A Process for Assessing Voting System Risk Using Threat Trees

Alec Yasinsac
yasinsac@gmail.com

Harold Pardue
hpardue@usouthal.edu

School of Computer and Information Sciences
University of South Alabama
Mobile, Al 36688-0002, USA

Abstract

Security continues to be a critical issue in the safe operation of electronic voting machines. Risk assessment is the process of determining if a particular voting system is at risk and what steps can be taken to mitigate the risk. We propose an iterative risk assessment process using threat trees. This process involves using a voting system risk taxonomy to categorize a threat, a schema to express logical hypothesis about a threat, generating a threat tree through functional decomposition, expressing threat instance semantics as nodal properties with metrics, validating the threat instance through independent representations, and finally pruning the tree for enhanced usability and understandability. This process provides guidance to an analyst in using threat trees to conduct risk assessment of electronic voting systems. Because this process is based on abstract and extendable structures, it facilitates the comparison and validation of independent risk evaluations. Prospective voting system risk assessment metrics are provided.

Keywords: electronic voting systems, risk assessment, threat trees, taxonomy

1. INTRODUCTION

In their 2004 seminal work Kohono, Stubblefield, Rubin and Wallach (2004) et al. closed the book on the question of whether security mechanisms were critical to safe operation of electronic voting machines. Their analysis showed that there were many critical vulnerabilities in a widely used voting system. That work also precipitated a firestorm of vulnerability analyses that further confirmed that existing electronic voting system security mechanisms were insufficient to ensure election integrity.

This paper represents a first step in providing guidance to analysts for systematically determining if particular voting systems are at risk and to identify steps that can mitigate that risk. There is significant work documented in the literature regarding fault analysis (Clifton, 1999) and threat tree analysis (Schneier, 1999; Uppal, 2007; Evans, Heinbuch, Kyle, & Porokowski, 2004), but our work details a specific approach for specifying voting system threats that can facilitate risk analysis.

As information systems go, voting applications are relatively simple. Their core function is to capture the will of the eligible voters. There are no complex algorithms; addition is simple arithmetic and the numbers are relatively small, as computer computations go.

On the other hand, voting systems have been under attack for centuries, with malicious par-

ties trying to influence, or control electoral outcomes. An important challenge to conducting effective elections is to protect against these manipulative threats.

In this paper, we introduce a process for identifying, categorizing, specifying, validating, and pruning voting system threats. At the core of this process is the threat tree.

A threat tree is a data structure for representing the steps that an attacker would take to exploit a vulnerability in order to accomplish malicious intent. While there has recently been much discussion of voting system threats and numerous voting system security vulnerability assessments, (Black Box Voting, 2005); Yasinsac, Wagner, Bishop, Baker, Medeiros, Tyson, Shamos, & Burmester, 2007; Gardner, Yasinsac, Bishop, Kohno, Hartley, Kerski, Gainey, Walega, Hollander, & Gerke, 2007; California Secretary of State, 2007; Epstein, 2007; & Alaska, 2008) we are unaware of any systematic or formal effort to catalog, specify, and validate voting system threat trees.

Threat trees allow the analyst to (1) Descriptively name nodes as threat goals and steps (2) Graphically express logical relationships between nodes and (3) Define attack goal and step semantic properties as nodal attributes. Collectively these three characteristics allow the abstraction and precision that are necessary to reason comparatively about fundamentally different threats.

The remainder of this paper provides a detailed description and discussion of the risk assessment process followed by a brief summary.

2. VOTING SYSTEM RISK ASSESSMENT PROCESS

The purpose of the voting system risk assessment process is to provide guidance to an analyst in using threat trees to conduct risk analysis of voting systems. The power of this process derives from the use abstraction to produce artifacts that categorize and illuminate important voting system security issues while facilitating a balance between detail and complexity. These artifacts, because they are based on generalizations that are flexible and extensible yet explicit in their construction, enable an analyst to compare and validate independent evaluations of risk. In other words, these generalizations provide a common structure upon which to express individual perceptions, metrics, and analyses.

The threat tree generation process consists of six iterative steps (see Figure 1). The first step is to identify the threat as a high level attack goal. In the second step, the analyst rigorously defines the high level goal by assigning relevant parameters from the voting system attack taxonomy, creating new taxonomy parameters where necessary. This level of detail provides the foundation for the refinement step that follows.

