs and scanners in a county share the same set of cryptographic keys, which are used for authentication and to protect election results on scanner memory cards. An attacker with brief access to a single ICX or a single Poll Worker Card and PIN can obtain the county-wide keys.
7. The ImageCast Precinct (ICP) scanner stores ballot scans in the order they were cast. A dishonest election worker (like that emphasized by the Defendants and their expert Michael Shamos) with just brief access to the scanner's memory card could violate ballot secrecy and determine how individual voters voted (Section 11).

Proof-of-Concept Attacks In addition to discovering and validating the vulnerabilities described above, I developed a series of proof-of-concept attacks that illustrate how vulnerabilities in the ICX could be used to change the personal votes of individual Georgia voters. I am prepared to demonstrate:

1. An attack that uses malicious hardware hidden inside the BMD's printer to alter the votes on printed ballots (Section 5).
2. Malware that runs on the BMD and alters votes while avoiding hash validation, firmware validation, and logic and accuracy testing (Section 7).
3. An automated method of installing malware by briefly unplugging the printer cable and attaching a malicious USB device (Section 8).
4. Vote-stealing malware that can be installed remotely from the EMS, by altering the BMD's election definition file (Section 9).

Mitigation Some of the critical vulnerabilities I discovered can be at least partially mitigated through changes to the ICX's software, and I encourage Dominion and the State of Georgia to move as quickly as possible to remedy them.[2] However, merely patching these specific problems is unlikely to make the ICX substantially more secure. I did not have the resources to find *all* possible exploitable security bugs in the ICX software. Once I found one that satisfied a particular adversarial objective, I usually turned to investigating other aspects of the system. It is very likely that there are other, equally critical flaws in the ICX that are yet to be discovered. Fully defending it will require discovering and mitigating them all, but attackers would only have to find one.

1.2 Main Conclusions

On the basis of the technical findings described in this report, I reach the following conclusions:

- The ICX BMDs are not sufficiently secured against technical compromise to withstand vote-altering attacks by bad actors who are likely to attack future elections in Georgia. Adversaries with the necessary sophistication and resources to carry out attacks like those I have shown to be possible include hostile foreign governments such as Russia—which has targeted Georgia's election system in the past [49]—and domestic political actors whose close associates have recently acquired access to the same Dominion equipment that Georgia uses through audits and litigation in other jurisdictions.
- The ICX BMDs can be compromised to the same extent and as or more easily than the AccuVote TS and TS-X DREs they replaced.[3] Both systems have similar weaknesses, including readily bypassed user authentication and software validation, and susceptibility to malware that spreads from a central point to machines throughout a jurisdiction. Yet with the BMD, these vulnerabilities tend to be even easier to exploit than on the DRE system, since the ICX uses more modern and modular technology that is simpler to investigate and modify.
- Despite the addition of a paper trail, ICX malware can still change individual votes and most election outcomes without detection. Election results are determined from ballot QR codes, which malware can modify, yet voters cannot check that the QR codes match their intent, nor does the state compare them to the human-readable ballot text. Although outcome-changing fraud conducted in this manner could be detected by a risk-limiting audit, Georgia requires a risk-limiting audit of only one contest every two years, so the vast majority of elections and contests have no such assurance. And even the

[2] Over the past six months, I have repeatedly offered (through Curling Plaintiffs' counsel) to meet with Dominion and share my findings, so that the company could begin developing software fixes where possible, but they have yet to take me up on this offer.

[3] I conducted similar analyses of the TS in 2006 [31] and the TS-X in 2007 [11].

most robust risk-limiting audit can only assess an election outcome; it cannot evaluate whether individual votes counted as intended.

- The ICX's vulnerabilities also make it possible for an attacker to compromise the auditability of the ballots, by altering both the QR codes and the human readable text. Such cheating could not be detected by an RLA or a hand count, since all records of the voter's intent would be wrong. The only practical way to discover such an attack would be if enough voters reviewed their ballots, noticed the errors, and alerted election officials, and election officials identified the problem as a systemic hack or malfunction; but human-factors studies show that most voters do not review their ballots carefully enough, and election officials likely would consider such reports the product of voter error. This means that in a close contest, ICX malware could manipulate enough ballots to change the election outcome with low probability of detection. In contrast, risk-limiting audits of *hand-marked* paper ballots, when used with appropriate procedural precautions, provide high confidence that individual votes are counted as intended and election outcomes are correct even if the election technology is fully compromised.
- Using vulnerable ICX BMDs for all in-person voters, as Georgia does, greatly magnifies the security risks compared to jurisdictions that use hand-marked paper ballots but provide BMDs to voter upon request. When use of such BMDs is limited to a small fraction of voters, as in most other states, they are a less valuable target and less likely to be attacked at all. Even if they are successfully compromised, attackers can change at most a small fraction of votes—which, again, creates a strong disincentive to undertake the effort and risk to change any such votes.
- The critical vulnerabilities in the ICX—and the wide variety of lesser but still serious security issues—indicate that it was developed without sufficient attention to security during design, software engineering, and testing. The resulting system architecture is brittle; small mistakes can lead to complete exploitation. Likewise, previous security testing efforts as part of federal and state certification processes appear not to have uncovered the critical problems I found. This suggests that either the ICX's vulnerabilities run deep or that earlier testing was superficial. In my professional experience, secure systems tend to result from development and testing processes that integrate careful consideration of security from their inception. In my view, it would be extremely difficult to retrofit security into a system that was not initially produced with such a process.

My technical findings leave Georgia voters with greatly diminished grounds to be confident that the votes they cast on the ICX BMD are secured, that their votes will be counted correctly, or that any future elections conducted using Georgia's universal-BMD system will be reasonably secure from attack and produce the correct results. No grand conspiracies would be necessary to commit large-scale fraud, but rather only moderate technical skills of the kind that attackers who

are likely to target Georgia's elections already possess. Unfortunately, even if such an attack never comes, the fact that Georgia's BMDs are so vulnerable is all but certain to be exploited by partisan actors to suppress voter participation and cast doubt on the legitimacy of election results.

1.3 Organization of this Report

I begin in Section 2 by providing an overview of the Democracy Suite voting equipment used in Georgia. In Section 3, I establish a threat model, including the most likely kinds of attacks and attackers facing the election system, and ways in which these attackers might attempt to manipulate BMD ballots. I then discuss my methodology and testing process in Section 4.

Next, I present my technical findings, which I organize into several parts. Section 5 explains how the barcodes on ICX ballots can be decoded and manipulated, and how such manipulation could be accomplished in the supply chain through alteration of the BMD printer hardware. In Section 6, I analyze the smart cards that the ICX uses to authenticate workers and voters, and I show numerous ways that they can be attacked to create counterfeit cards and to extract cryptographic secrets. Section 7 describes how I created malicious software that can run on the ICX and manipulate ballots while subverting Georgia's procedural defenses. In Section 8, I describe several ways that such malware could be installed on individual BMDs by attackers with temporary physical access, including by exploiting a weakness introduced in the process of installing the October 2020 BMD software update. In Section 9, I describe a remote code execution vulnerability that makes it possible to install malware over a wide area without physical access to individual BMDs. Section 10 explains how even non-technical attackers can easily manipulate the ICX's audit log and protective counters. Finally, Section 11 details security problems that I discovered in the ICP ballot scanner incidentally to my study of the ICX.

2 Georgia's Voting Equipment

As of November 2020, approximately 24 states used one or more components of the Dominion Democracy Suite voting system [88], which encompasses various models and versions of ballot scanners, BMDs, and election management system software. Georgia uses Democracy Suite version 5.5–A, including ImageCast X Prime (ICX) BMDs, ImageCast Precinct (ICP) precinct-count optical scanners, ImageCast Central (ICC) central-count optical scanners, and the Democracy Suite EMS.

My analysis focuses on the ICX BMD. In 2020, the ICX was used in parts of 16 states: Alaska, Arizona, California, Colorado, Georgia, Illinois, Kansas, Louisiana, Michigan, Missouri, Nevada, New Jersey, Ohio, Pennsylvania, Tennessee, and Washington. Although the vast majority of jurisdictions provide the ICX BMD to voters on request to assist with accessibility, Georgia is the only state to mandate ICX BMDs as the primary method of in-person voting state-wide [89].

2.1 Certification and Testing History

Democracy Suite 5.5–A is the successor to version 5.5, which was certified by the U.S. Election Assistance Commission (EAC) in September 2018 under the Voluntary Voting Systems Guidelines (VVSG) 1.0 (2005) standard [85, 86] following testing by Pro V&V, an EAC-accredited Voting System Test Laboratory (VSTL) [67]. Version 5.5–A was certified in January 2019 as a modification to 5.5. As a modification, it required only limited review, which was conducted by another VSTL, SLI Compliance [74].

Georgia entered into an agreement to purchase 5.5–A in July 2019 [34], and the Secretary of State engaged Pro V&V to evaluate it against state requirements. This evaluation was completed in August 2019 [66], and, two days later, the Secretary of State certified that the system was "in compliance with the applicable provisions of the Georgia Election Code and Rules of the Secretary of State" [33].

Over the past four years, Democracy Suite has been the subject of security testing on at least seven occasions as part of state certification processes in other states, as summarized in Table 1. In California and Pennsylvania, tests were conducted by Pro V&V and SLI, and in Texas by statutorily appointed examiners. These tests involved source code review and/or hands-on testing. Some of the tests raised serious concerns, but only Texas declined to certify the Dominion system. Based on the public test reports, it appears that none of these tests uncovered the critical security issues that I document here.

2.2 ImageCast X Hardware and Software

The ICX [25] is an Android-based touch-screen device that can be operated as either a BMD or a DRE. In Georgia, it is exclusively used as a BMD, allowing voters to mark ballots on-screen and print them to an attached laser printer.

The ICX hardware, shown in Figure 1, is a commercial off-the-shelf (COTS) Avalue HID-21V-BTX-B1R "Industrial Panel PC" [8]. On the front of the

REDACTED VERSION

Date	Version	State	VSTL	Findings	Result
Oct 2017	5.2	CA	SLI	Issues related to audit logging, passwords, anti-virus, and installation	Accept [13, 77, 78]
				Potential vulnerability related to software execution from attached USB drive	
Oct 2018	5.5	PA	SLI	Concerns regarding system hardening documentation	Reject [64]
Jan 2019	5.5–A	PA	SLI	None	Accept [64]
Jun 2019	5.5	TX	—	"concerns about whether [it] preserves the secrecy of the ballot [and] operates efficiently and accurately"	Reject [27]
Oct 2019	5.10	CA	SLI	Issues related to audit logging, passwords, anti-virus, and installation	Accept [12, 75, 76]
Jan 2020	5.5–A	TX	—	"concerns about whether [it] operates efficiently and accurately; and is safe from fraudulent or unauthorized manipulation"	Reject [28]
Jul 2020	5.10–A	CA	Pro V&V	None (source-code only)	Accept [55, 65]

Table 1: **Prior Security Testing During State Certifications.** Various versions of the ICX and ICP were subjected to forms of security testing during state certification tests in California, Pennsylvania, and Texas. Although some tests flagged concerns, only Texas declined to certified the equipment. None of these tests appear to have uncovered the critical security issues we found.

device, there is a 21.5-inch touch-screen display and a smart-card slot used for authentication. On the back, there are four externally-accessible compartments covered by plastic doors: three containing various ports and the machine's power button, and one with a battery for backup power.

The ICX I tested runs a modified version of the Android 5.1.1 ("Lollipop"). This version of Android was released in December 2015. Even at that time, the next major version of Android ("Marshmallow") had been available for months. Today, the current release is Android 11, which shipped in September 2020 [3].

Most of the ICX's functionality is provided by an Android application developed by Dominion, which I will refer to as the "ICX App". Unlike with consumer phones and tablets, the ICX App is not distributed through an "app store" (and could not be without connecting the ICX to the Internet). Instead, it is installed through a process called "side-loading", in which an Android application package (APK) file containing the software is loaded from a USB device.

The ICX App itself does not contain any election-specific information, such as races or candidates. Rather, these are loaded to the device from a USB drive before each election, in the form of an election definition file created using the Democracy Suite EMS software.

REDACTED VERSION



(a) ImageCast X [8] (b) ImageCast Precinct [23]

Figure 1: **The ICX BMD and ICP Scanner Used in Georgia**

2.3 ImageCast Precinct Hardware and Software

While not the focus of this study, I briefly examined the ICP scanner. The ICP [23], shown in Figure 1, is used to count voted ballots. It can process ballots that are produced by the ICX or those that are marked by hand. Inserted ballots are automatically pulled through the paper path, scanned on both sides, and deposited into a ballot box.

In contrast to the ICX, the ICP uses a custom hardware design. A small touch-screen display provides administrative controls and feedback to voters. A built-in thermal printer produces "poll tapes" that record vote tallies. Whereas the ICX uses standard smart cards for user authentication, the ICP uses a device called an iButton [47], which Dominion refers to as a "security key".

There are three externally-accessible compartments on the ICP, all with plastic doors that can be covered with a tamper-evident seal. A compartment on the right side contains a USB Type-A port and an RJ-45 jack. On the front are two compartments for inserting Compact Flash cards used to load the election definition and store results.

The ICP I tested runs a variant of the Linux operating system, μClinux version 20070130. μClinux is a Linux variant intended for use in embedded devices; version 20070130 was released in February 2007 [83] and is more than 14 years older than the most recent Linux version. A custom application named `cf200.sig` runs on top of μClinux and provides most of the scanner's functionality.

REDACTED VERSION

3 Threats to Georgia Elections

Georgia elections face a growing risk of attack by a range of capable adversaries, including hostile foreign governments, domestic political actors, and election insiders. Here I describe these threat actors, their capabilities, and what they are likely to seek to accomplish through technical manipulation. I also discuss strategies they could use to manipulate ballots voted using the ICX BMD.

3.1 Threat Actors

Hostile Foreign Governments. Georgia's election system continues to face a high risk of being targeted by hostile foreign governments, such as Russia, which mounted a complex campaign of cyber attacks against U.S. election infrastructure—including Georgia's—during the 2016 election [48, 49]. Hostile governments could attempt to hack Georgia's election system to achieve a variety of goals, including causing fraudulent election outcomes.

Russia and other foreign governments continue to threaten Georgia's elections today. Less than a year ago, the U.S. Intelligence Community assessed that foreign threats to the 2020 election included "ongoing and potential activity" from Russia, China, and Iran, concluding that "[f]oreign efforts to influence or interfere with our elections are a direct threat to the fabric of our democracy". These adversarial governments may "seek to compromise our election infrastructure for a range of possible purposes, such as interfering with the voting process, stealing sensitive data, or calling into question the validity of the election results." [63]

Nation-state actors are among the most well resourced and technically sophisticated adversaries, and some of the most difficult to defend against. They frequently discover vulnerabilities in widely used software with which they can compromise protected systems, and they capable of creating advanced malicious software tailored for individual high-value targets [29, 72].

Nation-state actors likely can obtain access to election equipment with which to develop attacks via physical intrusion, theft, or by purchasing it under false pretenses. They also have developed a variety of techniques for infiltrating non-Internet-connected systems, including by compromising hardware and software supply chains [15, 61, 62] and by spreading malware on removable media that workers use to copy files in and out of protected environments.[4] Such methods could be used to target the EMS systems that are used to prepare and distribute election definitions files for the ICX. The attackers could then exploit vulnerabilities I discovered to spread vote-stealing malware to BMDs throughout Georgia.

Domestic Political Actors. In addition to the threat from foreign governments, Georgia's election system faces increasing risks from domestic political actors. Politically motivated attackers might seek to directly alter individual votes and

[4] A well-known example of this ability, which is known as "jumping an air gap", is the Stuxnet computer virus, which was created to sabotage Iran's nuclear centrifuge program by attacking factory equipment that was not directly connected to the Internet [92].

thereby change the outcome of a future election through hacking. They are also likely to exploit the fact that the election system has vulnerabilities to cast doubt on the legitimacy of results or suppress voter participation.

My work demonstrates that discovering and exploiting vulnerabilities in the ICX requires only a moderate time investment from technical experts. In recent months, numerous technically-skilled outside parties have gained access [17, 68].

For example, contractors have been given unsupervised access to ICX and ICP equipment in Maricopa County, Arizona, in the context of a controversial forensic audit of the November election [14, 43]. The audit is being led by a cybersecurity firm called Cyber Ninjas, whose owner is said to promote baseless conspiracy theories that the 2020 Presidential election was hacked to defeat Donald Trump [26]. The proliferation of access to the equipment by possibly untrustworthy and politically-motivated actors and their associates has greatly increased the risk that information sufficient to attack Georgia's election system will fall into the wrong hands.

Election Insiders. As the Defendants and their expert Michael Shamos have emphasized, dishonest election insiders also pose a high risk to Georgia elections. County technicians, vendor support personnel, and poll workers need to have access to election equipment—sometimes without supervision—in order to carry out their job functions. I detail a wide variety of attacks that could be performed with such access, including infecting BMDs with malware on a wide scale.

Although discovering vulnerabilities and developing malware likely requires a degree of technical skill beyond that of most election workers, malware, once developed, can be implanted by unskilled attackers. Dishonest insiders could be recruited (or planted) by the sophisticated foreign and domestic threat actors described above to attack Georgia's voting system in this manner.

Voters. The least privileged category of attacker I consider is ordinary voters, who have brief access to the ICX within the polling place. Most voters are unlikely to have the technical expertise to develop attacks on their own, but, like election insiders, non-technical voters could be recruited by more sophisticated threat actors. I assume only that a dishonest voter has the ability to follow instructions provided by a more technical criminal. And, of course, there no doubt are many among the millions in Georgia who themselves possess the requisite technical skills to develop and implement one or more of the attacks I detail here, among others not yet identified. Under this model, I show that even typical voters could potentially infect Georgia BMDs with vote-stealing malware.

3.2 BMD Ballot Manipulation Attacks

The ICX, as used in Georgia, produces ballots like the one shown in Figure 2. They are printed on one or more sheets of letter-size paper. The ballot design uses a QR code (a kind of two-dimensional barcode) to represent the voter's selections in machine-readable form. Although the ballot also contains human-readable text

that summarizes the selected choices for each contest, Dominion scanners ignore the ballot text and exclusively count the votes that are encoded in the QR code. Voters have no practical way to read the QR codes, so they cannot verify the representation of their vote that is counted.

In later sections, I will show how attackers can manipulate ICX ballots through attacks on the BMD printer or on the ICX software. By either of these means, attackers could apply two different strategies for altering votes:

Altering only the barcodes. Attackers could cause the BMDs to print QR codes that differ from voters' selections while leaving the human-readable text of the ballot unchanged. Since voters cannot read QR codes unaided, they would be unable to detect the alterations, but, since the QR code is the only part of the ballot the scanners count, the impact would be a change to the tabulation of those individual votes affected and potentially to the election results. The only known safeguard that can rule out such an attack is to compare the human-readable text on every voted ballot to the QR codes, which Georgia has never done in any election and which does not appear to be required or anticipated for future elections.

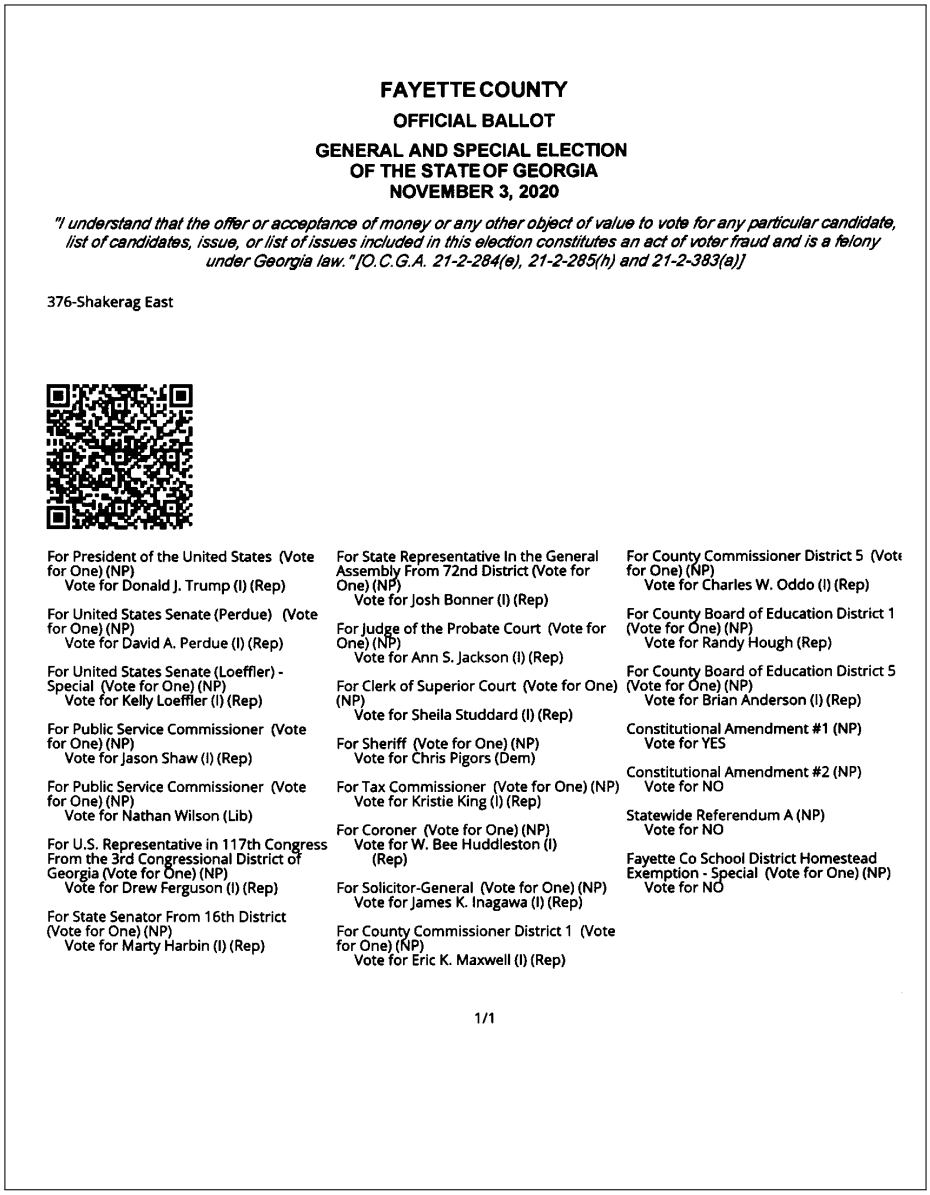
Since attackers might choose to target any race in any election, every race and every election would need to be subjected to a rigorous risk-limiting audit (RLA). Georgia rules currently require an RLA of only a single state-wide *contest* every two years [69]. In the vast majority of races—even high-profile ones, such as the U.S. Senate races in November 2020 and January 2021—the state does not audit the human-readable ballot text at all, and so it is highly likely that barcode-only attacks would go undetected.

Altering *both* the barcodes and the text. Attacks on the BMDs could also change *both* the barcode and the human-readable text on a fraction of the printed ballots, so that both represented the same set of fraudulent selections. Research shows that few voters carefully review BMD ballots [9, 54]. Consequently, when most voters use BMDs, manipulation of enough votes to change the winner of a close race would likely go undetected, and individual voters would be disenfranchised, even if the election outcome were unchanged.[5] No audit or recount could detect this fraud—not even an RLA—because all records of the voter's intent would be wrong. Pre-election or parallel testing also cannot reliably detect such cheating [80]. Even if officials did suspect that the BMDs had been attacked, there would probably be no straightforward way to determine the correct outcome and no way at all to determine each individual voter's intended vote. The only recourse might be to rerun the election.

Both attack strategies could be accomplished using the same technical methods, so attackers can choose between them depending on the contest being targeted. In contests where no audit or recount is likely, attackers can cheat

[5] I review the research concerning voter-verifiability of BMD ballots (which includes my own award-winning peer-reviewed work [9]) in a prior declaration [39, ¶ 23–33]. Data from subsequent research lends further support to my conclusions [54].

REDACTED VERSION



FAYETTE COUNTY

OFFICIAL BALLOT

GENERAL AND SPECIAL ELECTION
OF THE STATE OF GEORGIA
NOVEMBER 3, 2020

"I understand that the offer or acceptance of money or any other object of value to vote for any particular candidate, list of candidates, issue, or list of issues included in this election constitutes an act of voter fraud and is a felony under Georgia law." [O.C.G.A. 21-2-284(e), 21-2-285(h) and 21-2-383(a)]

376-Shakerag East

For President of the United States (Vote for One) (NP)
Vote for Donald J. Trump (I) (Rep)

For United States Senate (Perdue) (Vote for One) (NP)
Vote for David A. Perdue (I) (Rep)

For United States Senate (Loeffler) - Special (Vote for One) (NP)
Vote for Kelly Loeffler (I) (Rep)

For Public Service Commissioner (Vote for One) (NP)
Vote for Jason Shaw (I) (Rep)

For Public Service Commissioner (Vote for One) (NP)
Vote for Nathan Wilson (Lib)

For U.S. Representative in 117th Congress From the 3rd Congressional District of Georgia (Vote for One) (NP)
Vote for Drew Ferguson (I) (Rep)

For State Senator From 16th District (Vote for One) (NP)
Vote for Marty Harbin (I) (Rep)

For State Representative In the General Assembly From 72nd District (Vote for One) (NP)
Vote for Josh Bonner (I) (Rep)

For Judge of the Probate Court (Vote for One) (NP)
Vote for Ann S. Jackson (I) (Rep)

For Clerk of Superior Court (Vote for One) (NP)
Vote for Sheila Studdard (I) (Rep)

For Sheriff (Vote for One) (NP)
Vote for Chris Pigors (Dem)

For Tax Commissioner (Vote for One) (NP)
Vote for Kristie King (I) (Rep)

For Coroner (Vote for One) (NP)
Vote for W. Bee Huddleston (I) (Rep)

For Solicitor-General (Vote for One) (NP)
Vote for James K. Inagawa (I) (Rep)

For County Commissioner District 1 (Vote for One) (NP)
Vote for Eric K. Maxwell (I) (Rep)

For County Commissioner District 5 (Vote for One) (NP)
Vote for Charles W. Oddo (I) (Rep)

For County Board of Education District 1 (Vote for One) (NP)
Vote for Randy Hough (Rep)

For County Board of Education District 5 (Vote for One) (NP)
Vote for Brian Anderson (I) (Rep)

Constitutional Amendment #1 (NP)
Vote for YES

Constitutional Amendment #2 (NP)
Vote for NO

Statewide Referendum A (NP)
Vote for NO

Fayette Co School District Homestead Exemption - Special (Vote for One) (NP)
Vote for NO

1/1

Figure 2: **BMD Ballot, Showing QR Code.** This is a real ballot cast in Fayette County during the November 2020 election, as captured by an ICP scanner. Note the small and densely printed text. Although the selected candidates are printed in human-readable form, the scanners ignore the text and exclusively count the votes encoded in the QR code, which voters have no practical means to verify.

arbitrarily by altering only the ballot barcodes. Otherwise, as long as the margin of victory is likely to be small, attackers can still change the election outcome with low risk of detection by altering both the barcodes and the ballot text on a small fraction of ballots across many BMDs.

Notably, both styles of ballot manipulation are far greater risks when BMDs are used for all in-person voters, as in Georgia, than when only a small fraction of voters use them, as in most other states. When few voters use BMDs, even changing *every* BMD ballot could only affect the outcome of contests with very narrow margins, and successful fraud would usually require cheating on such a large fraction of BMD ballots that it would likely be discovered. This makes the BMDs an unappealing target and reduces the risk that they will be attacked at all. In contrast, Georgia's universal-use BMDs would be a very appealing target, since they expose all in-person voters to potential ballot manipulation.

In sections that follow, I demonstrate ballot manipulation attacks in several contexts: via attacking the BMD's printer, by installing malicious software onto BMDs with physical access, and by spreading malware to all BMDs across wide areas from central locations. In my implementations of these attacks, I alter only ballot barcodes, but altering both the barcodes and the ballot text would require only straightforward changes to the malicious code.

REDACTED VERSION

4 Methodology and Testing Process

4.1 Testing Methodology

Security testing is a widely-recognized best practice, especially for critical systems. My tests of the Georgia voting equipment applied a form of the security testing methodology known as Open Ended Vulnerability Testing (OEVT) [59], which was recommended for voting system testing by the U.S. EAC's Technical Guidelines Development Committee. As described by in a report by the National Institute of Standards and Technology (NIST) [59]:

> The goal of OEVT is to discover architecture, design and implementation flaws that have crept into the system which may not be detected using systematic functional, reliability, and security testing and can be exploited to change the outcome of an election, can provide erroneous results for an election, can cause denial of service, can compromise [the] secrecy of [the] vote, or can compromise [the] security audit log.

OEVT pursues this by having testers play the role of an adversary and attempt to compromise the system. They engage in an iterative process in which they: (1) work to understand how the system functions through observation, review of documentation, hands-on experimentation, and reverse-engineering; (2) generate hypotheses about how security might be compromised; and (3) validate those hypotheses through experiments. Forms of OEVT have been applied in comprehensive voting system security reviews commissioned by the Secretaries of State of California [10] and Ohio [57], and in numerous research studies of deployed election equipment [42].

Since I was provided with access to equipment but not to the software source code, I applied a "black-box" testing approach, in which I relied entirely on reverse-engineering and experimentation to discover vulnerabilities. Though less efficient than "white-box" testing (i.e., analysis conducted with access to source code), a black-box approach has the advantage of more closely mimicking the capabilities of the largest number of potential attackers. That is, any vulnerabilities I found could also be discovered by real attackers without access to Dominion's source code.

OEVT methodology has important limitations. It is highly dependent on the skill, resources, and experience of the testers, and also on good luck. I was fortunate that many of the observations that I decided to pursue through detailed testing proved to be productive, but there were many other observations that I decided not to pursue, and I almost certainly overlooked clues to other important weaknesses. Due to time and resource constraints, once I found one way to accomplish an adversarial objective (e.g., installing malware remotely), I usually moved on to another goal, rather than attempting to find all ways of accomplishing it. For these reasons, I stress that while my methodology is effective for discovering and proving the existence of security problems, the vulnerabilities I uncovered are almost certainly not the only such problems affecting the equipment I studied.

4.2 Materials Examined

I received access to Georgia election system components that were provided to Plaintiffs by Fulton County in compliance with this Court's orders. The major components were an ImageCast X Prime (ICX) BMD, serial number 1910250020, running software version 5.5.10.30, and an ImageCast Precinct (ICP) ballot scanner, serial number AAFAJKL0064, running software version 5.5.3-0002.

In addition, I received a Poll Worker Card and a Technician Card (but not a Voter Card) for the BMD, together with the PINs for both cards. I was also provided a Centon USB drive containing an ICX election definition file for a mock election used for testing and training. The scanner came with two compact flash cards prepared for use in the same mock election, as well as an iButton Security Key and passwords for operating administrative functions.

Fulton County did not provide the off-the-shelf laser printer used in conjunction with the BMD. Instead, Plaintiffs acquired a unit of the same model, an HP LaserJet M402dne, from a commercial source. I was not provided access to the Democracy Suite EMS software.

Analysis of the ICX's audit logs indicated that the unit had been previously tested but had not been used in an election. The unit provided to us had two tamper-evident seals on one of its four compartments. I opted not to remove the seals or open the device's chassis during the course of this investigation.

In June 2021, I received access to further election system data. State Defendants provided Plaintiffs copies of data sent to the Secretary of State by Georgia counties following the November 2020 and January 2021 elections. Although this data was significantly incomplete, many counties returned Election Packages—backups created by the Democracy Suite EMS—which I briefly examined to ascertain how counties typically configured their BMDs. I also extracted the ICX election definition file used by Fulton County in the November 2020 election, which I used for further tests.

Five of the county data sets provided by State Defendants contained copies of the installation file for the October 2020 ICX software update, version 5.5.10.32. The presence of this file may indicate that the counties returned data to the Secretary on the same USB drives that they used to receive or distribute the software update, without first wiping the device.

I completed initial testing with the original ICX software version, 5.5.10.30. I later used the update installation file and the official installation instructions (attached as Exhibit A) to upgrade to version 5.5.10.32 and reverify the findings. Except for a vulnerability reported in Section 8 that is a *consequence* of Georgia updating the software, both versions exhibited the same vulnerabilities.

4.3 Testing Process

Throughout the analysis, the voting equipment was maintained in a secure facility in Atlanta. Due to the COVID-19 pandemic, I performed most testing remotely via videoconference, directing on-site work by my assistant, Dr. Springall. I have

personally verified all findings in this report, and all opinions and conclusions are solely my own.

I began working with the Fulton County equipment on September 4, 2020 and conducted 11 work sessions through June 25, 2021. Between sessions, I reviewed documentation, analyzed collected data, performed reverse-engineering, and prepared tests, so as to make efficient use of time with the equipment. The entire process took approximately twelve person-weeks of effort.

4.4 Proof-of-Concept Attacks

For some of the ICX's vulnerabilities, I prepared proof-of-concept attacks that demonstrate how the problems could be exploited by a malicious actor. Such proof-of-concept "exploits" are widely recognized in the security field as a means of proving that a system suffers from a particular technical flaw. However, they are *not* intended to be exemplars of "weaponized" attacks, such as a sophisticated adversary would seek to deploy.

As such, some of these demonstrations have minor imperfections (such as delays or small visual glitches) that a real attacker could remove with moderate investments in engineering and testing. I also built the proof-of-concept vote-stealing attacks to be demonstrated with a particular election definition, rather than to work generally in any election as a real attacker would do. Implementing these refinements would not require the discovery of any further vulnerabilities, so I chose instead to use my limited resources to analyze additional aspects of the equipment.

REDACTED VERSION

5 Manipulating Ballots via the ICX Printer

In this section, I show how BMD-printed ballots can be manipulated without any malicious modifications to the ICX hardware or software. I first examine the structure of the ballot QR codes, show that they are unencrypted, and explain how they can be fully decoded. Next, I show that weaknesses in the QR code design make it possible to manipulate ballots in spite of a security mechanism intended to authenticate the QR codes. Finally, I demonstrate that attackers can automatically manipulate ballots cast on the ICX with no access to the BMD itself, by instead attacking the attached off-the-shelf laser printer. I show how such an attack can be implemented by adding concealed malicious hardware to the printer, which could be accomplished as part of a supply-chain attack.

5.1 Decoding Ballot QR Codes

Dominion's documentation claims that the QR codes are encrypted [19, § 2.6.1.1], and, at least as recently as January 2021, Secretary of State Chief Operating Officer Gabriel Sterling has repeated this claim to the media as a security feature of Georgia's voting system [91]. In actuality, as I testified last year, no part of the QR codes is encrypted [40, ¶ 37–40]. While voters have no practical way to read or verify the votes encoded in the QR codes, they can be decoded by attackers and can be replaced or manipulated to steal voters' votes.

Although the QR codes are not encrypted, they use a data format this is incompatible with most off-the-shelf barcode reader software. A QR code can encode data in several data formats: numeric, alphanumeric, or byte mode [30]. Byte mode can encode arbitrary data, but QR code readers typically interpret the byte sequence as UTF-8 or Latin-1 encoded text. If an application needs to represent arbitrary binary data in a QR code, the recommended practice for ensuring compatibility is to encode the data using characters available in alphanumeric mode (e.g., Base45 encoding) [30]. However, the ICX QR codes appear to be designed only for compatibility with Dominion scanners. They encode binary data in byte mode, and the data typically begins with a byte with value zero. As a result, most QR code reader software either fails to read them because the data does not represent valid UTF-8 or Latin-1 characters or incorrectly treats the zero byte as the end of a null-terminated string.

In August 2020, my research group tested reading the ICX QR codes with a variety of publicly available barcode reader apps for Android and iOS devices. At the time, only one app we tested was able to read them correctly (Scandit Barcode Scanner for iOS [73]), and later versions of that same app no longer do. Several publicly available programming language libraries for reading QR codes had similar compatible problems when used with their default settings. However, we found that we could correctly decode the data using recent versions of the open-source ZBar barcode reader library by setting the `ZBAR_CFG_BINARY` option to force the software to emit the data as raw bytes. For example, using ZBar version 0.23.90,[6] ICX QR codes can be decoded with the command:

[6] Available at https://github.com/mchehab/zbar/releases/tag/0.23.90.

```
zbarimg --quiet --raw -Sbinary ballot.png | hd
```

After extracting the raw data from the QR codes, my research group reverse-engineered the binary data format. To do so, we examined Dominion's ImageCast Remote Accessible Vote-By-Mail (RAVBM) software, a web-based app that generates a ballot with a similar QR code for printing and returning through the mail [24]. (Since ImageCast Remote runs in the voter's browser, the JavaScript source code that it uses to generate the QR codes is publicly visible.) We also examined ICX QR codes from publicly available ballot scans from a variety of elections and determined that they used the same data format. Through this process, we created a computer program (`dvsqrtool.py`) that interprets and unpacks all data fields in the ICX QR code.[7]

The decoded data contains the voter's selections, write-in votes, and ballot metadata. No encryption key is necessary to extract this data, which demonstrates that the QR code is not encrypted. The data structure represents voter selections as a series of binary digits (ones and zeros), as shown in Figure 3. Each digit corresponds to one of the available candidates, typically in the same order that the contests and choices are displayed on the BMD's screen or on the equivalent hand-marked ballot. A `1` signifies that the candidate was selected, and a `0` signifies that the candidate was not selected. Therefore, with knowledge of the ballot design, the selected choices can be readily extracted from the QR code.

5.2 Defeating QR Code Authentication

Issue: *ICX QR codes are not protected against "replay" attacks, so copies of valid QR codes will be accepted as genuine.*

As an authentication mechanism, the QR code contains a cryptographic message authentication code (MAC) computed using the HMAC-SHA256 algorithm. A MAC is a value (a number) calculated based on an input and a secret key. Without knowing the key, it is infeasible to calculate the correct MAC for a modified input. In a given election, the ICX and ballot scanner have copies of the same key. Whenever an ICX generates a QR code, it uses this key to calculate the MAC of the ballot data. When a scanner reads the QR code, it extracts the data, repeats the MAC calculation using its copy of the key, and verifies that the MAC value it calculated matches the MAC in the QR code. Under the assumption that an attacker cannot discover the secret key,[8] this arrangement allows the scanners to confirm that the data in the QR code really was generated by an ICX and was not subsequently modified.[9]

[7] This work was completed in connection with my research at the University of Michigan before Plaintiffs received the Dominion equipment and without use of confidential information. Therefore, I consider it to be outside the scope of the Protective Order.

[8] In fact, the MAC key is not well safeguarded. I show in Section 6.1 that the key used throughout a county can be easily extracted from any Poll Worker Card, given brief physical access to the card and its PIN. It can also be extracted from an ICX after L&A testing by escaping kiosk mode using the techniques in Section 8.

[9] A MAC is very different cryptographic algorithm than a digital signature, although Defendants' experts have repeatedly confused the two [40, ¶37–39]. Both are sometimes

REDACTED VERSION

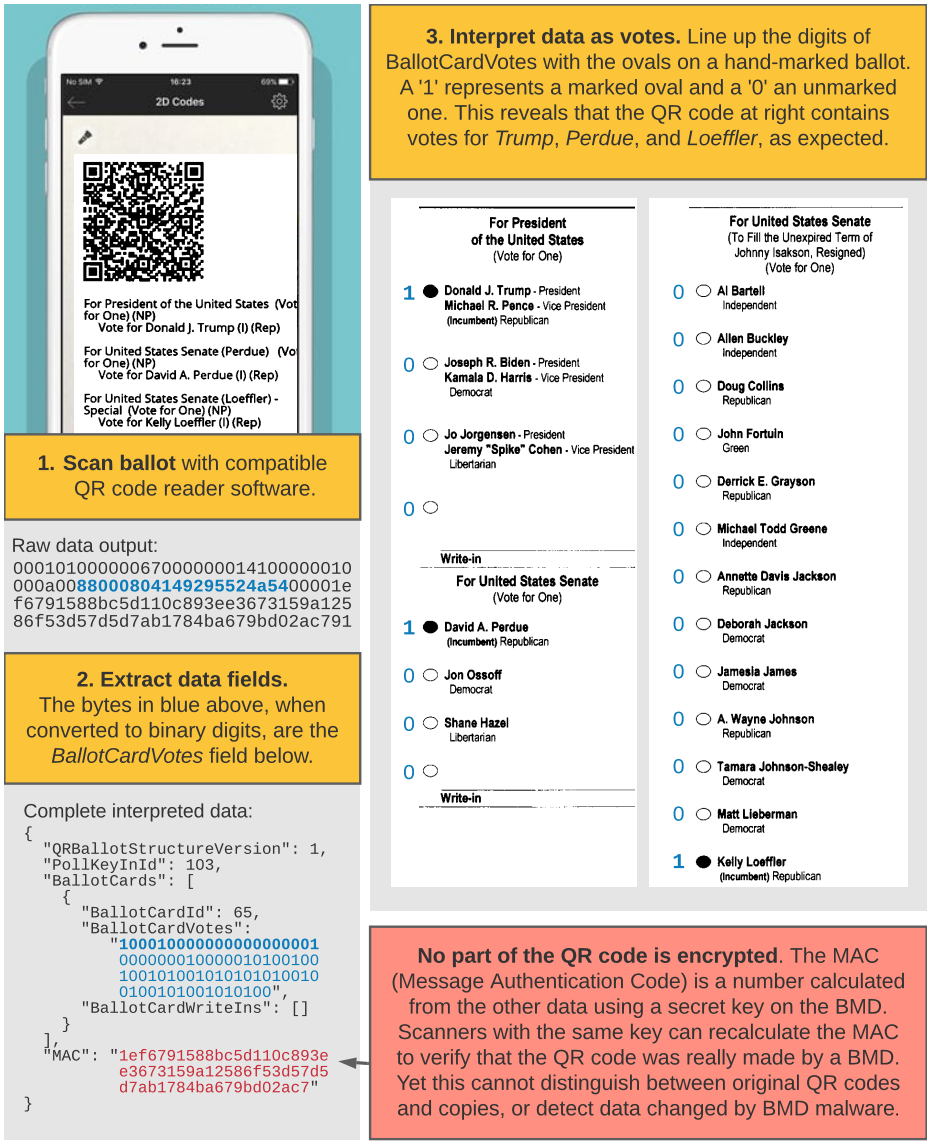


Figure 3: **Decoding the QR Code.** Using the procedure illustrated above, the QR code from Fig. 2 can be fully decoded. No secret information is required, because the QR code is not encrypted. Although the data includes a MAC, the design does not protect against duplicated QR codes or malware running on the BMD.

Despite this use of a MAC, attackers can manipulate ICX QR codes through several means to alter recorded votes or cast fraudulent votes. The ICX QR code design as used in Georgia has a serious weakness: the codes do not contain a serial number or other unique identifier, so, for a given ballot design, all QR codes that contain identical votes are indistinguishable, including having identical MACs. As a consequence, there is no mechanism for detecting *duplicate* QR codes. This enables two important attacks:

Copying Ballots A copy of a genuine ICX ballot will be indistinguishable from a second genuine ICX ballot with the same votes. In tests, the ICP accepted ballots copied using an office photocopier (see Section 11.1). This could allow a variety of ballot-box stuffing attacks.

Replay Attacks Although the MAC prevents attackers who do not know the secret key from generating new valid QR codes, they can still substitute other valid QR codes they have seen before. In a "replay" attack, attackers observe genuine printed ballots and save copies of QR codes with votes they favor. They then alter ballots with votes they disfavor by replacing the QR codes with the ones they have saved. Since the QR codes on the altered ballots contain valid MACs, the scanners accept them as genuine, even though they are duplicates. I demonstrate this style of attack and discuss the implementation details below.

5.3 Demonstration Hardware-Based Attack

An attacker can implement a fully-automatic ballot manipulation attack without tampering with the ICX itself in any way, by instead targeting the laser printer attached to the BMD. Georgia's BMDs use off-the-shelf HP LaserJet M402dne printers connected via a USB cable. Like most modern printers, they contain capable embedded computers that run complex, field-updatable software. By modifying the printer's software or hardware—or even by hiding tiny malicious hardware in a modified USB cable [38, 60]—an attacker can arbitrarily change what the printer prints. This can be employed to alter the ballot QR codes (alone or in conjunction with ballot text) and steal votes.

Since the printer is an off-the-shelf device, it is likely to receive less security scrutiny from officials than the ICX, even though attacks on the printer could be equally consequential. Attackers could potentially compromise the printers at any time during the lifecycle of the voting system, including before they are delivered (in the supply chain), while in storage, or during transport to or from polling places.

I developed a proof-of-concept attack to illustrate these risks. It consists of hardware hidden by the attacker inside the printer's housing that manipulates

used to verify data integrity. For purposes here, the most important difference is that anyone who has the key needed to verify a MAC can also *forge* valid MACs for any data they choose. In contrast, the information needed to verify a digital signature can be widely distributed or even made public without jeopardizing its security.



Figure 4: **Demonstration Malicious Hardware.** I developed a hardware-based attack that modifies data sent from the ICX to the printer, altering ballot QR codes to change recorded votes. The attack device (the two red modules seen in the right photo) is completely hidden inside the printer's plastic housing. Similar malicious hardware could be added in the supply-chain or while in storage.

the data sent from the ICX to the printer. I demonstrated an early version of the attack during the September 2020 hearing, after having access to the ICX for only about one week. To implement the attack, I used a pair of Raspberry Pi Zero W devices. These are small (approximately 1 × 2.5 inches), self-contained computers with WiFi and Bluetooth radios that are capable of simulating a USB device or host system. They are widely available for a cost of about $10.

In my attack implementation, one Pi Zero receives ballot data via the original printer cable that attaches to the BMD; I refer to this device as Pi-Input. The second Pi Zero connects to the printer itself and outputs data to be printed; I refer to it as Pi-Output. Both run the Linux operating system. A real attacker would likely integrate these functions into a single purpose-built hardware device, but I needed to split them because the off-the-shelf Pi Zeroes I used each have only a single functional USB port. Even when using two Pi Zeroes, the entire setup (shown in Figure 4) is small enough to be concealed in the empty space within the laser printer's housing, and it is low-power enough to be operated from the printer's internal power supply.

Redaction

Pi-Input is configured to behave as a USB peripheral [redacted] and runs software I developed (`in/device.py`) that simulates the printer. When connected to the BMD, the Pi Zero sends USB device descriptors and identification strings that match those of the LaserJet M402dne. This makes it indistinguishable from the real printer to the BMD's software. However, when the BMD sends data to print, Pi-Input relays that data (over a local wireless network) to the second Pi Zero, which proceeds to manipulate it.

Pi-Output connects to the real printer and operates as a normal USB host. It runs software I wrote (`out/prproxy.py`) that receives from Pi-Input the data that the BMD attempts to print. My software parses the PCL data to extract the QR code as a bitmap image, passes it to the `zbarimg` barcode reader tool to decode the QR code data, and uses my `dvsqrtool.py` tool (discussed in Section 3.2) to extract the votes.

For this attack, I assume that the adversary does not know the secret key used to compute the MAC in the QR code. Without this key, the attacker cannot modify the data in the QR code, but they can still manipulate votes by performing a replay attack, i.e., selectively copying valid QR codes from previously-seen ballots. To accomplish this, Pi-Output inspects the votes in each QR code to determine whether the attacker's preferred candidate is selected. Then:

If the attacker's candidate is selected, the device passes the ballot to the printer unmodified but saves a copy of the QR code to its internal storage.

Otherwise, the device picks one of the stored QR codes at random and substitutes it for the QR code sent by the BMD. Since the stored QR codes contained valid MACs, and the system design does not detect duplicated QR codes, these copied QR codes will be accepted as valid by the ballot scanner.

As a result, once at least one ballot has been voted for the attacker's preferred candidate, subsequently printed ballots will contain QR codes that encode votes for that candidate.[10]

For demonstration purposes, I hard-coded the target contest and favored candidate, and I programmed the device to cheat as often as possible.[11] In practice, an attacker could remotely (e.g., using WiFi or Bluetooth) select the fraction of votes to shift and which candidate in which contest should receive them. Similarly, the attacker could remotely enable or disable the cheating, thereby defeating any pre-election testing. With wireless control, the attack device could be installed in the printer once and cheat in any subsequent election.

Adding hardware to the printer is only one of several ways that attackers could manipulate ballots cast using Georgia's ICX BMDs. An easier and more powerful mode of attack would be to modify the software in the ICX itself. When I demonstrated the printer attack prototype in September 2020, I testified that software-based attacks on the ICX were very likely achievable with further analysis. This has proven to be the case. In later sections of this report, I will explain how it is possible to construct vote-stealing malware that runs entirely in the ICX, and how attackers can infect ICXs with such malware remotely throughout entire counties or even the entire state.

[10] In a realistic attack scenario, the attacker would likely choose to alter only a fraction of the ballots, so as to avoid drawing suspicion.

[11] My proof-of-concept implementation sometimes introduces a spurious delay of up to about 20 seconds before the ballot is printed. The most likely cause is a bug in the code. Having demonstrated the attack concept, I opted not to spend further resources debugging and removing the delay, and instead focused on attacking the ICX software.

6 Attacks Against ICX Smart Cards

Smart cards, such as many modern debit and credit cards, have an embedded integrated circuit chip that exchanges data with the card reader. Some smart cards are capable of storing secret data securely and performing cryptographic operations. Such cards are often used to authenticate identity in high-security applications, such as the U.S. Department of Defense (DoD) Common Access Card (CAC) that provides access to defense computer networks and systems [16].