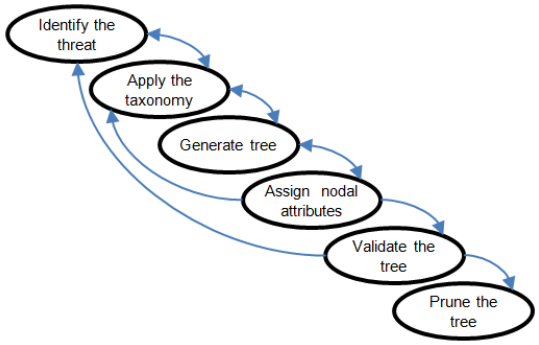

Figure 1. Risk Assessment Process.

In the fundamental step of the process, threat tree generation, the analyst conducts functional decomposition, recursively expanding each node into its requisite tasks. The recursive functional decomposition continues until the threat is refined sufficiently to conduct the necessary analysis. The result of this step is a threat tree.

With the threat tree defined, each node is assigned attributes that capture properties that are relevant to the analyst. These attributes may be metrics, data points that allow analysts to compute metrics, or simply observations that provide the analyst a point of reference for their analytical processes. They differ from the taxonomy parameters in that while taxonomy parameters are generic threat properties that allow threat categorization, these attributes are specific to the analyst's risk assessment goals.

In the fifth step, the analyst iterates the first four steps to validate and enhance the threat tree. Each of the first four steps increases specificity, adding detail to the threat processes and properties.

In the final step, the analyst prunes the threat tree through abstraction leaving a threat tree

that is well understood and whose threat instances can be comparatively analyzed.

The remainder of this section contains a detailed description of each step in using the voting system risk assessment process.

2.1. IDENTIFY THE THREAT

The first step is to identify the high level threat. The analyst may derive high level threats through literature searches, brainstorming, personal experience, newspaper articles, etc. To be most useful, the identified threat's impact must be tangible and measurable. For example, the threat: "Remove a ballot from a ballot box" is concrete while "Change an election result" is inherently ambiguous.

2.2. APPLY THE TAXONOMY

The second step of the process requires the analyst to define the high-level threat in abstract yet precise terms. In order for these definitions to be useful in making independent comparisons and analysis, threats must be categorized according to a common structure. We offer a voting system threat taxonomy for this purpose. Our extensible voting system risk taxonomy can capture important properties of voting system vulnerability and those that may seek to create corresponding exploits. This taxonomy employs a hierarchical structure based on attribute n-tuples , where the lower levels comprehensively describe the properties of the parent.

2.2.1. TAXONOMY CLASSIFICATION

Taxonomy fundamentally classifies the target group. That is, it provides commonality among group members in a way that can facilitate understanding and application. For example, our proposed taxonomy provides a mechanism for analysts to more precisely capture the threats that they are expected to analyze. This abstraction may be realized by searching, for example, against attribute wild cards, i.e. all attacks that accomplish wholesale impact, or all attacks that involve rogue poll workers.

These abstractions may allow elections officials to devise procedures that can systematically mitigate the defined threats. For example, preventing voters from accessing removable media eliminates the class of attacks that pairs the following:

<Role(Voter), AttackVector(RemovableMedia)>

Similarly, if the voting system does not include commercial off the shelf software, then all attacks associated with the attribute <Software(COTS)> are eliminated.

Finally, the taxonomy can allow the analyst to identify and syntactically prohibit conflicting attributes. For example, it may not be possible to conduct a DoS attack after the voting period ends. We term these "constraints" in the taxonomy and represent them as predicate pairs, e.g.:

<Objective(DoS), Phase(AfterVotingPeriod)>

One challenge of modeling any process or issue is to decide what level of detail is optimum. Excessive detail can unnecessarily complicate the model, while too little detail can limit its usefulness. Our voting system threat taxonomy's present form is easily extensible. As threat attributes emerge, they may be added to the tree depth or items of less interest may be removed. Moreover, the model can be automated to prompt manual entry guided by the taxonomy's syntax.

The content of the threat taxonomy is based on an extensive review of the extant literature and the experience and expertise of the authors. The taxonomy was constructed in a top-down process where each logical structure block was decomposed into non-overlapping sub-block structures.

We provide our voting system threat taxonomy as Appendix A.

2.2.2. SCHEMA

The voting system risk taxonomy enables the analyst to consistently classify threats through a common syntax. However, the usefulness of the resulting artifacts will be limited if 1) the analyst does not have a means of consistently expressing the logical hypothesis engendered by the definition of an attack and 2) a consistent means of expressing terms contained in those hypothesis. A schema serves both needs.