The ICX uses smart cards to authenticate voters, poll workers, and service technicians. There are kinds of cards:

Technician Cards Service technicians are assigned a Technician Card and PIN. By inserting the card and entering the correct PIN on the screen, they can access the Technical Administrative menu shown in Figure 5a. This menu is used before each election to load new election definitions from a USB drive. It also allows more sensitive actions, such as exiting the ICX application and accessing the underlying Android operating system, from which the ICX's software can be updated or modified.

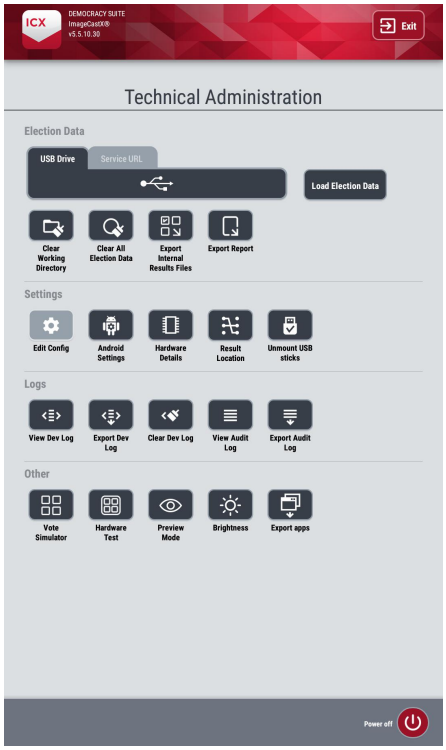
Poll Worker Cards Poll workers are assigned a Poll Worker Card and PIN, with which they can access the Poll Administration menu shown in Figure 5b. This menu allows the poll worker to open the polls using a previously-loaded election definition, manually activate a voting session (without use of a Voter Card), reset the machine's public counter, close the polls, or shut down the BMD. Poll Worker Cards are specific to each election and contain the cryptographic keys necessary to operate the ICX in the election.

Voter Cards When a voter checks in at a polling place, a poll worker uses an electronic poll book to issue them a Voter Card. The voter inserts the Voter Card into the ICX to unlock the BMD for a single voting session using the voter's assign ballot style. Upon printing the ballot, the ICX deactivates the Voter Card, preventing it from being used again, and the voter returns it to a poll worker.

I examined the communications between the ICX and the smart cards to determine the authentication protocol and evaluate its security. While I expected the BMD to use a modern cryptographic challenge-response protocol, which would render the cards resistant to cloning and forgery, the machine instead uses a simplistic and highly insecure protocol. The actual protocol is conceptually very similar to the protocol used by the Diebold AccuVote DREs, which also used smart cards. The Diebold smart card protocol was shown to be insecure by researchers as early as 2003 [53], and its vulnerabilities were documented in detail by the California Secretary of State's Top-to-Bottom Review in 2007 [11]. The ICX smart cards used today suffer from essentially all of the same vulnerabilities and some serious additional ones.

All three kinds of ICX smart cards use the same protocol, which is implemented as a series of ISO 7816-4 commands:

REDACTED VERSION



(a) Technical Administration Menu



(b) Poll Administration Menu

Figure 5: **ICX Administration Menus.** Unlocking the ICX with a Technician Card or a Poll Worker Card provides access to a variety of privileged operations.

1. The ICX attempts to open a particular file stored on the card, and the card responds with whether the file exists. Technician and Poll Worker Cards use file ID 0x[redacted], while Voter Cards use file ID 0x[redacted]. This allows the ICX to determine whether a voter card or an administrative card was inserted. **Redaction**
2. The ICX sends a password to the card to unlock the file. For Technician and Poll Worker Cards, the password is a PIN that is entered on-screen by the user. For Voter Cards, the ICX automatically sends a preconfigured PIN.
3. The card checks whether the PIN matches a value stored on the card before allowing access to the file. I assume (but did not verify) that the cards lock themselves and prevent further use if too many incorrect PINs are attempted.
4. Once the file is unlocked, the card allows the ICX to read or write to it.

The file formats for all three types of cards are simple and readily determined by inspecting the data. Each file consist of 36 records, each up to 15 bytes long, for a maximum length of 540 bytes. Their more relevant features are:

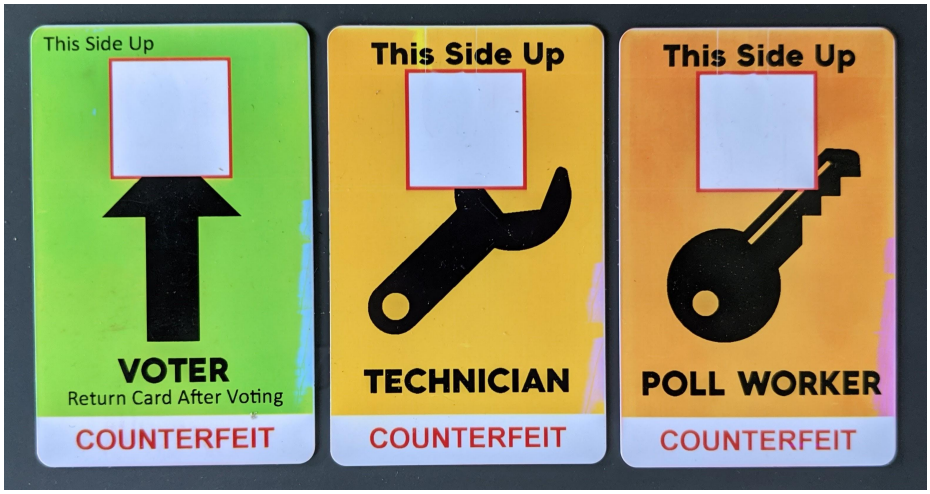


Figure 6: **Forged ICX Smart Cards.** Weaknesses in the ICX authentication protocol allow an attacker to read and forge Voter, Technician, and Poll Worker cards. I added explicit markings and allowed minor discolorations to minimize any risk of misuse, but a real attacker could create nearly indistinguishable counterfeits.

- *Technician Cards:* The first record is the value 0. Other records contain the user's name, the date and time that the card was created, and the date and time that it expires.
- *Poll Worker Cards:* The first record is the value 1. In addition to records found on the Technician Card, the file contains all of the election-specific cryptographic keys and other secrets that the ICX and scanner use for security: the admin PIN, the encryption key and IV, the MAC key, and the Election Signature (a secret value that uniquely identifies the election).
- *Voter Cards:* Records contain the ballot style, language, and accessibility mode, whether the card has been used, the date and time it was activated, and the Election Signature value (to prevent the card from being used in a different election).

I determined that an attacker can extract the data from all three kinds of cards, as well as create counterfeit cards (shown in Figure 6). In the sections that follow, I explain how these capabilities could be used for a variety of attacks.

6.1 Extracting Election Secrets from Poll Worker Cards

Issue: *Anyone with access to a single Poll Worker Card and the corresponding PIN can easily extract secret keys and other values used for securing election data throughout the county.*

The ICX smart card protocol does not authenticate the device reading the card. As a result, anyone with the correct PIN can read the data on the card in a

few seconds by simply following the protocol. I created a simple Python program (`cardutil.py`) that uses a commodity USB smart card reader and mimics the ICX's behavior, allowing us to extract the contents of the cards provided by Fulton County.

This weakness causes a serious information exposure vulnerability due to the cryptographic secrets stored on Poll Worker Cards. With access to the encryption and MAC keys from the Poll Worker Card, an attacker could decrypt or alter the ballot definitions used by the scanners and BMDs, forge ballot QR codes, or decrypt or modify election results on scanner memory cards before the results are returned to the EMS for reporting.

Poll Worker Cards and PINs are distributed to every polling place and entrusted to thousands of volunteer poll workers across the state during every major election. It would be practically impossible to ensure that none of these cards could be temporarily accessed by a malicious party.

County election databases from the November 2020 and January 2021 elections shows that Georgia counties use the same cryptographic keys county-wide for each election. This means that if a single Poll Worker Card and PIN anywhere in a county is temporarily accessed by an attacker, the attacker can easily obtain the keys necessary to compromise election data throughout the county.

To make matters worse, if a county suspected that its keys had been compromised, the only way to change them would be to load new election definitions into every ICX and ICP in the county. Doing so would likely take days or longer and might necessitate repeating logic and accuracy testing on every device.

There problems are the result of an extremely dangerous approach to cryptographic design. Best practice calls for avoiding sharing keys widely over multiple devices or authentication tokens, so as to prevent the compromise of any one device or authentication token from compromising them all.

6.2 Forging Technician Cards to Install Malware on any ICX

Issue: *Anyone can create forged Technician Cards without using any secret information. Such cards could be used to access any ICX's Android operating system and the ability to install malware.*

Although Technician Cards allow the user to access highly sensitive functions of the BMD, the ICX protocol does not authenticate them using any secret values. This makes it possible to create a forged Technician Card without knowledge of any passwords, PINs, or secret keys.

To create forged technician cards, I used the Java Card platform. A Java Card is a smart card that can execute small software applications written in the Java programming language, allowing it to emulate the behavior of other smart cards. I used [redacted] Java Cards, which are commercially available for less than $10 each [redacted]

Redaction
Redaction

I programmed a Java Card as follows. No matter what file ID the machine requests, the card always reports that it is present. (The first request is usually for an administrative card, so the attacker does not need to know what the real

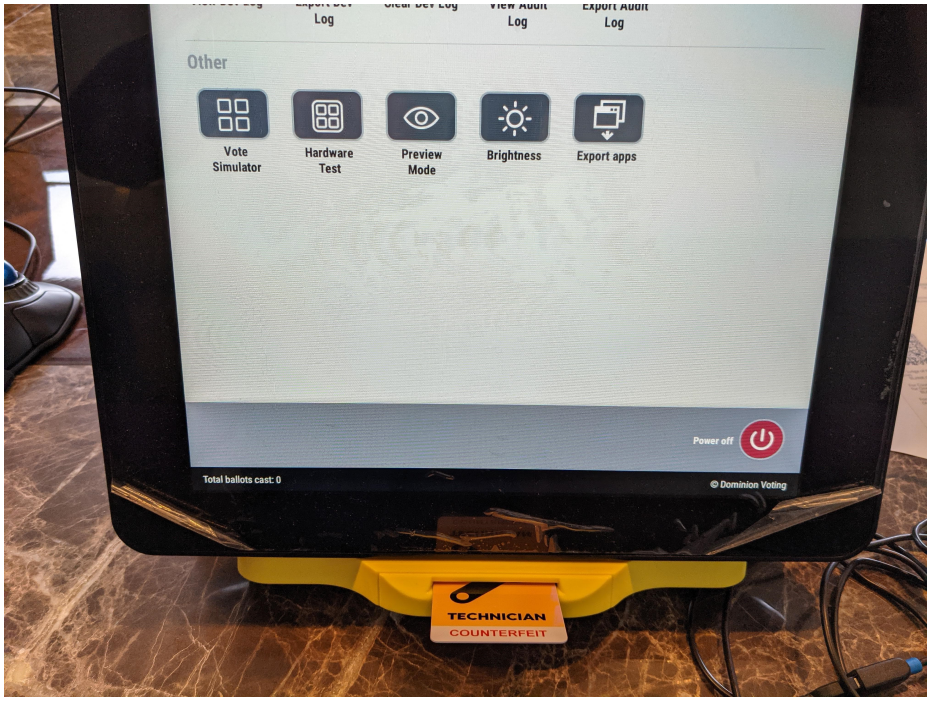


Figure 7: **Forged Technician Card.** Technician Cards can be forged without using any secret information. The self-created card can be used to unlock any ICX in Georgia (and likely those in other jurisdictions) and install malicious software.

file ID is.) To unlock the file, it accepts *any* password, so the user can enter any PIN. The card then returns a file that is *completely empty*, with every record consisting of zeroes. Remarkably, the ICX accepts the card as if it were a genuine Technician Card.

ICX Technician Cards are not restricted to a particular election or a particular jurisdiction. Consequently, the forged Technician Cards I created will work in any ICX across the State of Georgia, and likely in any other jurisdiction that uses a compatible version of the machine.

After forging a Technician Card, an attacker with physical access to a BMD can exit the ICX application and access the underlying Android operating system. With this access, the attacker can arbitrarily change the BMD's configuration, alter audit logs, or install malicious software.

6.3 Creating "Infinite" Voter Cards

Issue: *Voters can clone Voter Cards or create "infinite" Voter Cards that allow printing an unlimited number of ballots of any available ballot style.*

Forging Voter Cards requires additional steps compared to forging a Technician Card, because the BMD will only accept Voter Cards that contain the correct Election Signature, a secret value specific to the current election and county. An attacker could obtain the Election Signature by several means, such as the attack in Section 6.1, but I developed an attack method that requires only the level of access of an ordinary voter:

1. The voter enters the polling place and is issued a real Voter Card,[12] which contains the Election Signature. However, the voter cannot read the card without the election-specific Voter Card PIN.
2. To obtain the PIN, the voter inserts a specially programmed Java Card into any BMD in the polling place. I programmed a Java Card to mimic the initial steps of the ICX protocol. It reports that file ID 0x[redacted] is present, which causes the ICX to send the card the Voter Card PIN to unlock the file. The Java Card records the PIN in its internal memory and reports that it was invalid, causing the BMD to reject the card. **Redaction**
3. The voter uses a smart card reader to read the Voter Card PIN from the Java Card. They then insert the real Voter Card into the reader and use the PIN to extract its contents (as in Section 6.1), including the Election Signature.
4. Finally, the voter creates a forged Voter Card by loading the data into a second Java Card, using a process similar to the attack in Section 6.2.

These steps can be completely automated, so that the attacker need only insert the three cards into the reader for a few seconds each. An attacker in the polling place could use a battery-powered Raspberry Pi and card reader concealed on their person. Alternatively, they could smuggle the real Voter Card out of the polling place and prepare the forged card elsewhere.

When creating the forged card, the attacker can modify its data or behavior. For instance, the attacker can change the ballot style identifier to cause the BMD to print a ballot for a jurisdiction in which they are not eligible to vote.

An attacker can also create an "infinite" Voter Card that does not deactivate after the ballot is printed, allowing it to be used arbitrarily many times. Normally, the BMD deactivates voter cards by writing a value to a particular record on the card. To circumvent this, I programmed a Java Card to ignore these write operations and leave the card in an activated state, allowing it to be used indefinitely. An attacker and their accomplices could use an "infinite" Voter Card to each vote multiple ballots, although there is a risk that poll workers would notice someone printing or scanning more than one ballot in a voting session.

I previously demonstrated a similar infinite voter card attack against Georgia's old AccuVote DREs. The attack was more complicated to carry out against the DREs than against the BMDs, since the DREs verified that the card had been deactivated before returning it to the voter. The ICX does not even perform this basic check.

[12] Fulton County did not provide a Voter Card as part of the equipment. However, I was able to construct a working counterfeit Voter Card through reverse engineering.

REDACTED VERSION

7 Constructing ICX Malware

From an attacker's perspective, the most powerful position from which to manipulate votes is by using malicious software (malware) installed on polling place equipment. Georgia's ICX BMDs are highly vulnerable to malware-based attacks. This report describes numerous ways that an attackers can install malware on the machines, either with physical access (Section 8) or remotely (Section 9). Once malware is installed by any of these routes, other weaknesses in the ICX give the attacker complete control over the behavior of the machine and the ability to conceal the malware's presence very effectively.

In this section, I explain how an attacker can create malware for the ICX that, once installed by any of the methods described elsewhere in this report, manipulates printed ballots to steal votes. I explain how such malware can defeat the technical and procedural safeguards applied in Georgia, including logic and accuracy testing (LAT), firmware validation, and hash verification practices. I also present working proof-of-concept malware that demonstrates these attacks.

As I explain in Sections 8.3 and 9.3, the ICX is subject to multiple means of privilege escalation, which allow attackers to obtain "root access", i.e., full control of the device. Such access provides a variety of means by which attackers can modify the system's behavior to introduce malicious functionality, including modifying the operating system, intercepting system calls, patching the application in memory, and modifying stored data, among others. Malware could potentially utilize any of these means to manipulate ballots cast on the ICX. However, I will describe and demonstrate a different technique that is simple and effective: directly modifying the ICX Android application.

7.1 Overview of the Approach

Issue: *The ICX does not require that applications be signed by a trusted source, allowing the installation of arbitrary APKs.*

The ICX's election functionality is implemented as an Android application (the "ICP App") that is automatically launched when the device powers on. The ICX App is technically very similar to a smartphone app that a consumer would download from an app store, except that it is either pre-installed at the factory or manually installed as a software update in the form of an Android application package (APK). The actual APK filename can vary, but for simplicity I will refer to it as `ICX.apk`.

Widely-available software tools allow an attacker who can obtain a copy of the APK to quickly reverse-engineer its functionality and add arbitrarily-complex malicious logic. By this method, an attacker can alter the original APK to generate a new APK that appears identical to users but contains malicious behavior. This malicious app can simply be installed in place of the real software. Once installed, the modified APK has access to all of the same data, cryptographic secrets, and device capabilities as the original app, making possible a very wide range of attack payloads.

The Android platform requires every APK to be digitally signed by the software developer. Dominion could have used digital signatures to limit installation of apps to those signed by the company, which would complicate attempts to install a maliciously modified app. However, my tests show that the ICX does not verify the identity of the signing party. It allows the installation of APKs created, compiled, and signed by anyone. Consequently, APK code signing present no obstacle to installing malware on the ICX.

7.2 Obtaining the Real APK

Issue: *The ICX App's APK can be easily extracted given only brief, one-time access to a single BMD.*

In order to modify the ICX App, the attacker must first obtain a copy of the original software. This is easy to accomplish, given temporary physical access to an ICX. For this investigation, I copied the APK to a USB stick by using a Technician Card to access the Technical Administration menu and pressing the "Export apps" button, shown in Figure 5a.

For an attacker to do this, they would only need access a single ICX once for a few minutes. (As explained in Section 6.2, anyone can forge a Technician Card that will work in all ICXs, so access to a genuine Technician Card and PIN is *not* necessary.) Such access could potentially be gained with the help of an insider accomplice, or by breaching physical security at any point in the equipment's lifecycle: before it is delivered to the state, while it is in storage, while it is being set up and tested before an election, during transport to or from a polling place, or potentially while in use at a polling place.

Although Georgia uses physical security measures to make such access more difficult, these measures are imperfect (see Section 11.3), and it is implausible that they could prevent a determined attacker from ever accessing even a single device. Brief, one-time access to any one of the tens-of-thousands of BMDs used across Georgia would be sufficient—or to a machine from any of the many other states and local jurisdictions that use the same Dominion BMDs and software version for accessible voting. The physical security procedures for election equipment vary from jurisdiction to jurisdiction, and Georgia cannot ensure that ICXes used elsewhere are well protected.

Alternatively, an attacker could obtain a copy of the `ICX.apk` file used to update the ICX software, such as the update that Georgia installed in October 2020. The software update process involves distributing the file to all counties, copying it to hundreds or thousands of USB sticks (which are necessarily unencrypted), and having workers insert them into every BMD. An attacker who obtained a copy of any one of these USB sticks would have all the necessary information to create working ICX malware.

7.3 Decompiling and Reverse-Engineering

Having obtained the `ICX.apk` file, the next step is to reverse-engineering it to understand the functionality and how to modify it. This can be accomplished

by using the publicly available `apktool` software [46] to disassemble the original APK and translate the code into "smali" [52], an annotated, human-readable representation of the Dalvik bytecode used by Android. Although reverse-engineering the APK file is more labor-intensive than working with the original software source code, the APK file is likely to be more readily available to a wider range of potential attackers.

Based on my experiences developing similar malware for both Georgia's DREs and its new BMD system, I can compare the difficulty of attacking both types of equipment. Qualitatively, reverse-engineering the ICX app was much easier than reverse-engineering the software used in the AccuVote DREs [31]. The DREs ran Windows CE applications that were compiled into native code for SuperH and ARM processors. Unlike this native machine code, the Android Java bytecode as used in the ICX includes package, type, variable-name, and other information that makes it much easier for an analyst to interpret what the code is doing. The manual effort required to reverse-engineer it was significantly less than I expected, making it possible to alter the ICX App's functionality with relative ease. Quantitatively, reverse-engineering the app and developing basic proof-of-concept malware required approximately 25 hours of effort. This is far less effort than was required when I reverse-engineered the AccuVote DRE and developed similar malware in 2007. For these reasons, I conclude that malware is easier to create for the ICX system than it was for Georgia's old DRE system.

7.4 Modifying the ICX App to Change Votes

Due to the structure of Android applications, it is relatively straightforward to make arbitrary changes to the ICX App's behavior. We used Java, a high-level programming language, to implement demonstration malicious functionality as a Java package. Using a high-level programming language is much less labor intensive than writing the malicious logic in low-level bytecode. We compiled the Java package into low-level smali instructions using the publicly available `java2smali` software tool [51] and inserted the smali files into the disassembled APK's file structure. This arrangement allows the new code to be invoked with only small, targeted changes to the original app's code.

For example, in my demonstration malware, one place where such malicious logic is injected is in the code that generates the QR code for printing. Through reverse engineering, we located the existing code that constructs the vote data that will be encoded in the QR code. Changing just two bytecode instructions in this function[13] causes it to pass the data to a function in the new Java package, giving the malicious logic an opportunity to change the data before the QR code is produced.

As a simple demonstration, I implemented malicious logic that modifies the QR code so that the vote recorded for a specific "Yes or No" contest is always "No". The logic clears the "No" bit and sets the "Yes" bit for a specific byte

Redaction

[13] Specifically, the function [redacted]

within the data representation (see Figure 3) and returns the modified data to the original logic to be packaged into the QR code and printed. The result is that the data in the QR code—and the vote counted by the scanner—reflects a fraudulent choice I control, rather than the voter's intended selection.

An attacker could, of course, implement different or more complex logic to determine when and how to cheat. Malware on the ICX has access to the complete ballot design, and could be programmed to cheat in favor of candidates from a specific party, in contests for a particular office, or in particular kinds of elections. For example, it could always favor one party's candidate in U.S. House races during general elections. An attacker also could choose to change only the QR code or both the QR code and the human-readable text. Malware with such variations could be constructed in the same manner as the proof-of-concept malware described here.

7.5 Defeating Applicable Defenses

Malware running on the ICX can defeat the various technical and procedural defenses that the Dominion system and the State of Georgia currently employ.

Defeating Logic and Accuracy Testing In logic and accuracy testing (LAT), workers cast a small number of votes with known selections, then check whether the voting system's output reflects the correct totals. This form of testing is designed to detect errors in the ballot design or counting logic. It can be easily defeated by ICX malware.

Georgia's LAT procedures (Exhibit B) involve only minimal testing of the ICXs. Only a single test ballot per ICX is required to be printed. To avoid detection, the demonstration malware simply tracks how many ballots have been printed since the machine was turned on and skips cheating on the first n ballots (for an attacker-configuration number). If Georgia were to improve its LAT process by testing with a greater number of ballots, attackers could simply increase the number of ballots the malware skipped accordingly.

Even if the state adopted a much more complex LAT procedure, so long as the testing process was publicly documented, attackers could design malware to maximize cheating while minimizing the probability of getting caught. Much as Volkswagen's emission systems were famously designed to detect that they were being tested by the EPA and to only cheat while not under test [84], ICX malware can be programmed to detect and circumvent LAT. For example, malware could be programmed to only cheat on the day of the election, or only during specific hours on that day. It could also be programmed to monitor how the machine was used and to only start cheating if the rate of voting, pattern of votes, number of corrected mistakes, and other characteristics matched the expected behavior of real voters. No practical method of pre-election or parallel testing can rule out malware-based fraud [80].

Defeating the QR Code MAC Although the QR code contains a cryptographic message authentication code (MAC) that scanners use to verify its integrity (as explained in Section 5.2), this poses no obstacle to ICX malware.



Figure 8: **Defeating Hash Validation.** The ICX App displays the SHA-256 hash of its APK on the screen, as shown here. However, this behavior is controlled by the app itself, so a maliciously modified app can simply show the expected hash value instead of its real one, thereby avoiding detection.

The demonstration malware changes vote data before the app computes the MAC. This allows such malware to add, remove, change, or spoil votes in the QR code while ensuring that the MAC remains valid. Alternatively, since the secret key used to generate the MAC is necessarily accessible to the ICX App, malicious logic in a modified app could use the key to generate valid MACs itself.

Defeating APK Hash Validation As shown in Figure 8, the ICX can display the SHA-256 hash of the installed APK on its screen, supposedly allowing both election officials and voters to confirm what software is running. However, much like the QR code MAC, this hash value is computed by the ICX App itself and can therefore be trivially defeated by malicious logic added to the app.

In the demonstration malware, we identified the code that computes and displays the hash,[14] and modified it to simply replace the computed hash value with the hash of the unmodified APK. This ensures that the ICX always displays the official APK's hash even though it is running a maliciously modified APK.

Redaction

[14] Within the function [redacted]

Defeating External APK Validation As described in Section 7.2, the ICX App contains functionality to export the currently installed APK to a USB stick for verification. Once the APK file has been exported, its hash can be securely computed using a trusted, external device. Exhibit C shows that this was the method used by Pro V&V used in November 2020 to validate the software on a small number of ICXs in six Georgia counties.

A malicious ICX App can easily defeat this safeguard, too, because the export process is performed by the app itself. Just as the modified app can display the hash of the original APK, it can also export the original APK file instead of its own. To accomplish this, we store a copy of the original APK and modify part of the export code[15] to change the location from which the exported APK file is copied to be the location of the original APK. Since the exported APK is identical to the original APK, any hash validation or forensic analysis of it will fail to detect the malware, including the kind of analysis Pro V&V performed.

Defeating Voter Verification and Auditing As discussed in Section 3.2, voters have no practical way to verify the contents of the QR codes. Since the scanners read only the QR codes, and the voter can only review the printed text, there is no way for voters to verify the portion of their ballots that is actually tallied. Therefore, attacks that change the QR code and leave the human-readable portion of the ballot unmodified would almost certainly not be detected by voters.

In principle, *election officials* could verify the QR codes by decoding them and comparing the output to the text on the ballots. To our knowledge, no jurisdiction has ever done so, and Georgia has announced no plans to do so.

A rigorous risk-limiting audit (RLA) would also be likely to detect an attack that changed only the QR codes, if the attack changed sufficiently many votes to alter the outcome of the contest targeted by the audit. However, Georgia regulations call for an RLA only in the November election of even-numbered years, and only targeting a single, state-wide contest chosen by the Secretary of State [69]. Therefore, such cheating likely would not be detected in the vast majority of elections and contests.

As discussed in Section 5, attackers could also choose to cheat by changing both the QR codes and the human-readable text on a small fraction of ballots, such that both reflected the same fraudulent choices. This would be completely undetectable by an RLA or a hand count. Although, in principle, voters might notice that the printed ballots were wrong, human-subjects research indicates that only a small fraction of voters verify their ballots closely enough to notice such errors [9, 54]. As a result, when vulnerable BMDs are used for all in-person voting, as in Georgia, malware could alter enough votes to change the outcome of a close race while likely triggering too few voter complaints to alert election officials that there was a systemic problem [9].

[15]The function [redacted]. **Redaction**

7.6 Conclusions

I have demonstrated how it is possible to create a malicious version of the ICX App that selectively alters ballot QR codes to steal votes and favor an attacker's preferred candidate.

I have also demonstrated that such malware can take steps to effectively defeat Georgia's procedural defenses. Once installed on an ICX, the proof-of-concept malware I created would not be detected by the state's logic and accuracy testing, hash checking, and APK validation procedures. Even a post-election forensic audit, if conducted using the methodology that Pro V&V applied following the November election, would not detect well designed malware.

Although cheating by malware that changed only ballot QR codes could be detected by a rigorous risk-limiting audit if the malware altered enough votes to change the outcome of the contest targeted by the audit, the vast majority of elections and contests in Georgia (even high-profile ones) are not audited at all. Even in contests that are subject to an RLA, malware that changed both the QR codes and the ballot text could likely avoid detection while changing individual votes and the outcome of a close race.

While I have created a concrete example of BMD malware as a proof-of-concept, numerous variations are possible, both in terms of the technical means by which the malware affected the ICX's operation and the specific effects. Many of these variations could accomplish the same result: stealthily changing Georgia citizens' votes.

REDACTED VERSION

8 Installing Malware Locally

An attacker who has access to an ICX BMD has multiple ways to install malicious software, such as the vote-stealing malware described in Section 7. In this section, I describe three separate techniques for accomplishing this that I have successfully tested with the ICX from Fulton County.

These techniques do not require any secret passwords, PINs, or keys, nor does the attacker have to open the device's chassis or break any tamper-evident seals. They only need physical access to the BMD for a few minutes. Attackers could gain such access before machines are delivered from the manufacturer, while they are in storage, while they are being prepared for use in an election, or at the polling place. As I will show, malware could potentially even be installed by regular voters, without any special level of access or technical skill.

8.1 Attaching USB Devices to the ICX

Issue: *The ICX fails to adequately restrict the kinds of devices that can be attached to its USB ports, including the externally exposed USB cable that connects to the printer.*

The malware installation techniques described here involve attaching USB devices to the ICX. The machine has several external USB ports behind plastic doors on the rear of its enclosure. One of the USB ports is used to attach a cable that connects to the printer. They are also used to attach USB drives from which election definition files and occasional software updates are loaded.

Dominion could have designed the ICX to limit the kinds of devices that are allowed to attach to each USB port—i.e., by allowing only specific models of printers to communicate with the port used for the printer, and only specific models of USB drives to connect to the port used for loading data. Instead, all of the exposed USB ports can be used interchangeably, and there do not appear to be any technical restrictions on the devices that may be connected.

I understand that Georgia requires the USB port doors to be closed and secured with tamper-evident seals while the machine is in use at a polling place. The kinds of tamper-evident seals typically used in election systems are known to be easily bypassed using commonly available tools [7]. However, this is unnecessary for the attacks described here, because the seals present no practical obstacle to connecting new USB devices.

Figure 9 shows how the ICX is deployed in polling places in Fulton County and other Georgia localities. The USB cables that attach two printers to a pair of BMDs are visible. Observe that the ends of the cables that attach to the back of the printers are not sealed to the printers. It would be possible for voters to reach behind the printers and disconnect the cables without leaving physical evidence.

Using an inexpensive adapter, a USB drive or other device can be attached to the end of the cable, and it will function as if it was plugged in directly to the BMD's USB port. An example of this arrangement is shown in Figure 10.



Figure 9: **ICX USB Interfaces are Exposed to Voters and Unsealed.** A USB cable connects the BMD to an off-the-shelf laser printer. At polling places, the end of the cable attached to the printer is physically accessible to voters, and it is not protected by a tamper-evident seal. Voters could install malware on the ICX by attaching a device to the end of this cable. *Photograph taken by Harri Hursti during polling place observation in Fulton County, November 3, 2020.*

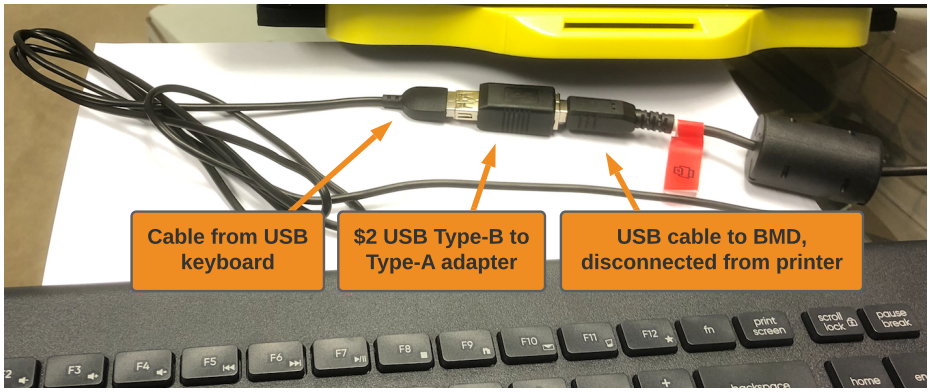


Figure 10: **Attaching a USB Device to the ICX via the Printer Cable.** The BMD's USB cable is not sealed to the printer, and voters can simply reach behind the printer and disconnect it. Using an inexpensive and widely available adapter, any standard USB device (such as the keyboard shown here) can attach to the end of the cable and operate as if it were plugged in directly to the ICX.

8.2 "Escaping" the ICX App

Issue: *As a result of Georgia's installation of a software update in October 2020, the ICX's Android operating system settings can be accessed by attaching a USB keyboard, allowing the installation of malware.*

In October 2020, shortly before the start of early voting in the November election, Georgia installed a purportedly *de minimis* software update on its BMDs to correct a user-interface glitch. In support of Plaintiffs' opposition to this change, I testified that "in complex computerized systems like Georgia's election equipment, last-minute changes, even seemingly small ones, can introduce serious and difficult-to-foresee consequences" [41, ¶ 5]. I drew an analogy to the Boeing 737 MAX aircraft, where a small, last-minute change to correct a single problem inadvertently created a much more dangerous failure mode that reportedly led to two fatal crashes [56].

My testing shows that installing the ICX software update did indeed create a dangerous security problem. It left the BMDs in a state where anyone with physical access, including non-technical voters, could install malicious software.

The problem relates to the method by which the Android operating system on the ICX is "locked down". When Android is used on consumer devices like phones and tablets, users can open any installed app, switch between apps, and access the Android Settings app, which allows the installation of new software and changes to security-critical settings. If Android is used for building special-purpose devices that serve the public (often referred to as "kiosks"), the manufacturer needs to take steps to restrict access to these functions, usually by preventing unauthorized users from leaving a particular app that provides the device's user interface.

REDACTED VERSION

Recent versions of Android provide a "dedicated devices" programming interfaces that device makers can use to securely lock down the operating system [4]. However, instead of using such an API, the ICX takes an *ad hoc* approach. It sets the ICX App as the system's "launcher", i.e., the app that provides the user interface for the device's "home screen" or "desktop" [18]. This ensures that the ICX App is automatically started when the device powers on, and it prevents users from directly launching other apps via the normal launcher interface.

This approach has dangerous limitations. Making the ICX App the launcher does not block users from *switching* to other apps. One way in which users can still switch to other apps is by attaching a USB keyboard and pressing the Alt+Tab key combination, which cycles through apps in the Android Overview screen[16] [1]. This keyboard shortcut does not allow the user to switch to any app *installed* on the device, but rather only to an app that has previously been started. In the version of Android installed on the ICX, apps are added to the Overview screen whenever they are used, and they remain accessible via Alt+Tab even after the device is rebooted, unless they are explicitly removed through the Overview interface [5, 90].

There would not be a problem if other apps had not previously been used, or if they had been properly removed. However, crucially, Dominion's 40-step process for installing the ICX software update (Exhibit A) used two sensitive apps, File Manager and Settings, and neglected to remove them from the Overview screen. This means these apps are accessible through the use of a keyboard on any BMD where the software was updated according to Dominion's instructions. It is a reasonable inference that the instructions were not subjected to rigorous security testing before use, since the update was installed on BMDs across Georgia only days after being created.

To prevent these apps from being accessible, it would have been necessary to perform a process like this after installing the update:

1. Use the "Toggle" button at the bottom of the screen to enable the Android navigation controls, confirm the change, and click OK to reboot the ICX.
2. Launch Android Settings from the Technical Administration menu.
3. Press the "Toggle" button again and press OK to confirm, but do *not* immediately reboot.
4. Press the square App Overview button at the bottom of the screen.
5. Swipe right on the pictures of every previously opened app to remove them from the Overview screen. Once all are closed, the ICX App will reappear.
6. Power off the device.

This would prevent the Alt+Tab vulnerability, but Dominion's instructions included no such steps. Instead, testing shows that after completing the update, the Settings and File Manager apps remain perpetually available through use of a keyboard, even after the device is powered off and on again multiple times. As I describe below, attackers can exploit ICXs in this vulnerable state to install malicious software.

[16]This key combination switches between previously started apps, just like Alt+Tab on Windows and Command+Tab on macOS switch between open windows.

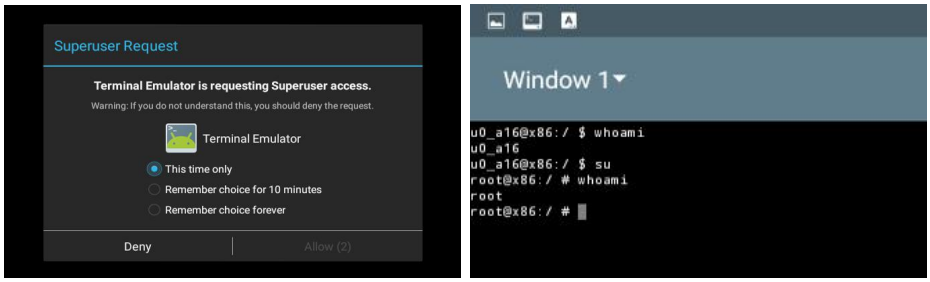


Figure 11: The ICX has a pre-installed **Terminal Emulator app** that provides access to a Linux command-line interface. Simply by confirming an on-screen prompt (*left*), the user can obtain "root" access (*right*), allowing subsequent commands to bypass Android's access control and privilege separation defenses.

8.3 Accessing a Root Shell via the Built-In Terminal App

Issue: *The ICX has a built-in Terminal Emulator app that is configured so that the user can easily obtain a command-line shell with supervisory privileges.*

After escaping kiosk mode, an attacker can easily launch any app installed on the ICX. The machine contains 20 pre-installed apps, most of which appear unnecessary for its use as a BMD. Most notably, there is a Terminal Emulator that provides access to a Linux shell, a powerful text-based user interface.

Moreover, the ICX is configured such that the Terminal Emulator user can easily obtain supervisory ("root") access privileges by simply selecting "Allow" at an on-screen prompt, shown in Figure 11. With root privileges, terminal commands can completely bypass the Android operating system's access control restrictions and make arbitrary changes to the device's data and software.

The Terminal Emulator made analysis of the device much more efficient, since I was able to easily access, control, and modify any part of the data or software. It also makes it easy for an attacker to install programs or run automated commands for malicious purposes.

8.4 Manual Malware Installation Process

I will now walk through a process that exploits the vulnerable state of the BMD to install malware. This involves the attacker attaching a USB keyboard and then a USB thumb drive to the machine, as described in Section 8.1. These manual steps are relatively cumbersome and time-consuming, and they would be impractical to carry out in the polling place, but I describe them here for expository purposes. I will later show how the entire process can be automated, so that the attacker need only briefly attach a single USB device.

Preparing for Installation The attacker attaches a standard USB keyboard to the BMD, by attaching it to the end of the exposed printer cable through a USB adapter, as shown in Figure 10. By pressing Alt+Tab, the attacker switches

from the ICX App to the Android Settings app, from which they can access the Terminal Emulator. From there, the attacker escalates to root privileges by typing the `su` command and confirming the on-screen prompt shown in Figure 11. They then use Linux shell commands to copy the APK of the installed ICX App to a temporary location, so that the malware can export it for verification, and to copy any election definition files stored by the ICX App to a location where the malware can access them.

Installing Malicious App Next, the attacker returns to Android Setting, then disconnects the USB keyboard and connects a USB drive containing the installation file for a malicious modified ICX App like the one created in Section 7. Using Android Settings, the attacker uninstalls the original ICX App, enables a configuration setting to "[a]llow installation of apps from unknown sources", installs the malicious ICX App from the USB drive, then disables the configuration setting, all in a manner similar to Dominion's official software update instructions.

Post-Installation Clean Up Finally, the attacker reattaches the USB keyboard and uses the terminal to clean up traces from the installation process and to restore the previously-saved election definition files to their original location in the now-malicious ICX App's storage hierarchy.[17] The attacker then disconnects the USB keyboard and launches the malicious app. The BMD appears to function normally, but the maliciously modified software can tamper with printed ballots to steal votes.

8.5 Automating Malware Installation

The process described above can be completely automated, so that an attacker can install malware by attaching a single USB device to the exposed printer cable for less than two minutes. The automated process is simple and fast enough that it could potentially be carried out by a voter in the polling place.

To automate the attack, I used a device called a "Bash Bunny", which is commercially available for less than $100 [37]. A widely-used tool for penetration testing, the Bash Bunny (shown in my hand in Figure 12) looks similar to a typical USB thumb drive, but it acts simultaneously as a USB storage device and a simulated keyboard. Once attached to a target machine, it sends a pre-programmed sequence of keystrokes to execute the attacker's objectives. I prepared the Bash Bunny by copying the malicious APK to its USB storage and programming it to send keystrokes that carry out the installation process, following a sequence of operations similar to those in Section 8.4.

Once the Bash Bunny is programmed, launching the attack requires no technical skills. A voter could do so by following simple directions like these:

1. Take a pre-programmed Bash Bunny and a USB adapter to a polling place. Check in normally, then select an out-of-the-way BMD with a screen that is difficult for poll workers to observe.

[17] In fact, the malicious app has the ability to execute commands with root privileges itself, as described in Section 9.3, so these steps could also be executed automatically by the malware once it was installed.



Figure 12: **Installing Malware in the Polling Place.** An attacker can install vote-stealing malware on the ICX by attaching a small USB device for under two minutes. This sequence shows me reaching behind the printer, unplugging the cable that leads to the BMD, and connecting the device to the end of the cable. This can be accomplished in seconds and does not require breaking any tamper-evident seals. It could potentially be carried out surreptitiously by a voter.

2. With one hand, reach behind the printer and unplug the USB cable. Attach the Bash Bunny to the end of the cable (as shown in Figure 12), and leave it out-of-sight behind the printer.
3. Stand in front of the BMD and pretend to vote, carefully blocking the screen. Wait until the process completes (less than two minutes).
4. Discreetly unplug the Bash Bunny and reconnect the cable to the printer.
5. On the BMD screen, tap the icon that says "ImageCast X".
6. Quickly proceed to print your ballot, then scan it like any other voter.

Of course, an attack at a polling place might be more likely to be detected (depending on the circumstances) than an attack conducted in a non-public setting. Similar steps, but with less need for subterfuge, could be used by election workers or outsiders who had brief private access to BMDs.

8.6 Local Malware Installation using a Forged Technician Card

While the attack method demonstrated above exploits the vulnerability created when the October 2020 software update was installed, there are also other means

of installing malware. One is to use a forged Technician Card created using the technique described in Section 6.2, which requires no secret passwords, keys, or PINs, but only a widely available $10 Java Card with some simple programming.

By inserting a forged Technician Card like the one I created, the attacker can access the Technical Administration menu, exit the ICX App, and then proceed to install malware using essentially the same on-screen process that is used to install official software updates. As before, a Bash Bunny could be programmed to automate the necessary steps, so that malware installation could be performed quickly by anyone with brief physical access to an ICX.

8.7 Local Malware Installation via Android Safe Mode

Issue: *A local user can reboot the ICX into "Safe Mode", allowing full control of the Android operating system.*

A third method for installing malware is to exploit a publicly known security flaw in the ICX. According to a Dominion customer advisory dated January 2020, "[i]f the mechanical power button (behind the ICX door) is pressed a power down option is presented. At this point, if the power down screen button is pressed and held, the 'safe mode' option is presented" [22].

I tested this behavior on the ICX. As shown in Figure 13, holding the power button and selecting "Reboot to safe mode" will cause the BMD to restart with the standard Android Launcher available, providing unrestricted control of the device, including access to the Android Settings, File Manager, and Terminal Emulator apps and the ability to install or remove software.

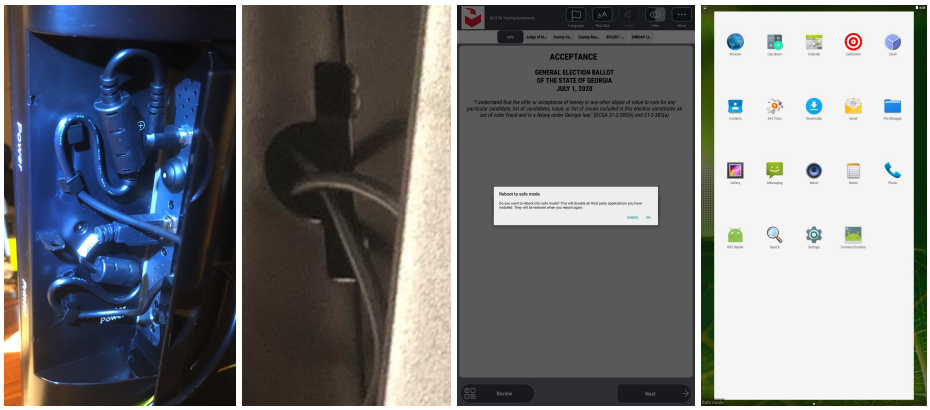


Figure 13: **Rebooting the ICX into Safe Mode.** *Left to right:* (1) The ICX power button is located behind a plastic door on the side of the machine; (2) Even with the door closed, an opening for cables allows the button to be pressed by inserting a metal tool; (3) Holding the button causes the machine to show a "Reboot to safe mode" prompt; (4) When the machine reboots, the user has unrestricted access to Android, including the ability to install malware.

"Safe mode" is an Android feature that is intended for use in recovering from software issues [36]. If the ICX used a more recent version of Android, Dominion could have easily disabled it with the `DISALLOW_SAFE_BOOT` setting introduced in Android 6.0 in 2015 [6]. Instead, Dominion advises that "[i]t is imperative that safety seals be used on the doors on the back side of the ICX to prevent unauthorized access to the mechanical power button" [22].

Unfortunately, the design of the door that covers the power button makes it difficult to secure. The door contains an opening to allow cables to pass through. By inserting a metal tool through this opening, an attacker can press the power button even with the door fully closed.

Even if both the door and the opening were securely sealed while the BMD was in use *by voters*, election workers need to access the power button so they can turn on the machine, both during pre-election testing and at the polling place. Dishonest election workers (like those emphasized by Defendants and their expert Michael Shamos) or intruders who gained access to the machine during these times could exploit the vulnerability to install malware.

REDACTED VERSION

9 Installing Malware Remotely

I have described several methods by which attackers can install malware with only brief physical access to an ICX. Although these are severe vulnerabilities, the ICX is also vulnerable to an even more dangerous method of malware installation. By modifying the election definition files that election workers copy to the BMDs before each election, attackers can spread malware to them remotely, with no physical access to the individual machines. By levering this vulnerability, an attacker who infiltrates a county Election Management System (EMS) can spread malware to every ICX in the county, and infiltrating other systems could allow vote-stealing malware to be spread to all ICXs state-wide.

This attack is somewhat more complex than the ones I have described so far, so I will explain it in stages. I first describe how BMD election definitions are produced and distributed. Then I will describe a critical vulnerability that allows a modified election definition file to run arbitrary code on the ICX. Finally, I will explain how this vulnerability can be exploited to remotely install malware.

9.1 ICX Election Definitions

Prior to each election, the ICX must be configured with the available ballot styles, contests, and choices. This data is created using the Democracy Suite EMS software and packaged into an election definition file that is distributed to the BMDs on USB sticks [19].

Structure and Encryption

My testing shows that ICX election definition files are Zip archives that are encrypted using the AES (a.k.a. Rijndael) algorithm. The filename can vary, but I will refer to it as "`ICX.dat`". The Zip archive contains a SQLite database (`electiondata.db3`) that defines the ballot designs and election-specific settings. It also contains assorted graphic files, audio files, and language translation files that are used for presenting the ballots to voters.

I analyzed county election data from the November 2020 and January 2021 elections produced by State Defendants. The data shows that, under current Georgia practice, all BMDs within a county are loaded with the same `ICX.dat` file, which provides every local ballot design used in the county. Moreover, all scanners and BMDs within each county use the same encryption key and initialization vector (IV) during a particular election. Given access to the county EMS or Election Package, the key and IV can be retrieved from the election project database using the SQL command:

```
SELECT RijndaelKey, RijndaelVector from ElectionEvent;
```

I show in Section 6.1 that the same key and IV can also be extracted from any Poll Worker Card in the county, given brief access to the card and PIN.

After obtaining the key and IV, the ICX election definition file can be decrypted using the following shell command:

```
openssl enc -d -aes-128-cbc -K $(xxd -pu <<< 'RijndaelKey')
    -iv $(xxd -pu <<< 'RijndaelVector') -in ICX.dat -out ICX.zip.
```

Issue: *ICX election definition files are not digitally signed, and they can be modified by anyone with access to a symmetric encryption key that is shared by all scanners and BMDs within each county.*

Dominion could have used digital signatures to secure the ICX election definition files against malicious modification. Instead, there does not appear to be any cryptographic integrity protection, beyond verifying that the decrypted file is a properly formed Zip archive. As a result, anyone with access to the encryption key and IV discussed above can decrypt the `ICX.dat` file, modify it, and re-encrypt it using a command similar to the one shown above. My testing shows that the ICX will accept the modified file as if it were genuine.

Distribution and Points of Attack

In Georgia, each county operates a separate election management system (EMS)—a collection of servers and computers that operate the Dominion Democracy Suite EMS application software [20]. Before each election, Dominion centrally prepares an initial Election Project for each county (the data that defines the contests and candidates on the ballots).[18] The company sends each county its Election Project in the form of an Election Package, a Zip archive that contains the election project database used by the EMS software, ballot PDF files for printing, and individual election definition files to be copied to the ballot scanners and BMDs. At the county, workers import the Election Package into the county EMS. As described in Exhibit B, workers then copy the election definition file from the EMS to one or more USB drives, which they insert into each ICX to load the election definition into the machine's internal storage.