We generate voting system threat tree definitions and schema by creating logical hypothesis regarding prospective voting system attacks and we capture that hypothesis as n-tuple expressions. For example, we posit, as definition, that the only two overarching voting system attack goals are to either alter or ensure a contest result or to negatively impact voter confidence. We capture that hypothesis as follows:

VSAttack = <AlterContestDecision, UndermineVoterConfidence>

We similarly posit that there are only four ways that an attacker can alter a contest decision, given as:

AlterContestDecision = <AddVotes, DeleteVotes, FlipVotes, AlterCount>

Further, votes are either physical or electronic, so:

DeleteVotes= <DeleteAcceptedBallotsPhysical, DeleteAcceptedBallotsElectronic>

Finally, we propose the following hypothesis regarding any attacker's ability to delete an accepted physical ballot, stated as a schema:

schema: DeleteAcceptedBallotsPhysical.[Phase].[Control] = <GainPrivateAccessToABPs.

RemoveABPsFromControlledCustody,

MoveABPsToPrivateSpace>

This schema stands as a template or skeleton for any voting system attack that involves deleting physical ballots.

The definitions and schema above reveal the pseudo-formal language approach that we adopt. Our conventions include:

- Use short phrases coupled as long words, with the first letter of each word in caps
- Only abbreviate well known terms or phrases
- Establish a data dictionary of node names

We provide an extended set of definitions and schema as Appendix B.

2.3. GENERATE THREAT TREE

Step three involves the recursive functional decomposition of a threat into a collection of goals and steps necessary to carry out a threat. The recursive functional decomposition continues until the threat is refined sufficiently to conduct the necessary analysis. The result of this step is a threat tree.

2.3.1. THREAT TREES

For our purposes, a threat defines the process that one or more attackers might take to accomplish a malicious act in an election. The "tree" is a powerful abstraction that graphically captures relationships among nodes that are hierarchically connected by directional edges, while allowing analysts to express individual node properties as nodal attributes. The tree structure allows a systematic approach to threat analysis, including facilitating abstraction and decomposition and allows analysts to categorize goals and steps so they can focus on those that are most critical.

For threat trees to be most useful, node names must capture the node's core function, whether the node is a goal or a step. Short, succinct names allow the analyst to recognize the collective meaning of the tree based on node type, name, and connectivity.

2.3.2. THREAT TREE COMPONENTS

In order to leverage tree structures to represent threat processes, we define voting system threat trees so that their graphical properties capture important process relationship properties. We accomplish this by establishing the three node types of AND, OR, and TERMINAL . Subordination reflects specification through functional decomposition, so nodes higher in the tree are abstractions of subordinate nodes. All nodes that are immediately subordinate to an AND node must be carried out in order to meet higher level goals, while OR node subordinates reflect alternate means to accomplish an intended function. TERMINAL nodes have no subordinates, thus reflect the primitive operations (i.e. steps) that accomplish the modeled threat, while AND and OR nodes reflect intermediate attack goals. Figure 2 illustrates a generic threat tree composed of AND [A, D], OR [B, I], and TERMINAL [C, E, F, G, H, J, K] nodes.

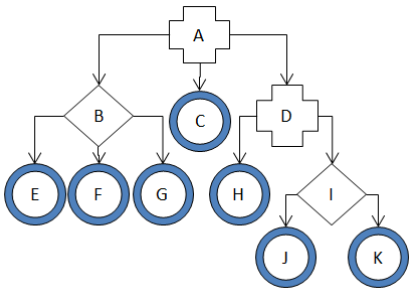

Figure 2. Generic Voting System Threat Tree.

A tree represents many threat instances, or attacks, as a combination of TERMINAL nodes that satisfy the logical requirements of the tree. For example, in order to realize threat A, an attacker would have to carry out goals B, C

and D. Accomplishing E, F, or G would accomplish B, while H and J or K would be needed to accomplish D. Thus, <E, C, H, K> is one attack represented in Figure 1, as is <G, C, H, K>. There are four other TERMINAL node (step) combinations (threat instances) that realize threat A.

We can identify several properties of the threat instances captured in this tree without knowing any of the nodes' semantic properties. We know for example that:

- The tree depth is four and its breadth is seven
- This tree represents exactly six distinct threat instances
- Each threat instance requires four steps (i.e. four TERMINAL nodes)
- Nodes C and H are necessarily steps in every threat instance

These are computations that can be applied to all tree structures and all other routine tree algorithms and provability properties similarly apply to these trees. Thus, we know that splitting a TERMINAL node into an OR node doubles the number of represented distinct attack instances. If the split is an AND node, it adds one step to each attack instance that includes the replaced node. The practical importance of these properties and computations will be evidenced in the validation of threat tree metrics.