This election definition distribution process introduces two kinds of opportunities for remote malware attacks:

At the county level. An attacker who infiltrates a county's EMS can modify the county's `ICX.dat` file before it is copied to USB drives, and thereby spread malware to all BMDs in the county.[19]

At Dominion. An attacker who infiltrates the facility where Dominion prepares Election Projects could modify the election definitions distributed to all Georgia counties, and thereby spread malware to every ICX used in Georgia.

Such attacks could be automated through the use of further malicious software installed on infiltrated EMS systems. That software would be programmed to detect when a new Election Package was loaded. It would then locate the `ICX.dat` file and modify it using the key and IV from the EMS database. Any BMDs on which the modified election definition was subsequently loaded would become infected.

[18] In March 2020, Eric Coomer testified that a single Dominion employee was preparing the Election Projects for all 159 Georgia counties [82, 65:17-69:22].

[19] Alternatively, an attacker with only access to the USB drives could modify the file before it was loaded into the BMDs, given access to a Poll Worker card and PIN from which to obtain the encryption key.

REDACTED VERSION

9.2 Directory Traversal Vulnerability

Issue: *The ICX software contains a critical directory traversal vulnerability that allows a maliciously modified election definition file to overwrite arbitrary files.*

The ICX contains a critical vulnerability in the code that loads election definition files. The problems is a so-called "Zip Slip" vulnerability, a common but severe flaw in software that processes Zip files, which has been observed to be "especially prevalent" in Java-based software such as the ICX App [81].

Zip files, such as the ICX election definitions, can contain a hierarchy of folders and files. The Zip format represents this by storing each file's name together with its directory path. For example, a file `logo.png` in a folder `resources` within would be represented in a Zip file using the name `resources/logo.png`.

Normally, when software extracts a Zip file's contents, it recreates the contents inside a specified target folder. The Zip Slip vulnerability allows a maliciously-crafted Zip file to create or overwrite files in any writable location on the system.

To do so, the attacker changes the path names in the Zip file to begin with "`../`". Secure Zip extraction code will detect and ignore these characters, but software that suffers from the Zip Slip vulnerability will treat them as part of the file's location. Operating systems interpret these special characters not as a literal name but as a reference to the target location's *parent* folder. By repeating these characters multiple times, the attacker can traverse to the root folder and direct the file to be stored in any writable location on the system. For example, a Zip file crafted to contain a file named `../../etc/passwd` that was extracted by a vulnerable application inside the target folder `/root/tmp/` would result in an existing file named `/etc/passwd` being overwritten with the new file's contents (so long as the running process had permission to write to `/etc/passwd`).

The ICX suffers from exactly this problem. When an election definition file is loaded, the system decrypts it to a Zip file and extracts the contents in a specific storage location. However, the ICX fails to check whether the file names contain parent folder references. As a result, an attacker can create a modified election definition file that will create or overwrite files in any location on the device that is writable by the ICX App. As I explain below, an attacker can leverage this capability to execute arbitrary code and install malware.

9.3 Arbitrary Code Execution as Root

Issue: *The BMD runs code with root privileges from a file that is writable by the ICX App. When combined with the directory-traversal vulnerability, this allows a malicious election definition file to execute arbitrary code as root.*

The Android OS employs access control and privilege separation to limit what files an app can modify. These defenses normally prevent an app from accessing another app's data, changing its own APK, or installing a new app. However, I find that weaknesses in the ICX software allow attackers to circumvent these defenses. A malicious election definition file can cause attacker-supplied code to

be executed with "root" privileges—complete control of the device's software, including the ability to override all file access restrictions and install malware.

The ICX App contains a native-code executable file named [redacted] Redaction that is delivered as part of the APK. The app does not run this file directly, but rather it makes a copy in the folder [redacted] Redaction, for which the app has write permission. Each time the app starts, it checks whether the file is already at that location, and, if not, it extracts it from the APK and places it there.

The ICX Android distribution includes a vendor-specific system service called [redacted] Redaction that it uses to control various hardware functions. This service uses a dangerous and insecure design that allows the ICX App to execute arbitrary commands with root privileges. Every time the ICX App starts, it uses [redacted] Redaction to run [redacted] as root.

An attacker can exploit these behaviors in combination with the directory traversal vulnerability to create an election definition file containing malicious code that will be executed with root privileges. The attacker merely has to modify the `ICX.dat` file so that, when the Zip archive is extracted, it overwrites [redacted] Redaction with the malicious code. The next time the BMD is powered on, the ICX App will use [redacted] Redaction to run the file with root privileges, giving the attacker's code full control of the device.

9.4 Installing Malware from the Election Definition File

Given the ability to execute arbitrary code as root, the last step to remotely installing malware is replacing the ICX App's code with a maliciously modified version, which can be constructed as described in Section 7. An attacker could replace the app's code by several means; I demonstrate one particularly efficient method that is a variation of a technique presented at the Black Hat Asia conference in 2015 by Paul Sabanal [70].

The ICX App, like most Android apps, is written in the Java programming language. Prior to distribution, the Java source code is compiled into a Dalvik Executable (DEX) file that is combined with other resources to create a self-contained APK file. When the APK is installed on the machine, Android performs a process called ahead-of-time compilation to generate code that is optimized for execution efficiently on the device's hardware. This involves translating the Java bytecode in the DEX file into native code for the machine's processor, which gets stored as what is called an OAT file [2]. When the ICX App runs, it is the translated code in the OAT file that actually gets executed, *not* the original code from the APK.

Sabanal's technique is to replace the OAT file with one containing malicious code, rather than the more obvious approach of replacing the DEX file. In addition to other technical advantages that I discuss below, this avoids introducing a potentially noticeable delay caused by the ahead-of-time compilation process. Though I did not attend Sabanal's original presentation, I found that his publicly available slides were effectively a walk-through of how to perform the

technique [71]. To streamline the process, rather than directly using the `dex2oat` tool to create the malicious OAT file, I simply installed my malicious APK and copied out the OAT file that was generated by the Android installation process. I then modified the OAT file to reflect the DEX path and checksum expected by the operating system, as described in Sabanal's presentation.

Putting the Pieces Together I created proof-of-concept malware that installs automatically when a surreptitiously altered election definition is loaded into the BMD. The key steps are described below:

I started by decrypting the original election definition file and then modified the internal Zip archive to add two new files:

- A maliciously modified version of the ICX App, in the form of an OAT file.
- A shell script (a simple program) that, when run on the BMD with root privileges, overwrites the OAT file for the installed ICX App with the malicious OAT file extracted from the Zip archive, then restarts the ICX App.

I added the shell script to the Zip archive in a way that exploits the directory traversal vulnerability, so that, when the BMD extracts the election definition file, the existing [redacted] program is replaced with the shell script. Finally, I encrypted the modified Zip file with the original encryption key. (These steps are performed automatically by a shell script I created.)

Redaction

The result is a malicious election definition file that appears to behave identically to the original election definition when loaded onto a BMD by an unwitting election worker. However, the next time the BMD is powered on, the shell script runs and invisibly replaces the ICX App's logic with malicious code.

9.5 Defeating Security Precautions More Easily

Like locally installed malware, remotely installed malware could use the mechanisms described in Section 7 to defeat Georgia's logic and accuracy testing (LAT), hash verification, and external APK validation. However, an advantage to the infection technique described here is that can intrinsically bypass these protections with no additional effort on the attacker's part.

Defeating Logic and Accuracy Testing Georgia's pre-election testing procedures (Exhibit B) specify that an election worker should:

1. Insert a USB stick containing the election definition file.
2. Use a Technician Card to copy the file to the BMD.
3. Use a Poll Worker Card to open polls using the election definition.
4. Vote and print at least one test ballot from the BMD.
5. Use a Poll Worker Card to close polls and power off the BMD.
6. Seal the BMD for delivery to the polling place.

When the election definition is loaded from the USB stick in step 2, the file is merely copied to the BMD's storage. Its contents are extracted during step 3,

when the encryption key from the Poll Worker card is provided. This sets the stage for the malware to be installed when the BMD is next powered on, at the polling place. Note, however, that LAT is performed immediately, without restarting the BMD first. This means that testing for the current election will be finished before the malware is activated, so no LAT-circumvention logic is required.[20]

Defeating Hash Validation and External APK Validation The hash value that the ICX App displays is computed by the app itself by hashing its installed APK file, which is stored within the Android filesystem. However, the malware installation technique described here overwrites the dynamically generated OAT file and leaves the original APK intact. As a result, the hash reported by the app does not change, even though the running logic has been maliciously altered. Similarly, when the ICX App exports its APK for external verification, it copies the same locally-stored APK to a USB drive. Since the remotely installed malware does not change the APK, the exported file will contain no evidence of infection.

9.6 Conclusions

I have identified critical vulnerabilities in the ICX software that enable an attacker to remotely execute arbitrary code on the device. These vulnerabilities can be exploited by maliciously altering the election definition files that workers copy to all ICXs before every election.

Security experts consider arbitrary code execution to be one of the most dangerous classes of vulnerabilities, particularly when it can be exploited to run code with root privileges, as it can on the ICX. In 2006, Harri Hursti discovered a similar arbitrary code execution vulnerability that affected Georgia's old AccuVote TS-X DREs [45]. At the time, Defendants' expert Michael Shamos called it "the most serious security breach that's ever been discovered in a voting system" [44]. The vulnerabilities in the ICX are as or more severe.

Using these vulnerabilities, I developed functional proof-of-concept malware that automatically and invisibly installs itself on any ICX on which an infected election definition file is loaded, then manipulates voters' printed ballots to steal votes. By compromising election definition files in this way, an attacker with access to a county's EMS could spread malware to all ICXs in the county, and an attacker who infiltrated the systems that Dominion uses to prepare initial election projects for all Georgia counties could spread vote-stealing malware to every ICX used in Georgia. As I discussed in Section 3.2, even the ICX's use of a paper trail poses no obstacle to vote-stealing attacks in the vast majority of elections and contests, and malware can also evade Georgia's other technical and procedural defenses.

[20] Of course, an attacker might aim to create malware that would cheat in *future elections* too. In that case, the methods in Section 7 could still be used to defeat future rounds of LAT.

10 Manipulating Logs and Protective Counters

Two additional protections that the ICX maintains are an "audit log" of events before, during, and after the election, and a "lifetime counter" of the number of ballots printed. Both can be easily defeated.

Public and Lifetime Counters The ICX App uses two counters to track the number of ballots it prints: the "public counter" and the "lifetime counter". These values are used by election workers to confirm that all ballots are accounted for and that the counts match between the BMDs and scanners.

The public counter is displayed on the Poll Administration Menu and at all times in the lower-left corner of the screen, where it is readily visible to voters. By design, the public counter can be reset by election workers when the poll is closed by using the "Reset" button in the Poll Administration Menu.

The lifetime counter is designed to be a tally of all ballots printed by the machine since its manufacture. It is only displayed on the Poll Administration Menu, and the software does not provide a way to reset it.

Audit Logs The ICX audit logs record timestamped entries related to important events, such as opening or closing the poll, Poll Worker or Technician log-ins, attaching or detaching USB storage devices, software errors, etc. Although the time at which a ballot was displayed to a voter is recorded, the audit log does not contain information about the voters' selections. From the Technical Administration menu, the audit log can be viewed on-screen or exported to a USB drive for review by pressing the "Export Audit Log" button.

10.1 Vulnerable Storage Design

Issue: *ICX audit logs and protective counters are stored in regular files with no protection beyond filesystem permissions, which can be easily bypassed.*

Issue: *The ICX does not provide any mechanism to verify the integrity of exported audit logs.*

Internally, the audit logs and counters are simply files stored in the device's Android filesystem. Reverse-engineering of the ICX App shows that they are stored at these locations:

Redaction – Audit logs: [redacted]
Redaction – Public counter: [redacted]
Redaction – Lifetime counter: [redacted]

Access to these files is controlled using filesystem permissions, but the data they contain does not appear to be protected by any kind of encryption or cryptographic integrity mechanism, such as a MAC or a digital signature. Nor are the audit logs cryptographically protected when exported to a USB device.

To advance the counters, the ICX App simply reads, increments, and overwrites the values in the files. Similarly, to make a log entry, it simply appends

to the current day's log file. When exporting logs, the app merely packages the existing audit log files into a Zip file and writes it to the USB drive.

This leaves the counters and logs highly vulnerable to modification. As I explained in previous sections, weaknesses in the ICX allow attackers to easily gain root privileges, which lets them bypass all filesystem access controls. Consequently, attackers can arbitrarily edit or erase the audit logs, and they can change the protective counters to any value they choose.

10.2 Manual and Automated Modification

An attacker with physical access to the BMD can manipulate the logs and counters via several routes. First, they need to escape from the ICX App, using any of the methods described in Section 8. After accessing the underlying Android operating system, it is a simple matter to locate the applicable file and change its contents to suit the attacker's purposes. Attackers can do this either by installing malicious software that modifies the files automatically or by manually editing them using Android apps that are pre-installed on the ICX.

To give a concrete example, suppose the attacker wants to manipulate the ICX access log. They can hold the power button on the side of the machine to reboot it in "safe mode" (see Section 8.7), then open the File Manager app and navigate to the log file location shown above. By tapping on the log file icon, they can open it in the Android Text Editor app and simply use the touch-screen to select and delete arbitrary log entries.

Modifying the protective counters is slightly more involved due to the need to bypass filesystem permission checking. To do so, the attacker can open the pre-installed Terminal Emulator app and (using the on-screen keyboard or a physical keyboard), execute the `su` command to gain root privileges, as described in Section 8.3. They can then write new values to the counter files using any standard command-line method. I confirmed that this technique can successfully "roll back" the lifetime counter to a previous value, allowing the attacker to conceal having printed arbitrarily many ballots.

While I describe manual modification techniques here, malware can also obtain root privileges (see Section 9.3) and can be programmed to modify the logs and counters in an automated fashion. For example, malware could easily be programmed so that, on first run, it removed any log entries associated with its installation. Since modifying the log files would demonstrate no additional security insights beyond those required to install malware in the first place, I did not include such clean-up behavior in the proof-of-concept malware, but it would be a simple matter for a real attacker to do so.

REDACTED VERSION

11 Weaknesses in the ICP Scanner

The Dominion ImageCast Precinct (ICP) ballot scanner was not the focus of my investigation, and time constraints precluded conducting a complete security analysis of the device. Nevertheless, I did uncover some security problems related to the ICP, which I report in this section.

11.1 The ICP Accepts Photocopied Ballots

Issue: *The ICP as tested did not require ballots to be printed on security paper, and it accepted ICX ballots photocopied on normal office paper.*

Georgia uses special "security" paper stock for official ballots, including those printed by BMDs [32, 35]. However, when I tested the Fulton County ICP using ballots printed on normal copier paper, it accepted and counted them normally. I also tested scanning photocopies of BMD-printed ballots, and the ICP again accepted and counted them normally.

As Section 3.2 explains, the message authentication codes in the QR codes do not allow the scanners to distinguish between original and duplicate ballots, so, absent a check on the physical paper stock, the scanners cannot detect photocopied ballots.

Use of security paper *is* potentially valuable during a risk-limiting audit or a hand recount. Assuming access to such paper is carefully controlled, ballots printed on non-official paper could be detected during the auditing process. However, I note once again that Georgia requires risk-limiting audits of only once race in November elections of even numbered years, leaving other contests and elections potentially unprotected.

11.2 A Dishonest Poll Worker with Access to the ICP Memory Card can Deanonymize All Voted Ballots

Issue: *The ICP tested does not encrypt ballot images stored on its memory card.*

Issue: *ICP memory cards store ballot images in the order they were cast.*

The ICP stores a complete digital image of every scanned ballot on its removable memory card, and these images are returned to the EMS for possible later review or adjudication. On the Fulton County scanner I tested, the ballot images were not encrypted, and I could easily extract them. Moreover, my testing shows that the unencrypted ballot images are stored in the order in which they were cast, potentially deanonymizing the secret ballots.

Encrypting ballot images appears to be a configuration option that jurisdictions can enable. That option was not enabled in the ICP I tested, which was purportedly configured in the same way as the scanners used during Georgia elections. In any event, even if jurisdictions were to enable this encryption option, the county-wide encryption keys can be be extracted from any ICX Poll Worker Card, given brief access to the card and PIN (see Section 6.1).

I determined the ballot image storage format by examining what data on the memory card changed when I scanned an additional ballot. The ballot images are not stored as regular files in the card's filesystem. Rather, they are stored in a proprietary data structure in a secondary partition. [redacted] Redaction

Following this volume header, the ballot images are stored sequentially. [redacted] Redaction I created a Python program (`cfextract.py`) to extract the ballots images from the memory card, in the order they were voted, and output them as TIFF files.

Storing the ballots in voted-order raises serious risks to ballot secrecy. A dishonest poll worker could observe voters as they used the scanner and secretly note their names, in order. If, after voting was finished, the poll worker had brief access to the scanner memory card, they could read its contents with an inexpensive and widely available Compact Flash card reader, then use a program like mine to view all the ballots and associate each with the voter's identity.

11.3 Installed Tamper-Evident Seal could be Bypassed or Defeated

Issue: *The ICP modem port door is incompletely closed when sealed, allowing access to connectors inside.*

Issue: *The tamper-evident seal on the ICP tested was improperly installed, leaving it easily defeated.*

The Fulton County ICP was delivered to Plaintiffs with only one tamper-evident seal installed. On the right side of the ICP, a plunger-style security seal was affixed to a small plastic door that the ICP User Guide refers to as the "Modem Port" [21, p. 11], which covers an RJ45 Ethernet port and a USB Type-A port. The seal, Intab part number 03-1366 [50], consists of a braided wire that passes through a metal loop in the machine's case, preventing the door from being fully opened. The sealed door, as we received it, is shown in Figure 14a.

One problem with this sealing arrangement is that, by applying tension to the door, it can be opened several millimeters without removing the seal. As shown in the figure, this is sufficient access to see both ports, and an attacker could almost certainly attach electronic equipment to either port by inserting conductive probes through the gap in the door. The problem could have been avoided by using a different kind of seal. Dominion's manual states that "[a] lock, tamper evident label, or tamper evident tie wrap should be placed on the door lock loop" [21, p. 49], but the seal that was installed is a wire seal, which is thinner and more flexible than a typical tie wrap, allowing more play.

Furthermore, the seal was improperly installed and could easily be removed without breaking it. According to the manufacturer's instructions, when installing the seal, the metal plunger needs to be fully depressed into the seal housing. In the condition we received it, the plunger was incompletely inserted, as shown in

REDACTED VERSION



(a) ICP sealed door, as received (b) Defeating the seal

Figure 14: **Defeating the ICP Tamper-Evident Seal.** The ICP scanner from Fulton County used a plunger-style wire seal to guard access to the "Modem Port". Even with the seal installed, an attacker could open the door enough to access the telephone and USB ports inside it (*left*). While the seal's documentation [50] shows that a properly installed seal will have the metal plunger fully inserted (*top*), the seal as installed only had the plunger partially depressed (*middle*). This allowed easy removal of the seal in such a way that it could be reattached without leaving any visible evidence of tampering (*bottom*).

Figure 14b. I watched as my assistant used his bare fingers to grasp the plunger and simply pull it out of the seal's plastic housing. With the plunger removed, he was able to free the wire from the seal housing using a gentle tugging motion, thus removing the seal and allowing the door to fully open.

After inspecting the internals of the seal's housing, I determined that the wire running through the metal hasp had been only slightly bent due to the incomplete insertion of the plunger. This allowed the seal to be removed without damaging any of its components. It would be possible to reaffix it without leaving any obvious signs that it had been breached.

That Fulton County election workers selected an inappropriate seal *and* failed to properly install it—on a scanner they knew would be subjected to security testing—suggests that Georgia security seal practices are insufficient to reliably protect the state's election equipment from undetected physical access.

REDACTED VERSION

Expert Qualifications

My name is J. Alex Halderman. I am Professor of Computer Science and Engineering, Director of the Center for Computer Security and Society, and Director of the Software Systems Laboratory at the University of Michigan in Ann Arbor. I hold a Ph.D. (2009), a master's degree (2005), and a bachelor's degree (2003), *summa cum laude*, in computer science, all from Princeton University. My background, qualifications, and professional affiliations are set forth in my *curriculum vitae*, which is available online at https://alexhalderman.com/home/halderman-cv.pdf.

My research focuses on computer security and privacy, with an emphasis on problems that broadly impact society and public policy. Among my areas of research are software security, network security, computer forensics, and election cybersecurity. I have authored more than 90 articles and books, and my work has been cited in more than 12,000 scholarly publications. I have served as a peer-reviewer for more than 35 research conferences and workshops.

I have published numerous peer-reviewed research papers analyzing security problems in electronic voting systems used in U.S. states and in other countries. I have also investigated methods for improving election security, such as efficient techniques for auditing whether computerized election results match paper ballots. I regularly teach courses in computer security, network security, and election cybersecurity at the graduate and undergraduate levels. I am the creator of Securing Digital Democracy, a massive, open, online course about computer security and elections that has attracted more than 20,000 students.

I serve as co-chair of the State of Michigan's Election Security Advisory Commission, by appointment of the Michigan Secretary of State. I have also performed security testing of electronic voting systems for the Secretary of State of California. I have testified before the U.S. Senate Select Committee on Intelligence and before the U.S. House Appropriations Subcommittee on Financial Service and General Government on the subject of cybersecurity and U.S. elections.

I received the John Gideon Award for Election Integrity from the Election Verification Network, the Andrew Carnegie Fellowship, the Alfred P. Sloan Foundation Research Fellowship, the IRTF Applied Networking Research Prize, the Eric Aupperle Innovation Award, the University of Michigan College of Engineering 1938E Award for teaching and scholarship, and the University of Michigan President's Award for National and State Leadership.

Affirmation

I declare under penalty of the perjury laws of the State of Georgia and the United States that the foregoing is true and correct, and that this report was executed this 1st day of July, 2021.

J. Alex Halderman

REDACTED VERSION

References

[1] Android Developers. *Android Lollipop*. 2014. URL: https://developer.android.com/about/versions/lollipop.

[2] Android Developers. *Android Runtime (ART) and Dalvik*. URL: https://source.android.com/devices/tech/dalvik.

[3] Android Developers. *Codenames, Tags, and Build Numbers*. URL: https://source.android.com/setup/start/build-numbers#usecase-codename-references-and-resources.

[4] Android Developers. *Dedicated devices overview*. https://developer.android.com/work/dpc/dedicated-devices. July 2020.

[5] Android Developers. *Recents Screen*. URL: https://developer.android.com/guide/components/activities/recents.

[6] Android Developers. *UserManager*. Apr. 2021. URL: https://developer.android.com/reference/android/os/UserManager.

[7] A. W. Appel. "Security seals on voting machines: A case study." In: *ACM Transactions on Information and System Security* 14.2 (Sept. 2011). URL: https://www.cs.princeton.edu/~appel/voting/SealsOnVotingMachines.pdf.

[8] Avalue. *HID-21V-BTX*. URL: https://www.avalue-tech.com/products/Panel-PC/Industrial-Panel-PC/Light-Industrial-Panel-PC/HID-21V-BTX_2758.

[9] Matthew Bernhard, Allison McDonald, Henry Meng, Jensen Hwa, Nakul Bajaj, Kevin Chang, and J. Alex Halderman. "Can Voters Detect Malicious Manipulation of Ballot Marking Devices?" In: *41st IEEE Symposium on Security and Privacy*. May 2020. https://doi.org/10.1109/SP40000.2020.00118.

[10] D. Bowen et al. *Top-to-Bottom Review of Voting Machines Certified for Use in California*. Tech. rep. https://www.sos.ca.gov/elections/ovsta/frequently-requested-information/top-bottom-review/. California Secretary of State, 2007.

[11] Joseph A. Calandrino, Ariel J. Feldman, J. Alex Halderman, David Wagner, Harlan Yu, and William Zeller. *Source Code Review of the Diebold Voting System*. Part of the California Secretary of State's "Top-to-Bottom" Voting Systems Review. July 2007. URL: https://votingsystems.cdn.sos.ca.gov/oversight/ttbr/diebold-source-public-jul29.pdf.

[12] California Secretary of State. *Conditional Approval of Dominion Voting Systems, Inc. Democracy Suite Version 5.10*. Oct. 2019. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds510-cert.pdf.

[13] California Secretary of State. *Conditional Approval of Dominion Voting Systems, Inc. Democracy Suite Version 5.2*. Oct. 2017. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds52-cert.pdf.

[14] Cyber Ninjas. *Arizona Audit Statement of Work*. Mar. 31, 2021. URL: https://www.clerkofcourt.maricopa.gov/home/showpublisheddocument/2557/637551085573500000.

[15] Cybersecurity & Infrastructure Security Agency. *Supply Chain Compromise*. URL: https://www.cisa.gov/supply-chain-compromise.

[16] Defense Human Resources Activity. *Common Access Card (CAC)*. https://www.cac.mil/common-access-card/.

[17] Matthew S. DePerno et al. *Plaintiff's Collective Response to Defendants' and Non-Party Counties' Motions to Quash and For Protective Orders. Bailey v. Antrim County*, Michigan Circuit Court for the County of Antrim, Case No. 20-9238-CZ. Apr. 9, 2021. URL: https://www.depernolaw.com/uploads/2/7/0/2/27029178/ex_5-10.pdf.

[18] Android Developers. *Dedicated devices cookbook*. https://developer.android.com/work/dpc/dedicated-devices/cookbook. July 2020.

[19] Dominion Voting Systems. *2.02—Democracy Suite System Overview*. STATE-DEFENDANTS-00047612 *et seq.*

[20] Dominion Voting Systems. *Democracy Suite Election Management System*. URL: https://www.dominionvoting.com/democracy-suite-ems/.

[21] Dominion Voting Systems. *Democracy Suite ImageCast Precinct User Guide (Version 5.5-A.GA::59)*. STATE-DEFENDANTS-00047951 *et seq.*

[22] Dominion Voting Systems. *ICX behavior when powering on in "safe mode"*. Customer advisory notice. Jan. 2020. URL: https://www.eac.gov/sites/default/files/2020-09/ICX%20Safe%20ModevFINAL.pdf.

[23] Dominion Voting Systems. *ImageCast Precinct*. URL: https://www.dominionvoting.com/imagecast-precinct/.

[24] Dominion Voting Systems. *ImageCast Remote Brochure*. URL: https://www.votescount.us/Portals/16/New%20voting%20system/ImageCast%20Remote%20Brochure%20FINAL.pdf.

[25] Dominion Voting Systems. *ImageCast X*. URL: https://www.dominionvoting.com/imagecast-x/.

[26] Jeremy Duda and Jim Small. *Arizona Senate hires a "Stop the Steal" advocate to lead 2020 election audit*. Arizona Mirror. Mar. 31, 2021. URL: https://www.azmirror.com/2021/03/31/arizona-senate-hires-a-stop-the-steal-advocate-to-lead-2020-election-audit/.

[27] Jose Esparza. *Report of Review of Dominion Voting Systems Democracy Suite 5.5*. June 2019. URL: https://www.sos.texas.gov/elections/forms/sysexam/dominion-democracy-suite-5.5.pdf.

[28] Jose Esparza. *Report of Review of Dominion Voting Systems Democracy Suite 5.5–A*. Jan. 2020. URL: https://www.sos.texas.gov/elections/forms/sysexam/dominion-d-suite-5.5-a.pdf.

[29] Nicolas Falliere, Liam O Murchu, and Eric Chien. *W32.Stuxnet Dossier*. Feb. 2011. URL: https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf.

[30] P. Faltstrom, F. Ljunggren, and D. van Gulik. *The Base45 Data Encoding*. IETF Internet-Draft. June 14, 2021. URL: https://datatracker.ietf.org/doc/draft-faltstrom-base45/.

[31] Ariel J. Feldman, J. Alex Halderman, and Edward W. Felten. "Security analysis of the Diebold AccuVote-TS voting machine." In: *USENIX/ACCU-*

RATE Electronic Voting Technology Workshop. Aug. 2007. https://www.usenix.org/legacy/events/evt07/tech/full_papers/feldman/feldman.pdf.

[32] Georgia Secretary of State. *Amendment 1 to the Master Solution Purchase and Service Agreement between Dominion Voting Systems, Inc and the Secretary of State of the State of Georgia*. July 2019. URL: https://sos.ga.gov/admin/uploads/Dominion_Contract-_Amendment_1-_Executed.pdf.

[33] Georgia Secretary of State. *Democracy Suite 5.5–A Certification*. Aug. 2019. https://sos.ga.gov/admin/uploads/Dominion_Certification.pdf.

[34] Georgia Secretary of State. *Election Security is our Top Priority*. URL: https://sos.ga.gov/securevoting/.

[35] Georgia Secretary of State. *Master Solution Purchase and Service Agreement by and between Dominion Voting Systems, Inc as Contractor, and Secretary of State of the State of Georgia as State*. July 2019. URL: https://sos.ga.gov/admin/uploads/Contract.zip.

[36] Google. *Find problem apps by rebooting to safe mode on Android*. Android Help. URL: https://support.google.com/android/answer/7665064.

[37] Hak5. *Bash Bunny*. https://shop.hak5.org/products/bash-bunny.

[38] Hak5. *O.MG Cable – * to USB-A*. URL: https://shop.hak5.org/products/o-mg-cable-usb-a.

[39] J. Alex Halderman. *Declaration*. Dkt. 682. Dec. 16, 2019.

[40] J. Alex Halderman. *Declaration*. Dkt. 855. Sept. 1, 2020.

[41] J. Alex Halderman. *Declaration*. Dkt. 923-1. Sept. 29, 2020.

[42] J. Alex Halderman. "Practical Attacks on Real-world E-voting." In: *Real-World Electronic Voting: Design, Analysis and Deployment*. Ed. by Feng Hao and Peter Y. A. Ryan. 2016, pp. 145–171.

[43] Rosalind Helderman. *Arizona's Maricopa County will replace voting equipment, fearful that GOP-backed election review has compromised security*. June 2021. URL: https://www.washingtonpost.com/politics/arizona-maricopa-2020-audit-review/2021/06/28/98da5e64-d863-11eb-9bbb-37c30dcf9363_story.html.

[44] Ian Hoffman. *Scientists Call Diebold Security Flaw 'Worst Ever'*. Inside Bay Area. 2006. URL: https://www.eastbaytimes.com/2006/05/11/scientists-call-diebold-security-flaw-worst-ever-2/.

[45] Harri Hursti. *Critical Security Issues with Diebold TSx*. Black Box Voting. 2006. URL: https://web.archive.org/web/20120623161935/http://www.bbvdocs.org/reports/BBVreportIIunredacted.pdf.

[46] iBotPeaches. *Apktool*. Github. URL: https://github.com/iBotPeaches/Apktool.

[47] *iButton – iButton Devices – One Wire*. URL: https://www.maximintegrated.com/en/products/ibutton-one-wire/ibutton.html.

[48] Robert S. Mueller III. *Report on the Investigation into Russian Interference in the 2016 Presidential Election (Volume I of II)*. United States Department of Justice. Mar. 2019. URL: https://www.justice.gov/storage/report.pdf.

[49] *Indictment, United States v. Netyksho*. No. 1:18-cr-00215-ABJ, D.D.C. July 3, 2018.

[50] Intab. *Combo Seals*. https://www.intab.net/Combo-Seals/productinfo/03-1366/.

[51] izgzhen. *java2smali*. Github. URL: https://github.com/izgzhen/java2smali.

[52] JesusFreke. *smali*. Github. URL: https://github.com/JesusFreke/smali.

[53] Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach. "Analysis of an Electronic Voting System." In: *IEEE Symposium on Security and Privacy*. 2004, p. 27. URL: https://doi.org/10.1109/SECPRI.2004.1301313.

[54] Philip Kortum, Michael D. Byrne, and Julie Whitmore. "Voter Verification of BMD Ballots Is a Two-Part Question: Can They? Mostly, They Can. Do They? Mostly, They Don't." In: *Election Law Journal* (Dec. 17, 2020). URL: https://www.liebertpub.com/doi/full/10.1089/elj.2020.0632.

[55] Susan Lapsley. *Letter to Waldeep Singh*. July 2020. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds510a/ds510a-approval.pdf.

[56] Majority Staff of the Committee on Transportation and Infrastructure. *The Design, Development & Certification of the Boeing 737 MAX*. Sept. 2020. URL: https://transportation.house.gov/download/20200915-final-737-max-report-for-public-release&download=1.

[57] P. McDaniel, M. Blaze, and G. Vigna. *EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing*. Tech. rep. https://www.eac.gov/assets/1/28/EVEREST.pdf. Ohio Secretary of State, 2007.

[58] National Academies of Sciences, Engineering, and Medicine. *Securing the Vote: Protecting American Democracy*. Washington, DC: The National Academies Press, 2018. ISBN: 978-0-309-47647-8. URL: https://www.nap.edu/catalog/25120/securing-the-vote-protecting-american-democracy.

[59] National Institute of Standards and Technology. *Open Ended Vulnerability Testing for Software Independent Voting Systems*. 2007. URL: https://www.nist.gov/system/files/documents/itl/vote/OEVT.pdf.

[60] National Security Agency. *NSA ANT Catalog*. Media leak. URL: https://www.eff.org/files/2014/01/06/20131230-appelbaum-nsa_ant_catalog.pdf.

[61] National Security Agency. *NSA Cybersecurity Advisory: Malicious Actors Abuse Authentication Mechanisms to Access Cloud Resources*. Dec. 2020. URL: https://www.nsa.gov/News-Features/Feature-Stories/Article-View/Article/2451159/nsa-cybersecurity-advisory-malicious-actors-abuse-authentication-mechanisms-to/.

[62] National Security Agency. *Stealthy Techniques Can Crack Some of SIGINT's Hardest Targets*. Media Leak. Original: http://www.spiegel.de/media/media-35669.pdf. Alternative: https://www.aclu.org/foia-document/stealthy-techniques-can-crack-some-sigints-hardest-targets.

[63] Office of the Director of National Intelligence. *Statement by NCSC Director William Evanina: Election Threat Update for the American Public*. Aug. 7, 2020. URL: https://www.dni.gov/index.php/newsroom/press-releases/item/2139-statement-by-ncsc-director-william-evanina-election-threat-update-for-the-american-public.

[64] Pennsylvania Department of State. *Report Concerning the Examination Results of Dominion Voting Systems Democracy Suite 5.5A*. Jan. 2019. URL: https://www.dos.pa.gov/VotingElections/Documents/Voting%20Systems/Dominion%20Democracy%20Suite%205.5-A/Dominion%20Democracy%20Suite%20Final%20Report%20scanned%20with%20signature%20020119.pdf.

[65] Pro V&V. *Dominion Democracy Suite 5.10-A Software Test Report for the State of California*. June 2020. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds510a/provv-source.pdf.

[66] Pro V&V. *Dominion Voting Systems D–Suite 5.5–A Voting System: Georgia State Certification Testing*. Aug. 2019. https://sos.ga.gov/admin/uploads/Dominion_Test_Cert_Report.pdf.

[67] Pro V&V. *Test Report for EAC 2005 VVSG Certification Testing: Dominion Voting Systems Democracy Suite (D-Suite) Version 5.5 Voting System*. Aug. 2018. https://www.eac.gov/sites/default/files/voting_system/files/Dominion_Voting_Systems_D-Suite_5.5_Test_Report_Rev_A.pdf.

[68] Russell Ramsland. *Antrim Michigan Forensics Report (v2)*. Dec. 2020. URL: www.13thcircuitcourt.org/DocumentCenter/View/15743/Antrim-Michigan-Forensics-Report-12-13-2020--v2-REDACTED.

[69] Rules and Regulations of the State of Georgia. *Rule 183-1-15-.04. Audit*. URL: https://rules.sos.ga.gov/GAC/183-1-15-.04?urlRedirected=yes&data=admin&lookingfor=183-1-15-.04.

[70] Paul Sabanal. *Hiding Behind Android Runtime (ART)*. Black Hat Asia 2015. URL: https://www.blackhat.com/asia-15/briefings.html#hiding-behind-android-runtime-art.

[71] Paul Sabanal. *Hiding Behind Android Runtime (ART)*. Black Hat Asia 2015. Presentation slides. URL: https://www.blackhat.com/docs/asia-15/materials/asia-15-Sabanal-Hiding-Behind-ART.pdf.

[72] David E. Sanger. *Obama Order Sped Up Wave of Cyberattacks Against Iran*. The New York Times. June 2012. URL: http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html.

[73] Scandit AG. *Scandit Barcode Scanner*. Apple App Store. URL: https://apps.apple.com/us/app/scandit-barcode-scanner/id453880584.

[74] SLI Compliance. *Certification Test Report—Modification: Democracy Suite 5.5–A*. Dec. 2018. https://www.eac.gov/sites/default/files/voting_system/files/Dominion_Voting_Systems_D-Suite_5.5-A_Test_Report_v1.1.pdf.

[75] SLI Compliance. *Dominion Democracy Suite 5.10 Security and Telecommunications Test Report*. Aug. 2019. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/dvs510security-report.pdf.

[76] SLI Compliance. *Dominion Democracy Suite 5.10 Voting System Software Test Report for California Secretary of State*. Aug. 2019. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/dvs510software-report.pdf.

[77] SLI Compliance. *Dominion Democracy Suite 5.2 Security and Telecommunications Test Report*. 2017. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds52-sectel.pdf.

[78] SLI Compliance. *Dominion Democracy Suite 5.2 Source Code Test Report for California*. 2017. URL: https://votingsystems.cdn.sos.ca.gov/vendors/dominion/ds52-sc.pdf.

[redacted]

[80] Philip B. Stark and Ran Xie. *Testing Cannot Tell Whether Ballot-Marking Devices Alter Election Outcomes*. July 2020. URL: https://arxiv.org/pdf/1908.08144.pdf.

[81] Snyk Security Team. *Zip Slip*. https://res.cloudinary.com/snyk/image/upload/v1528192501/zip-slip-vulnerability/technical-whitepaper.pdf. June 2018.

[82] *Transcript of Status Conference Proceeding*. Dkt. 722. Mar. 6, 2020.

[83] Greg Ungerer. *[uClinux-dev] [ANNOUNCE] uClinux-dist-20070130 release*. Google Web Cache. Feb. 2007. URL: http://mailman.uclinux.org/pipermail/uclinux-dev/2007-February/041846.html.

[84] United States Environmental Protection Agency. *Volkswagen Violations*. URL: https://www.epa.gov/vw/learn-about-volkswagen-violations.

[85] U.S. Election Assistance Commission. *Democracy Suite 5.5*. Sept. 2018. https://www.eac.gov/voting-equipment/democracy-suite-55.

[86] U.S. Election Assistance Commission. *Voluntary Voting System Guidelines 1.0*. 2005. URL: https://www.eac.gov/sites/default/files/eac_assets/1/28/VVSG.1.0_Volume_1.PDF.

[87] Verified Voting. *The Verifier*. URL: https://verifiedvoting.org/verifier/.

[88] Verified Voting. *The Verifier: Dominion Voting Systems (November 2020)*. URL: https://verifiedvoting.org/verifier/#mode/search/year/2020/make/Dominion%20Voting%20Systems.

[89] Verified Voting. *The Verifier: ImageCast X (November 2020)*. URL: https://verifiedvoting.org/verifier/#mode/search/year/2020/model/ImageCast%20X.

[90] Ryan Whitwam. *Lollipop Feature Spotlight: The Recent Apps List Now Persists Through Reboot*. Android Police. Oct. 17, 2014. URL: https://www.androidpolice.com/2014/10/17/lollipop-feature-spotlight-the-recent-apps-list-now-persists-through-reboot/.

[91] Paul Woolverton. *Fact check: QR codes on Georgia ballots record votes as cast*. USA Today. Jan. 2021. URL: https://www.usatoday.com/story/news/factcheck/2021/01/02/fact-check-qr-code-georgia-us-senate-votes-warnock-loeffler-perdue-ossoff/4115918001/.

[92] Kim Zetter. *An Unprecedented Look at Stuxnet, the World's First Digital Weapon*. Wired. Nov. 3, 2014. URL: https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/.

REDACTED VERSION

Exhibit A: October 2020 Software Update Instructions

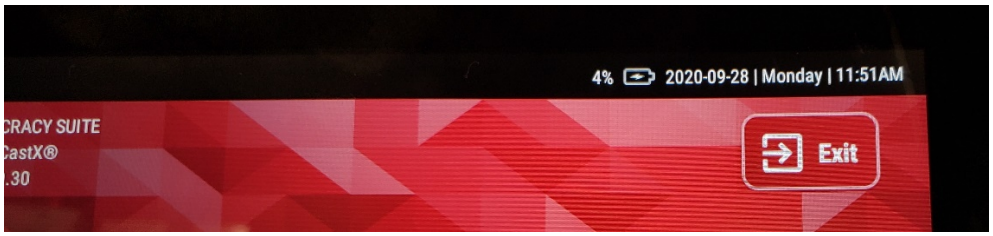
1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

Software Installation for ICX:

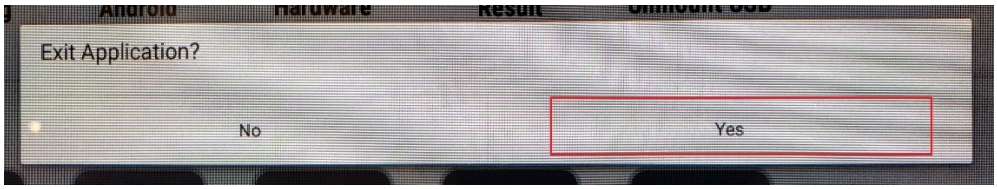
1. Press the power button to turn on the ICX.



2. Insert tech card, input password, then click login.
3. Click exit in the top right corner of the screen.



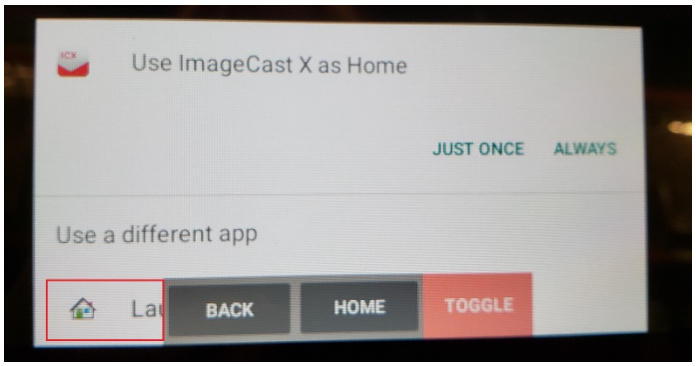
4. Click Yes when asked if you would like to exit the application.



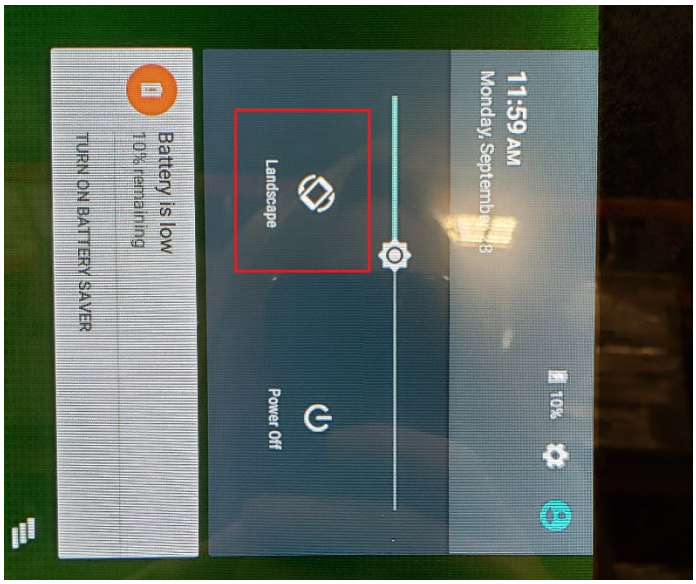
DOMINION VOTING

1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

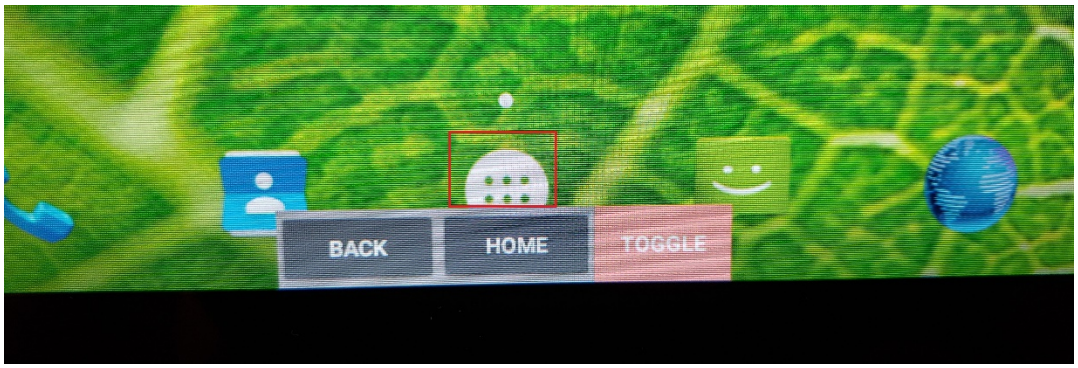
5. Click Launch *note: the launch button may be partially hidden by the back button.