We also know that canonical limitations that apply to tree structures also apply to our voting system threat tree, most importantly that their size expands rapidly relative to their breadth and depth. In our approach, tree depth is controlled by the level of detail necessary to describe the goal or activity represented in the node. These decisions are made by the analyst. For example, if a particular threat may involve the task of "Picking a lock", one analyst may encode that task as a TERMINAL node, while another may encode it as an AND node with the subordinate TERMINAL nodes of "Acquire necessary skill and knowledge" AND "Attain Necessary Access" AND "Acquire necessary tools" AND "Pick the lock". The latter approach adds one level of depth to its branch.

Note that we intentionally avoid temporal notions of step or goal sequencing in the tree's graphical representation. If sequencing is important to a specific analysis, temporal dependencies may be expressed as nodal properties.

2.4. ASSIGN NODAL PROPERTIES

At this stage in the process, the focus shifts from the syntax of generic threat categorization to the semantics of the primitive operations (steps) of a threat in the context of a specific risk assessment. The analyst must define a threat instance for an attack (a realization of a threat) and assign attributes specific to the threat instance. The two attributes required by our process are likelihood and impact. Likelihood is the probability that an attack will be realized and impact measures the consequences of an attack. Both likelihood and impact are expressed and measured as quantifiable metrics.

2.4.1. THREAT INSTANCE

The unit of evaluation for voting system threat trees is a threat instance, or equivalently, an attack, thus an attack is the realization of a threat. We choose to focus on primitive operations (steps) because steps can be associated with a metric. For example, an analyst can estimate how much or how little of some resource is required to carry out a given set of steps. A goal represents an attacker's purpose or objective. As such, it is more difficult to assign quantifiable metrics to a purpose or objective than it is to a concrete activity or sequence of steps.

Metrics are important because they allow the analyst to compare and validate independent evaluations. This allows the analyst to reason comparatively about fundamentally different threats to voting systems. However, it is not always possible or feasible to provide direct evaluations of all possible sets of primitive operations or steps in a threat tree because of the potential for state space explosion.

We use goal nodes to abstract multiple sets of steps into a single logical unit of evaluation and thus mitigate this problem. Abstraction can reduce tree depth and make evaluation tractable. For example, in Figure 2, if we understood the properties of node I sufficiently to collapse it into a TERMINAL node, thus eliminating nodes J and K, it would reduce the number of threat instances by half (from six to three). Thus, it may make sense to decompose goals in order to reason about them, but where that understanding is sufficiently detailed, to

evaluate the tree at a higher abstraction level to reduce the evaluation state space.

2.4.2. THREAT INSTANCE METRICS

Threat tree nodes may have many, sometimes seemingly contradictory, properties that dictate or influence a goal or step's occurrence LIKELIHOOD or its potential IMPACT. These are, of course, the two parameters for assessing voting system risk. Voting systems in the United States are highly complex. Consequently, risk LIKELIHOOD and IMPACT are varied and difficult to capture and express. It is not uncommon for two highly qualified election experts to disagree vehemently regarding the voting system risk.

We highlight some voting system threat node attributes that capture a perspective of each of these properties in this section.

2.4.2.1. LIKELIHOOD METRICS

We may measure LIKELIHOOD and IMPACT as a continuous variable on a 0 to 1 scale. For the former, 0 (as the lower LIKELIHOOD extreme) would indicate that the event will not (or cannot) occur, while 1 (at the upper extreme) means that the event is certain to occur. For the latter, 0 would reflect no impact while a catastrophic result would represent the opposite extreme impact. Alternatively, a simple three step discrete metric of high, medium, and low could also represent LIKELIHOOD and/or IMPACT.

The only absolute in estimating risk likelihood is that there are no absolutes. Issues of relativity, temporality, uncertainty, and other qualifications render even the most intuitively accurate assumptions invalid, or worse yet, counterproductive. The best that we can hope for is to leverage heuristics to find metrics that incorporate best practice experience and offer analysts a chance at estimating comparative risk. We offer a few such prospective voting system risk assessment metrics below.

Cost. The resource commitment required to carry out a voting system attack always bounds the prospective attacker's options. Money, labor, time, and equipment are canonical resources that are represented in a cost metric.

Necessary expertise. We may expect that a requirement for specialized knowledge or skill diminishes the likelihood of an attack occurring. The obvious likelihood limitation is that specialized expertise injects is to reduce the pool of potential attackers or increases the time and resources that an attacker needs to carry out the attack. It also likely indicates that there is an advanced sophistication, and a resulting elevated complexity, in the prospective attack.

Detectability. Detection can enable prevention of many types of voting system attacks. It can also allow officials to punish perpetrators after the fact and can allow correction of damage caused by a voting system attack.