6. Swipe left from the right side of the screen and click the rotate button 3 times. Swipe up when finished.

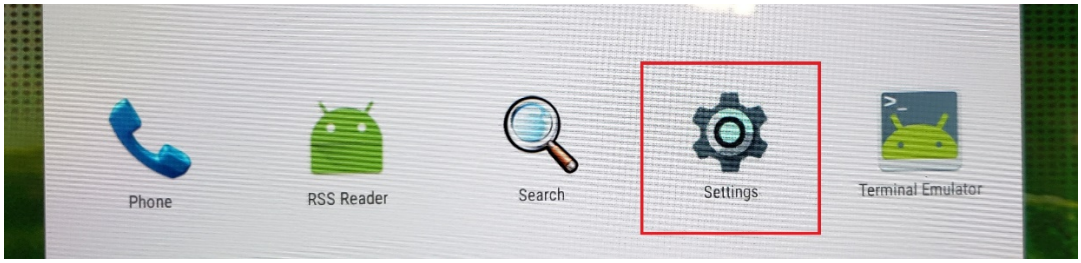


7. Click App Button

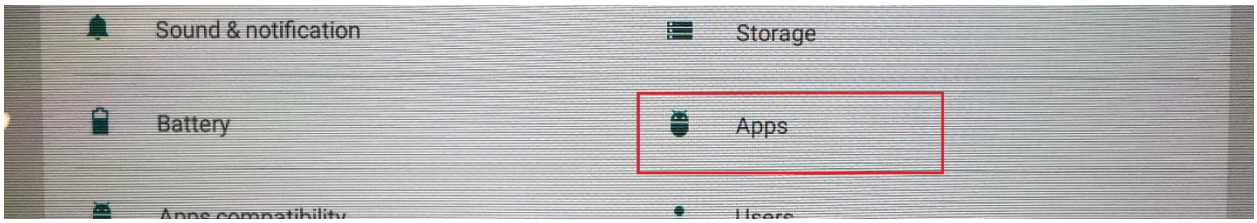


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

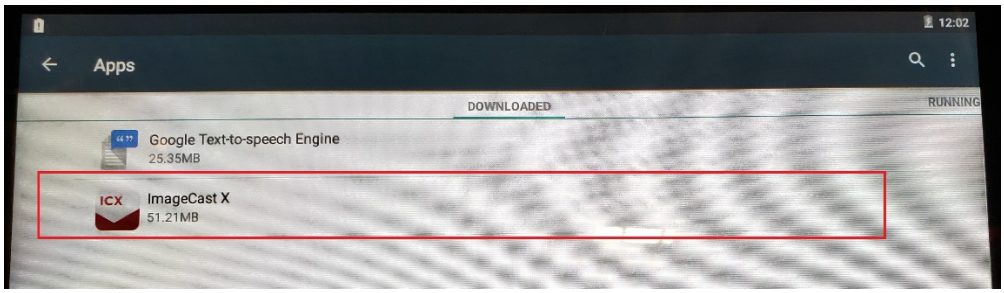
8. Click Settings



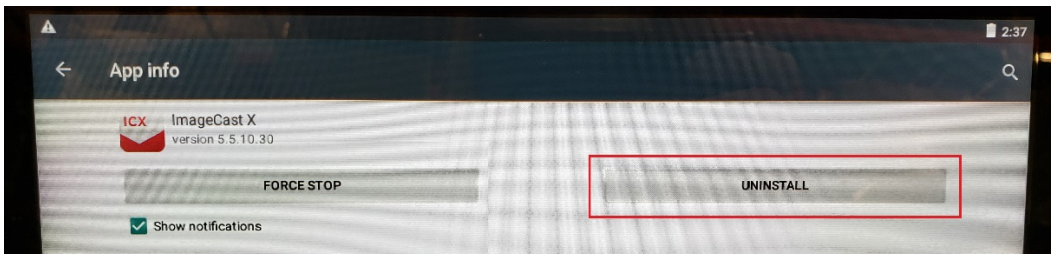
9. Click Apps



10. Click ImageCast X



11. Click Uninstall

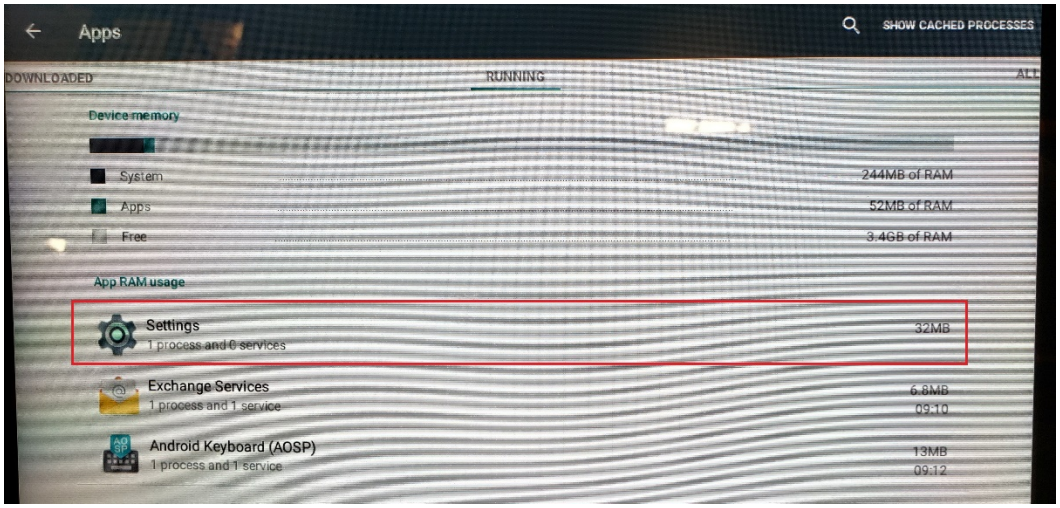


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

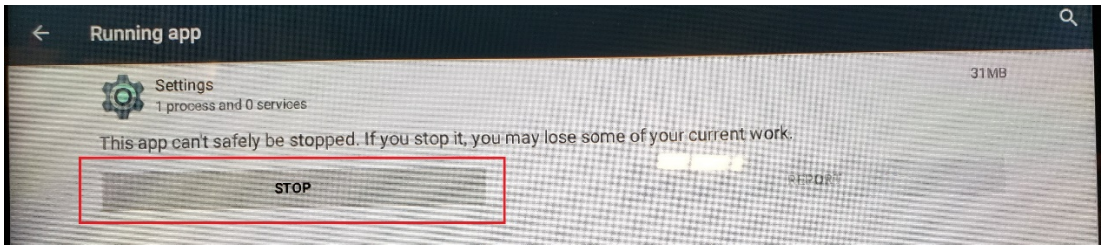
12. Click Ok



13. Swipe left to 'Running'. Click Settings

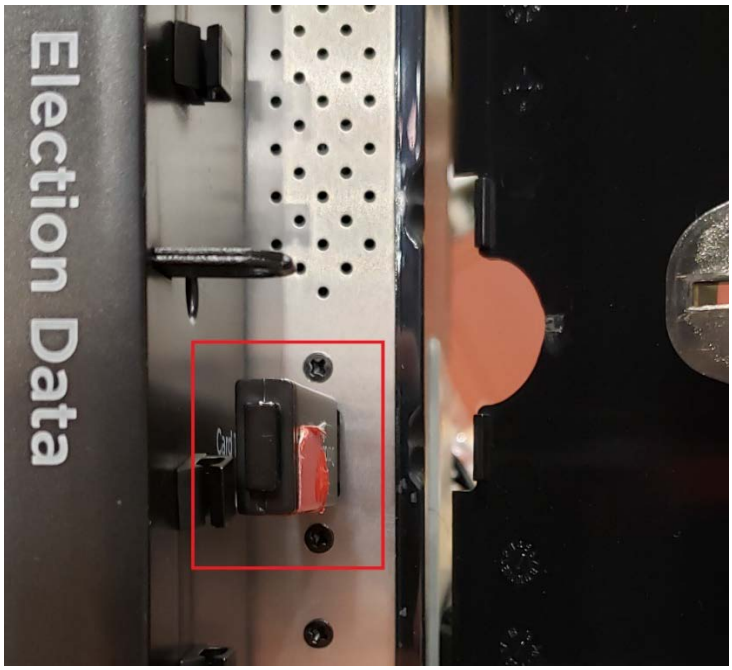


14. Click stop

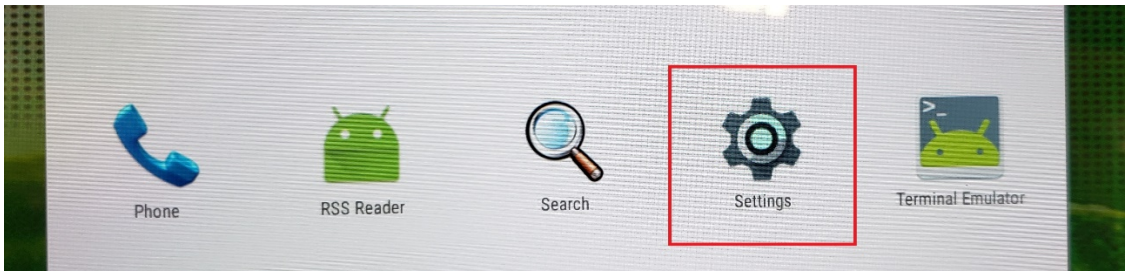


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

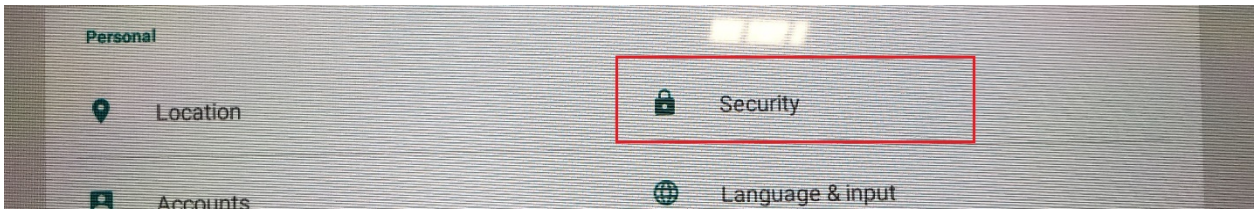
15. Insert USB



16. Click Settings



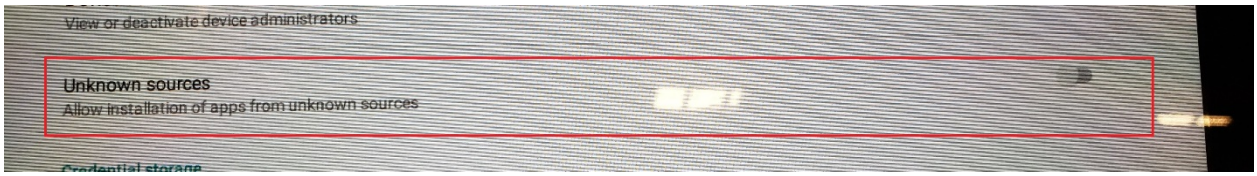
17. Click Security



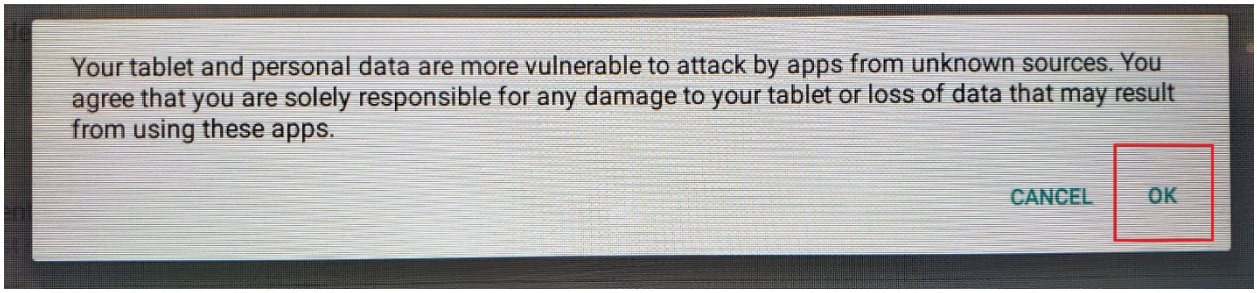
DOMINION VOTING

1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

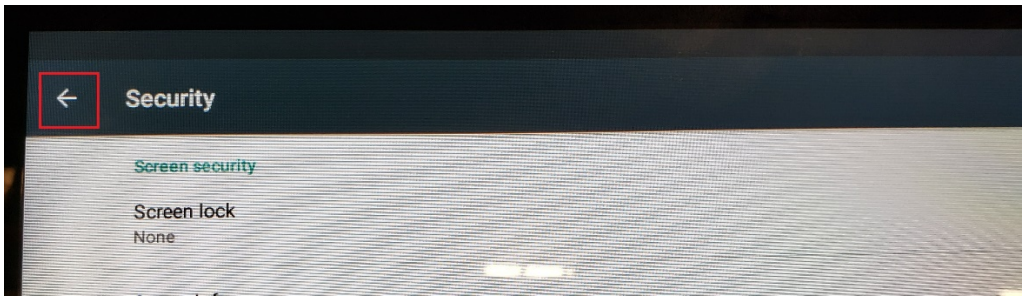
18. Toggle on Unknown Sources *Note: The toggle should be blue after toggling



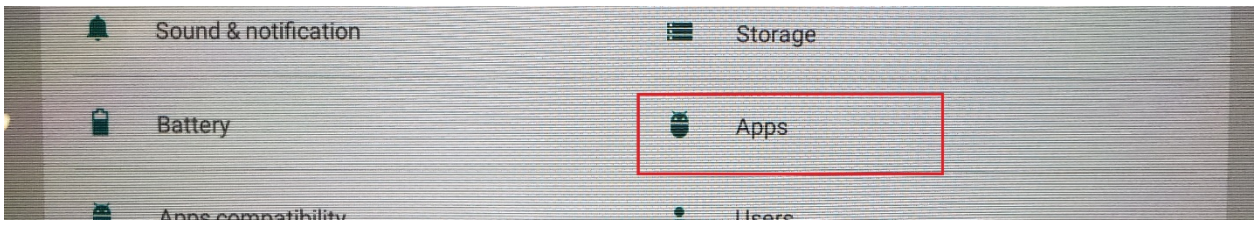
19. Click Ok



20. Click the back arrow in the top left corner.

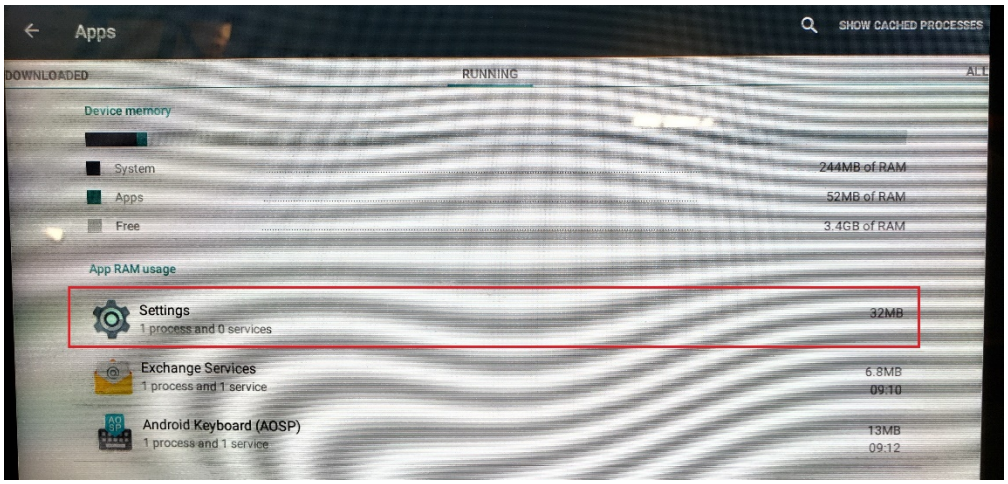


21. Click Apps

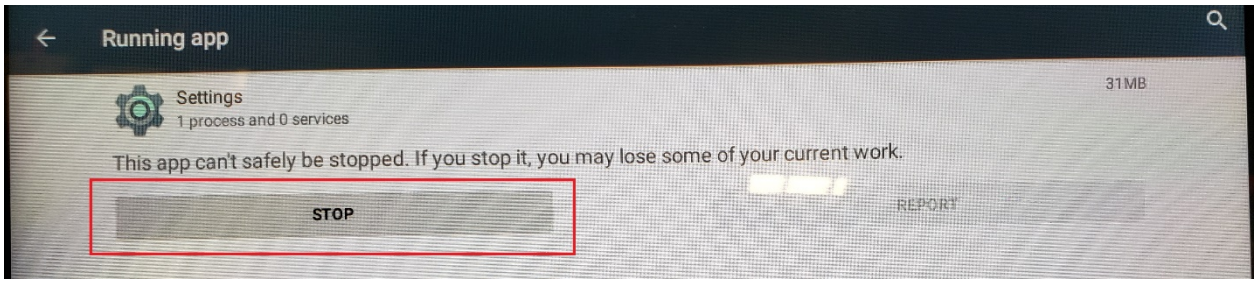


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

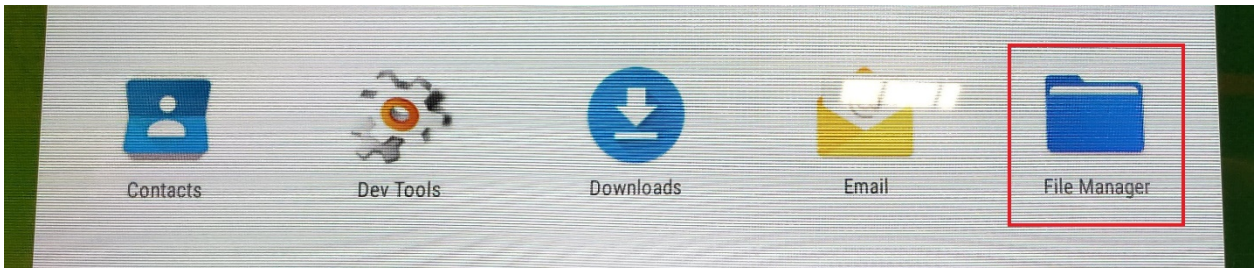
22. Swipe left to 'Running'. Click Settings



23. Click Stop

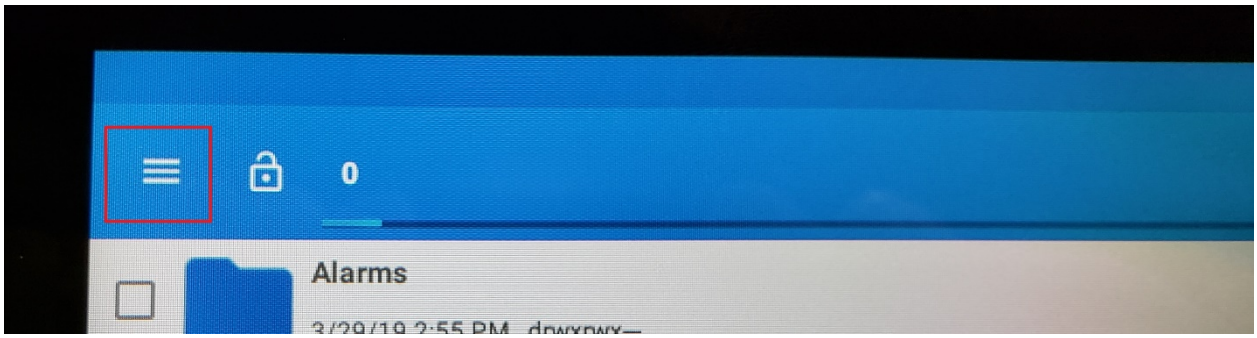


24. Click File Manager

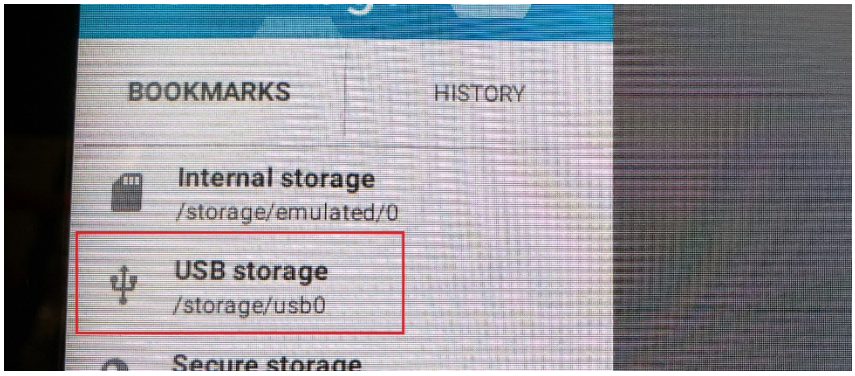


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

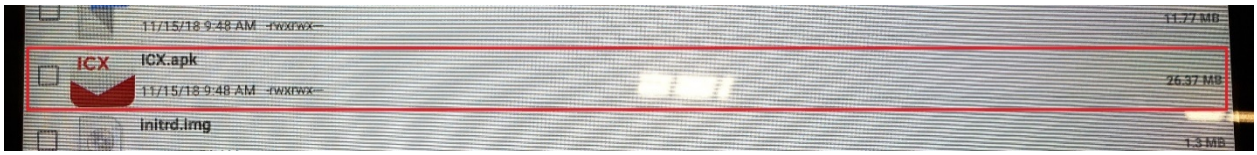
25. Click menu button in the top left corner.



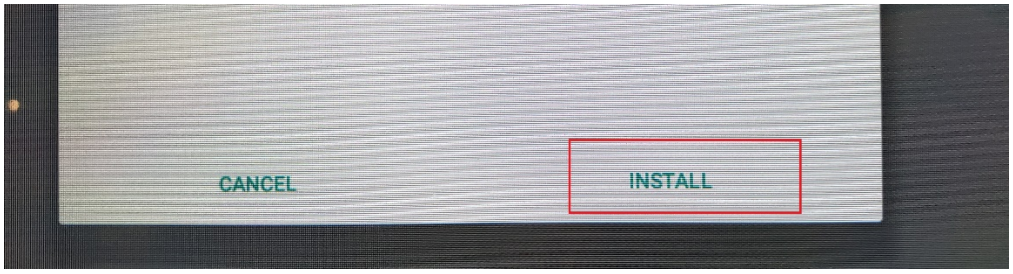
26. Select USB Storage



27. Click ICX.apk

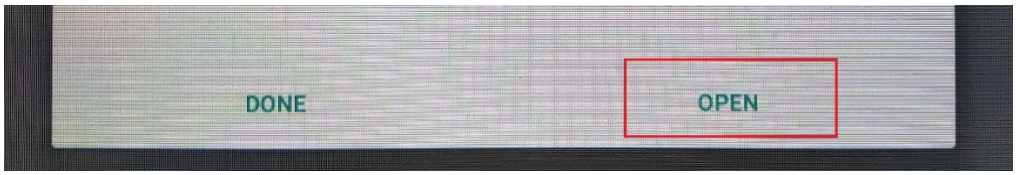


28. Click Install

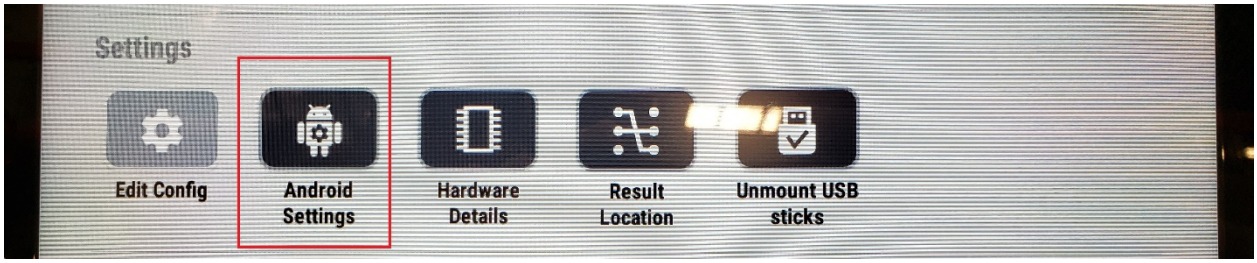


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

29. Click Open



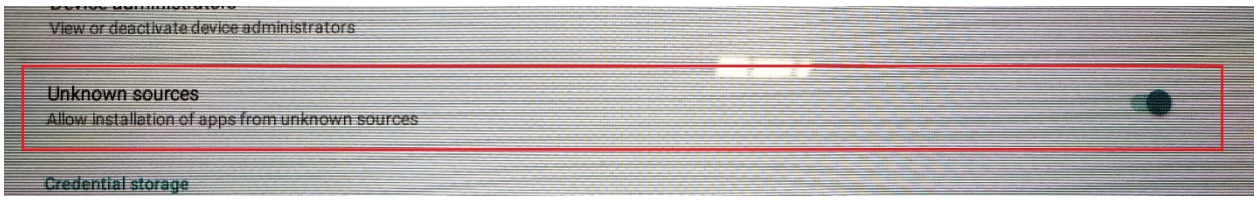
30. Input password and click login
31. Click Android Settings



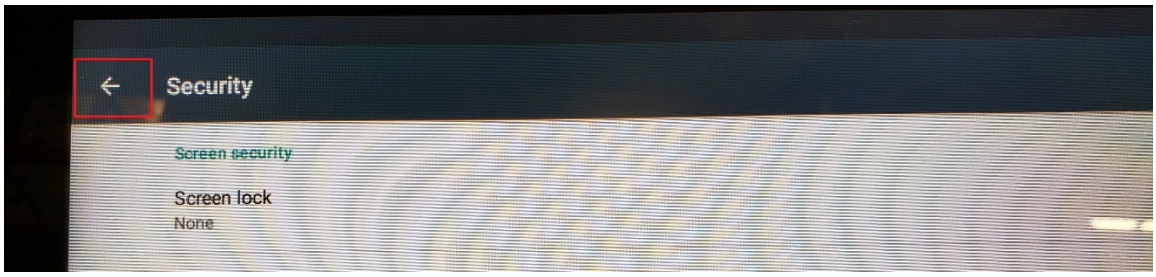
32. Click Security



33. Click to toggle off Unknown Sources *Note: The toggle should be gray when toggled off

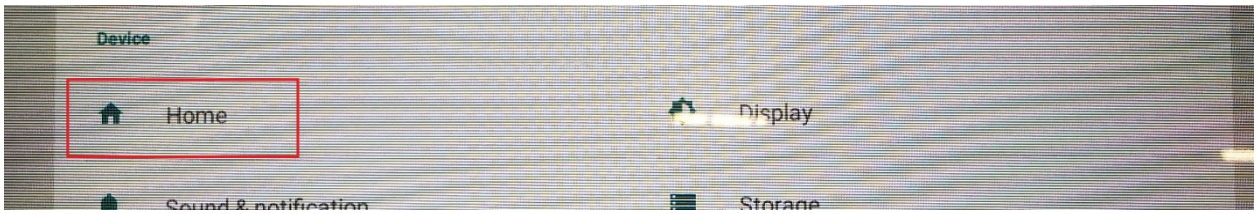


34. Click the back arrow in the top left corner

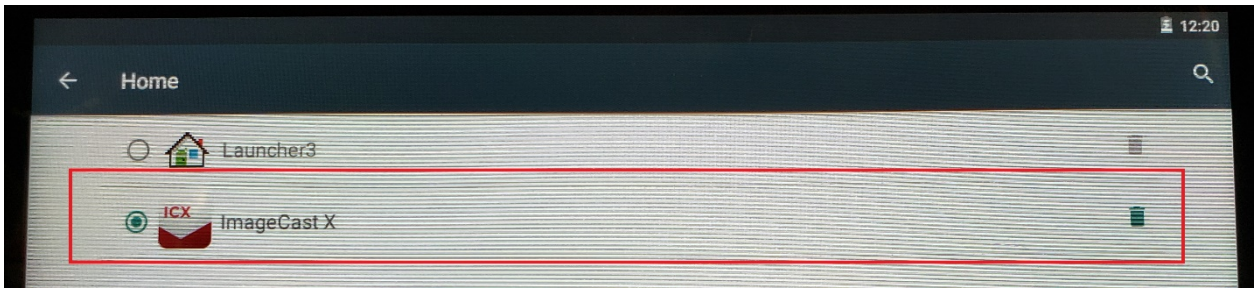


1201 18TH Street, Suite 210
DENVER, CO, 80202
1.866.654.8683
www.dominionvoting.com

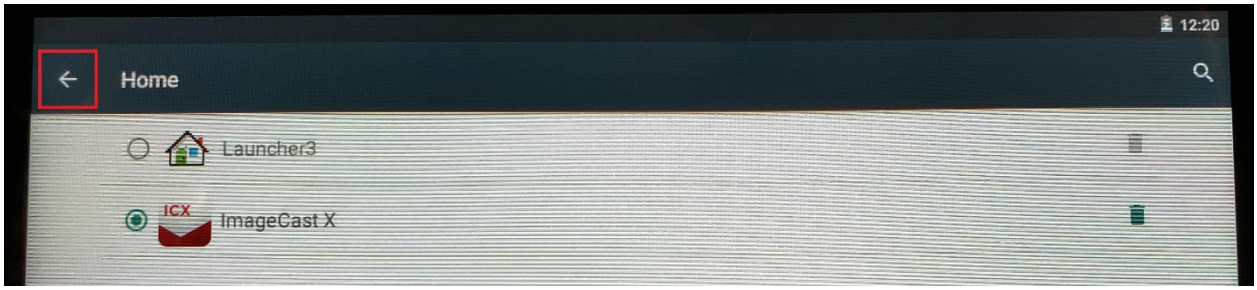
35. Click Home



36. Click Imagecast X



37. Click back arrow



38. Click the Home button at the bottom of the screen.



39. Remove the tech card and USB drive.
40. Software version should now read 5.5.10.32

REDACTED VERSION

Exhibit B: Georgia Logic and Accuracy Procedures



Logic and Accuracy Procedures

Version 1.0
Georgia Secretary of State – Brad Raffensperger
© January 2020

Logic and Accuracy Procedures

Items needed when testing:

From EMS workstation computer create the following items from the Election Project associated to the election for which testing is being conducted:

- Use Election Event Designer Application (EED) for the following:
  - Programmed Technician Card
  - *Programmed Poll Worker Card*
  - USB Drive containing information from GA ICX BMD programming group
  - *Print out of Ballot Activation Codes*
  - *Programmed Compact Flash Cards for Polling Place Scanner*
  - *Programmed Security Key Tab for Polling Place Scanner*
    - **Recommendation**: Create the * items above for each polling location and then use these to L&A test the equipment designated for the same polling place; at completion of L&A test on designated equipment package these items with the tested equipment for delivery to the designated polling place
- Provided by SOS after Election Project Obtained
  - Election Project User names and Passcodes
  - Technician Card Passcode
  - Poll Worker Card Passcode
  - Security Key Tab Passcode
  - Polling Place Scanner Re-zero Passcode
  - Poll Pad User name and Passcode
  - Poll Pad Menu Code

Testing Steps:

A. Preparing the BMDs

- Connect BMD to Printer
- Connect BMD and Printer to power supply
- First, Power Printer On
- Power the BMD On
- Verify installed version in top left corner of screen; v5.5.10.30
- Confirm presence of State Acceptance Test Sticker and seal on top left of BMD
- Insert Technician Card and enter passcode for specific election
- Verify date and time are properly set
  - If time or date needs to be adjusted, touch Modify and set the time and date
  - If time and date are correct, touch Confirm

- Touch Clear All Election Data
- Touch Yes
- Enter passcode
- Touch OK
- Insert USB Drive into an available USB slot in the Election Data compartment of the BMD
- Touch Load Election Data
- Select the data file to be loaded from the USB Drive
    - Touch Select
    - Touch Copy
    - Touch Ok
- Remove the USB Drive from the Election Data compartment of the BMD
- Close the Election Data compartment and attach seal, notate the number of the attached seal on paperwork
- Remove Technician Card
- Insert Poll Worker Card and passcode for specific election
- Touch Select Tabulator
- Select the BMD for the Polling location to which this BMD is being assigned
- Touch OK
- Touch Manual Selection Activation and confirm a checkmark appears in the box
- Touch AVS Controller and confirm a checkmark appears in the box
- Touch Open Polls
- Touch Yes
    - If Warning displayed regarding printer, confirm the Printer is connected and On
    - Touch OK
    - Touch Open Polls
    - Touch Yes
- Name of Polling Place BMD is assigned will display in Black on the top left of BMD Screen; confirm correct Polling Place shown
- Name of Election will display in Gray on the top left of BMD Screen; confirm correct Election shown
- Remove the Poll Worker Card
- Confirm Total Ballots Printed in bottom left corner shows zero (0)
- BMD is now ready

B. <u>Preparing the Polling Place Scanner</u>

- Insert the Primary Compact Flash Card into the Poll Worker Slot
- Insert the Backup Compact Flash Card into the Administrator Slot
- Confirm the Polling Place Scanner connected properly to the Ballot Box
- Confirm presence of State Acceptance Test Sticker on right side of scanner
- Power the Polling Place Scanner ON by plugging the Ballot Box into an AC power supply

- When the Polling Place Scanner begins to beep, beep, beep; align and carefully press down the Security Key Tab to the Security Key Slot
- When prompted on the screen, key in the passcode for the specific election, then press Enter
- Confirm Date and Time, modify the date and time if necessary
- Touch Utilities
- Touch Diagnostics
    - Touch Simple
    - Touch Yes after Thermal Printer test
    - Touch Print
    - Touch No
    - Review printed tape, confirm software version 5.5.3-0002
    - If any item in the Diagnostic test fails, do not proceed
- Touch Open Poll
    - Enter passcode for specific election, if prompted
- Touch Zero
    - Confirm tape shows zero results for all candidates in all races
    - If results are not zero, do not proceed
- Touch No for additional copies
- Confirm Polling Place Scanner shows zero (0) ballots cast
- Polling Place Scanner is now ready to accept ballots

C. <u>Preparing Poll Pads for BMD LA Testing</u>

- Notification of the LA/Advance Voting data set for Poll Pad along with a QR Code image for scanning by Poll Pad will be forwarded to those locations with a scheduled election
- Reference Poll Pad training documents and materials for assistance if the following steps need further explanation
- Power on Poll Pads to be used for LA/Advance Voting; this will not be ALL Poll Pads but only specific Poll Pads
- Connect designated Poll Pads to the appropriate connection
- Launch the Poll Pad application and scan the QR code image; follow prompts displayed on Poll Pad to obtain the Poll Pad LA/Acceptance Data set for the scheduled election
- Once download of the data file is complete, close the Poll Pad application
- Disconnect the Poll Pad from the appropriate connection
- Launch the Poll Pad application again
- Touch Get Started
- Enter User name and Passcode for specific election
- Touch Manual Entry
- Key in the Precinct Name or Precinct ID into the Last Name field
- Touch Search
- Touch the Precinct and Combo record desired and follow the prompts to create a voter card

- Create a voter card from Poll Pad for each unique ballot style within the designated Polling Location
    - Recommend labels be placed on card identifying what ballot style will be displayed by BMD once card is inserted
    - BMD removes the activation code from the Voter Card once used, therefore create the card again from Poll Pad after each use by a BMD

D. Testing the BMD and Printer

Use a combination of Poll Worker Card with Ballot Activation Codes for the polling location, and Voter Cards created from a Poll Pad loaded with the LA/Advance Voting dataset to bring up ballots on the BMD

- Produce at least one printed ballot from each BMD assigned to the polling location
- Produce a test deck from the BMDs assigned to the polling location for each unique ballot style within the polling location. The test deck must contain at least one vote for each candidate listed in each race within the unique ballot style
    - **Example**: Ballot from BMD 1 contains a vote for only the first candidate in each race listed on Ballot Style 1, Ballot from BMD 2 contains a vote only for the second candidate in each race on Ballot Style 1, and continue through the line of devices until all candidates in all races within the unique ballot style have received a single vote
    - **If Number of BMDs outnumber the number of vote positions on the unique ballot style**, start the vote pattern over until all BMDs have produced one printed ballot
    - **If Number of unique ballot styles in the polling place is greater than 1,** once the vote pattern is complete for a unique ballot style, proceed to the next BMD in line to start the review of the next unique Ballot Style
    - **All unique ballot styles do not have to be tested on each BMD**
- Review BMD-generated Test Deck and confirm the vote content before placing in the designated Polling Place Scanner

E. Testing the Polling Place Scanner

- Scan the BMD-generated Test Deck into the Polling Place Scanner
- Scan one blank optical scan ballot style(s) associated to the Polling Place to verify the Polling Place Scanner will recognize the ballot style in case of emergency
- Verify Scanner(s) shows a number of Ballot Cast equal to the number of ballots in the BMD-generated test deck plus the scanned blank Optical Scan ballot styles
- Firmly place the Security Key Tab in the Security Key Slot
- Touch Close Polls
- Enter the passcode
- Touch Enter
- Touch Yes
- Touch No for additional tapes (Scanner will automatically produce 3 copies of the closing tape)

- Review the results tape and confirm result printed matches the known vote content of the BMD-generated and Optical Scan ballot test deck
    - If results do not match, the scanner has failed, do not proceed
- Touch Power Down
- Touch Yes
- Unplug the Ballot Box from the AC power supply
- When the unit is OFF, open the Poll Worker card slot door and remove the Poll Worker Compact Flash Card
- The Poll Worker Compact Flash card for each Polling Place Scanner **MUST** be uploaded to the RTR application to confirm the Compact Flash card can be recognized and results transferred to the EMS for tabulation; then validate and publish the uploaded result file
- After the Compact Flash Card is uploaded to RTR, return the Compact Flash card to its designated Polling Place Scanner and re-insert it into the Poll Worker card slot

F. Preparing the BMD for Election

- Insert the Poll Worker Card and enter passcode
- Touch Admin Menu
- Touch Close Polls
- Touch Yes to confirm
- Touch Reset
- Touch Yes to confirm
- Enter Passcode for specific election
- Touch Ok to confirm
- Confirm Public Counter is at Zero (0); center of BMD screen
- Touch Power off, bottom right corner of screen
- Touch Yes to confirm
- Remove Poll Worker Card
- Turn Printer Off
- Disconnect the Power and Printer from BMD
- Close and seal the Power and Election Data compartments on the right side of the BMD
    - Make note of the seals attached
- Make the BMDs and Printers ready for delivery to the Polling Place

G. Preparing the Polling Place Scanner for Election

- Confirm that both the Poll Worker and Administrator Compact Flash cards are inserted into their assigned slots
- Power the Polling Place Scanner ON by plugging the Ballot Box into an AC power supply
- When the Polling Place Scanner begins to beep, beep, beep; align and carefully press down the Security Key Tab to the Security Key Slot

- When prompted on the screen, key in the needed passcode, then press Enter
- Confirm Date and Time, modify the date and time if necessary
- Touch Utilities
- Touch Re-Zero
- Enter Re-Zero passcode
- Confirm Ballot Cast shows as Zero (0)
- Touch Utilities
- Touch Report
- Touch Election Report
- Touch Zero
- Enter Number of Reports to print = 1
- Touch Enter
- Touch No for additional copies
- Confirm tape shows zero results for all candidates in all races
- Remove the tape and place with L&A Paperwork
- Touch Power Down
- Touch Yes
- Unplug Ballot Box from AC power supply
- Open Ballot Box and remove all test ballots from the Main bin, from the Write-In bin, and from the emergency bin
- Confirm that all bins are empty and properly secured
- Close and seal the ballot box, make note of the seals applied
- Place seals on the Poll Worker and Administrator Compact Flash Card doors, make note of the seals applied
- Make Ballot Box with attached Polling Place Scanner ready for delivery to the Polling Place

H. <u>Testing ICC Workstation and Central Scanner</u>

- Load ICC ABS tabulator Data Set to ICC workstation computer DVS folder
- Launch ICC Application
- Import Tabulator
- Attach Security Key Tab; Enter Passcode
- Enter Name of Project equal to name of ICC ABS tabulator
- Click Load
- Click Configuration
- Set secondary results path
- Verify Scanner is On and recognized by the ICC application
- Click Scan Options
- Set Ballot configuration to Dynamic
- Set scanner to Stop on Overvotes for ALL races

- Set Scanner Continuity to Continuous Scan
- Click OK
- Click Scanning
- Click YES
- Insert Test Deck with known result
- Click Scan
- Verify Scanner recognizes and scans all ballots within the test deck
- Verify Scanner recognizes any error ballots that may be included within the test deck
- Verify Scanner recognizes any overvotes
- Accept the Batch
- Close the ICC application
- Remove ICC ABS tabulator Data Set from ICC workstation computer DVS folder
- Open RTR Application on EMS Workstation
- Open election project
- Load Results from ICC via Secondary path established on ICC workstation
- Click Load
- Click Election Summary Report and generate Election Summary Report prior to validating and publish result file from ICC
- Click Result Files
- Click Search
- Select ICC ABS result file
- Click Validate and Publish
- Click Election Summary Report and generate Election Summary Report
- Verify Results shown for ABS match the known result of the Test Deck scanned by the ICC

I. Upload LA results to ENR

- After results have been loaded to RTR from the Polling Place and ICC scanners
- Create folder on the Desktop of EMS computer labeled State Export
- Open RTR, click Export
- Click Export Type
- Click Search
- Verify that ONLY the GA Export File type is active (contains a checkmark)
- In tool bar, click Settings>Transfer Points
- Click Add
- Click Browse; Select the folder on the Desktop labeled State Export
- In Connection Name type State Export
- Click OK
- Click Save
- Below Tool Bar, Click Start Results Export

- Minimize RTR
- Open State Export folder on Desktop
- Confirm Export file present in State Export folder
- Extract export file and upload to State ENR
- Open EED
- Create folder to Desktop labeled "*Name of Election*-Backups"
- Create a Backup copy of the Election Project
- Copy the saved Backup zip file and accompanying SHA file and place in Backups folder; copy the folder containing the backups to removable media
- Clear results from RTR
- Print new Election Summary Report from RTR confirming all LA results have been cleared
- Close RTR and EED on EMS workstation

J. <u>Loading Election Day Dataset to Poll Pad</u>

- Approximately one week prior to the scheduled Election Day, notification of Election Day data files for Poll Pad along with a QR Code image for scanning by Poll Pad will be forwarded to those locations with a scheduled election
- Power on Poll Pad
- Connect Poll Pads scheduled for use on Election Day to the appropriate connection
- Launch the Poll Pad application and scan the QR code image; follow prompts displayed on Poll Pad to obtain the Poll Pad Election Day Data set for the scheduled election
- Once download of the data file is complete, close the Poll Pad application
- Disconnect the Poll Pad from the appropriate connection
- Launch the Poll Pad application again
- Touch Get Started
- Enter the User name and Passcode for the specific election
- Confirm the proper Election and Polling Location are shown at the top of the screen
- Confirm the number of Precinct Records (voters assigned to location) is accurate
- Confirm Check-Ins are Zero (0)
- Connect Voter Card Encoder to Poll Pad and confirm encoder is recognized by Poll Pad (green indicator at top right of screen)
- Connect power cord to Voter Card Encoder and verify power flows through Voter Card Encoder and charges the Poll Pad
- Touch Scan Barcode
- Confirm camera is operational
- Touch Cancel
- Touch Manual Entry
- Key in last name of known voter in polling place
- Touch Search

- Touch selected voter record, confirm voter information shown is accurate
- Touch Accept
- Confirm Voter Certificate is displayed with signature line
- Put in example signature
- Touch Done Signing
- Confirm Poll Officer Initial box is operational
- Touch Submit
- Touch Touchscreen
- Insert Voter Card into Voter Card Encoder
- Verify Ballot Style and Ballot Activation Code display at bottom of screen
- Confirm Create Card button at top of screen becomes active
- Touch Create Card to verify Voter Card can be created
- Touch Manual Entry
- Find previous Voter
- Touch Wheel and Enter password; confirm password for specific election recognized
- Cancel Voter Check-in
- Spoil Ballot
- Verify mark has been removed
- Press iPad Home button to Close Poll Pad Application

K. <u>Loading Update File to Poll Pad</u>

- On the Saturday prior to the scheduled Election Day, notification of Election Day update data files for Poll Pad will be forwarded to those locations with a scheduled election
- Power on Poll Pad
- Connect Poll Pads scheduled for use on Election Day to the appropriate connection
- Launch the Poll Pad application and follow prompts displayed on Poll Pad to obtain the Poll Pad Election Day Data set for the scheduled election
- Once download of the data file is complete, close the Poll Pad application
- Disconnect the Poll Pad from the appropriate connection
- Launch the Poll Pad application again
- Touch Get Started
- Enter User name and Passcode for the specific election
- Confirm the proper Election and Polling Location are shown at the top of the screen
- Confirm the number of Precinct Records (voters assigned to location) is accurate
- Confirm Check-Ins are Zero (0)
- Connect Voter Card Encoder to Poll Pad and confirm encoder is recognized by Poll Pad (green indicator at top right of screen)
- Connect power cord to Voter Card Encoder and verify power flows through Voter Card Encoder and charges the Poll Pad

- Touch Menu
- Touch Summary Report
- Touch Absentees; confirm expected number of Absentee Voters for polling location
- Touch Home
- Touch Get Started
- Touch Scan Barcode
- Confirm camera is operational
- Touch Cancel
- Touch Manual Entry
- Key in last name of known voter in polling place
- Touch Search
- Touch selected voter record, confirm voter information shown is accurate
- Touch Accept
- Confirm Voter Certificate is displayed with signature line
- Put in example signature
- Touch Done Signing
- Confirm Poll Officer Initial box is operational
- Touch Submit
- Touch Touchscreen
- Insert Voter Card into Voter Card Encoder
- Verify Ballot Style and Ballot Activation Code display at bottom of screen
- Confirm Create Card button at top of screen becomes active
- Touch Create Card to verify Voter Card can be created
- Touch Manual Entry
- Find previous Voter
- Touch Wheel and Enter password; confirm password for specific election recognized
- Cancel Voter Check-in
- Spoil Ballot
- Verify mark has been removed
- Press iPad Home button to Close Poll Pad Application
- Power Poll Pad off
- Place Poll Pad along with Voter Card Encoder, stand, charging cord and AC plug into case
- Close Case and Seal; notate seal on paperwork

REDACTED VERSION

Exhibit C: Pro V&V Field Audit Report

Field Audit Report

Dominion Voting Systems
Democracy Suite (D-Suite) System
Version 5.5-A

Approved by: Jack Cobb

Jack Cobb, Laboratory Director

Approved by: Wendy Owens

Wendy Owens, VSTL Program Manager

December 2, 2020

1.0 INTRODUCTION

The purpose of this Report is to document the procedures that Pro V&V, Inc. followed to perform a Field Audit of the Dominion Democracy Suite (D-Suite) 5.5-AVoting System as fielded in selected counties in the State of Georgia.

1.1 References

The documents listed below were utilized in the development of this Report:

- Election Assistance Commission (EAC) 2005 Voluntary Voting System Guidelines (VVSG) Version 1.0, Volume I, "Voting System Performance Guidelines", and Volume II, "National Certification Testing Guidelines"
- Election Assistance Commission Testing and Certification Program Manual, Version 2.0
- Election Assistance Commission Voting System Test Laboratory Program Manual, Version 2.0
- National Voluntary Laboratory Accreditation Program NIST Handbook 150-2016, "NVLAP Procedures and General Requirements (NIST Handbook 150)", dated July 2016
- National Voluntary Laboratory Accreditation Program NIST Handbook 150-22, 2008 Edition, "Voting System Testing (NIST Handbook 150-22)", dated May 2008
- United States 107th Congress Help America Vote Act (HAVA) of 2002 (Public Law 107-252), dated October 2002
- Pro V&V, Inc. Quality Assurance Manual, Version 7.0
- EAC Requests for Interpretation (RFI) (listed on www.eac.gov)
- EAC Notices of Clarification (NOC) (listed on www.eac.gov)

1.2 Terms and Abbreviations

The terms and abbreviations applicable to the development of this Test Report are listed below:

"COTS" – Commercial Off-The-Shelf

"DRE" – Direct Record Electronic

"EAC" – United States Election Assistance Commission

"EMS" – Election Management System

"FCA" – Functional Configuration Audit

"HAVA" – Help America Vote Act

"ICC" – ImageCast Central

"ICX" – ImageCast X

"ICP" – ImageCast Precinct

"ISO" – International Organization for Standardization

"NOC" – Notice of Clarification

"QA" – Quality Assurance

"RFI" – Request for Interpretation

"VSTL" – Voting System Test Laboratory

"VVSG" – Voluntary Voting System Guidelines

1.3 Background

On Thursday, November 12, 2020, Pro V&V received a request from the Office of the Georgia Secretary of State to perform a Field Audit of the Dominion Democray Suite (D-Suite) 5.5-A Voting System in multiple counties, as selected by the Office, throughout the State. The purpose of this Field Audit was to verify the software/firmware and hardware used during the 2020 General Election was the same as the software/firmware and hardware that were Certified for Use by Georgia's Secretary of State Office.

1.4 System Description

The Democracy Suite 5.5-A Voting System is a paper-based optical voting system consisting of the following major components: the ImageCast Central (ICC) optical ballot scanner, the ImageCast Precinct (ICP) precinct count tabulator, and ImageCast X (ICX) BMD ballot marking device.

ImageCast Central (ICC) Count Scanner

The ICC is a high-speed, central ballot scan tabulator based on Commercial off the Shelf (COTS) hardware, coupled with the custom-made ballot processing application software. It is used for high speed scanning and counting of paper ballots.

ImageCast X (ICX) Ballot Marking Device (BMD)

The ICX consists exclusively of COTS available hardware and operating system, while the applications installed on top customize its behavior to turn it into a Ballot Marking Device (BMD). The ICX is designed to perform the following functions: ballot review and second chance voting, accessible voting, and ballot marking.

ImageCast Precinct (ICP)

The ICP device is a precinct optical scan paper ballot counter designed to provide six major functionalities: ballot scanning, second chance voting, ballot review, tabulation, and poll worker functions.

For ballot scanning functionality the ICP scans marked paper ballots, interprets voter marks on the paper ballots and stores the ballots for tabulation when the polls are closed.

Second Chance voting refers to scenarios in which an error has been detected on the voter's paper ballot (e.g., blank ballot, undervoted ballot, overvoted ballot, misread ballot, cross-over voted ballot), and the ICP notifies the voter by displaying a message or providing an audio visual cue, that one of these situations has been detected, and offers the voter an opportunity to reject and fix their ballot, or to cast the ballot as-is.

The Ballot Review feature allows a voter to review their vote selections using a visual representation, which displays to the voter a complete listing of all contests contained on the ballot and an indication of the results which will be recorded for each contest once the voter's ballot is cast.

The Tabulation of paper ballots cast by voters is performed when the polls are closed on the ICP unit and the unit tabulates the results, generates results files for aggregation into RTR, and prints a results report containing the results of the ballots cast.

For poll worker functions the ICP contains a small touch-screen LCD to allow the poll worker to initiate polling place activities, diagnostics and reports..

1.5 Scope

Pro V&V randomly selected components of the D-Suite system (an ICP, an ICX, and an ICC) from the system in each county that had been utilized in the November 2020 General Election. It was at the discretion of the Pro V&V on-site team which units were subject to verification. The Georgia Secretary of State Office contacted the selected counties and arranged for the Pro V&V team to be granted access to the systems. The selected counties were given less than six hours notice before the Pro V&V team arrived.

2.0 AUDIT OVERVIEW

The evaluation of the D-Suite 5.5-A Voting System consisted of removing a copy of the software/firmware from each component and evaluating the software/firmware against a known SHA-256 hash value outside of the system.

3.0 AUDIT PROCESS AND RESULTS

The following sections outline the audit process that was followed to evaluate the D-Suite 5.5-A Voting System under the scope defined in Section 1.5.

3.1 General Information

The Field Audit was conducted under the guidance of Pro V&V by personnel verified by Pro V&V to be qualified to perform the audit.

3.2 Audit Configuration

An ICX was selected at random from the warehouse in each county. The team member then photographed the seals and the device. All seals that needed to be removed were then removed. After all photographs were taken, the team member inserted a clean USB drive from Pro V&V into left hand access compartment. Next the team member then plugged in the unit and powered it on. At the prompt the team member inserted a Tech Key smart card and selected the option to "Extract Application". The team member then verified the SHA-256 generated by the unit and photographed the popup screen. The team then took the USB drive containing the exported application to a Pro V&V laptop to compare the SHA-256 hash values to the known value from previous testing.

An ICP was selected at random from the warehouse in each county. The team member then photographed the seals and the device. All seals that needed to be removed were then removed. After all photographs were taken, the team member removed any compact flash cards under county supervision and inserted two compact flash cards (one blank and the other containing techextract.enc that was created by Pro V&V during certification testing). The unit was plugged in and powered on. A password was entered and a tech iButton was then read by the ICP and the option to "Extract Firmware" was selected. The compact flash cards, if present, were returned to the same ICP. The team member then took the compact flash card containing the exported firmware to a Pro V&V laptop to compare the SHA-256 hash values to the known value from previous testing.

An ICC was selected at random in each county central office if there were multiple units. The team member then photographed the device. The county provided the creditals to login to the workstation. The Pro V&V team member navigated to the ICC folder on the root of the workstation and copied all application files onto a Pro V&V USB drive. The team member then took the USB drive containing the exported firmware to a Pro V&V laptop to compare the SHA-256 hash values to the known value from previous testing.

3.3 Summary Findings

During the Field Audit, a total of eighteen (18) components located among six (6) counties were evaluated to verify the version of software/firmware running on each device. It was discovered that allversions on all components matched the known certified SHA-256 hash value.

4.0 CONCLUSIONS

Based on the results obtained during the Field Audit, Pro V&V determines the D-Suite 5.5-A Voting System, on all evaluated components, is the voting system certified by the Georgia Secretary of State Office.