We use the term "detectability" to capture the notion of how difficult or likely it is that an attack will be detected. We posit generally that attacks, events, and actions that are more likely to be detected are less likely to be attempted and that they are less likely to achieve maximum impact than those that are more difficult to detect.

2.4.2.2. IMPACT METRICS

Generically, we think of threat IMPACT as the *magnitude or degree of damage* that will, or is expected to, occur as a result of a realized threat. In practice, IMPACT is context exclusive to the extent that the same voting system threat may have a catastrophic impact in one environment, but be essentially benign in a different environment. Assignment of the IMPACT metric is a major and important task of the analyst and requires significant subject matter expertise.

The two primary overarching goals of voting system attacks are either to impact election integrity or to influence public's perception about the election. Thus, we partition IMPACT metrics according to these two aspects and address IMPACT as the magnitude of the effect on voting system integrity or public perception.

2.4.2.3. INTEGRITY IMPACT METRICS

Voting system integrity attacks are what we think of when we discuss election fraud, that is, integrity attacks maliciously influence a contest result in an election. This encompasses canonical election fraud issues, such as ballot stuffing.

Voting system integrity attack impact ranges from deleting one legal vote (or equivalently, injecting one illegal vote) with no impact on any contest selection, to controlling the selected candidate or issue decision in all contests. Voting system integrity issues are either

related to vote counting (process where each voter selection is added to the total, one by one) or aggregation (where subtotals are combined to reflect the cumulative result). The following metrics are illustrative (as opposed to comprehensive) and represent issues that are relevant to risk assessment.

Without knowing a contest result a priori, an attack waged during the voting period has the best chance to be decisive if it can effect a large volume of votes . Such attacks are similar in many ways to wholesale purchasing tactics and the term "wholesale vote fraud" has become part of the election integrity vernacular. Wholesale attacks optimize effort-to-effect ratio, or more mathematically, retail attacks are linear in terms of the effort-to-effect ratio, while wholesale attacks are geometric (or exponential) in effort-to-effect ratio.

Knowing the magnitude of change necessary to control an electoral decision can be important to an attacker, allowing a small number of votes to be decisive. We have recently seen two federal elections (Minnesota Senate 2008 election and New York's 2009 special election for their 20th Congressional district) decided by only a few hundred votes. Each of these contests was vulnerable to post voting period attacks where a relatively small malicious change could be decisive.

2.4.2.4. PUBLIC PERCEPTION IMPACT METRICS

For a malicious party that desires to negatively influence election-related public perception, the prospective damage ranges from generating isolated incidents of misunderstanding to wrongfully creating widespread belief that one or more electoral decisions were influenced by error or malice. While election integrity attacks against voting systems predominantly involve data and processes that are integral to conducting an election, perception issues are uniformly driven through mass information dissemination media that is separate from the voting system. The voting system responsibility in this process is to be able to provide strong, accurate information about election activity. Thus, attacks on public perception are either voting system independent, or involve modifying data reported to public dissemination media, as reflected in the following illustrative metrics.

Elections officials uniformly rely on validation mechanisms both to ensure election integrity and to reassure the public of election accuracy. Virtually all validation mechanisms employ some type of redundancy, so attackers may attack either the primary electoral product or the validation data in order to create a negative perception (Yasinsac & Bishop, 2008). For example, ballot accounting procedures measure the number of ballots issued against the counted. A public perception attack may target the records of the number of ballots issued so that validation will suggest that there were more voters than ballots. The greater the disparity, the greater the potential to create negative public perception.

2.4.3. THREAT INSTANCE STOPPING FUNCTION

A challenge to any system based on functional decomposition is how to fashion a stopping function. That is, it can be difficult to identify the best or most effective abstraction level to ensure that the decomposition process does not reach a point of diminishing returns.

In our case, decomposition stops when the analyst can assign values to the nodal attributes with sufficient precision to accomplish the necessary global computations. For example, if our metric is cost, the analyst must decompose the task to the level that the cost of each step is clear and justifiably assigned. Justification may be based on the skill of the analyst or upon some predefined threshold, but the degree of precision is always dictated by the metric's context.

Cumulative analysis must then begin at the TERMINAL nodes that comprise each threat instance, which is our unit of evaluation. To illustrate, we compute the cost (C) of instance (i) of threat (a) as (C(a, i)), which is the sum of the costs of the steps required to carry out threat instance (a, i). For example. if <E, C, H, K> is instance 1 of threat A, as shown in Figure 1 on page 5 above, we compute:

$$C(A,1) = C(E) + C(C) + C(H) + C(K)$$

Thus, the fundamental voting system threat tree unit of evaluation is horizontal. That is, metrics are assigned at the TERMINAL nodes and those values are accumulated by threat instance, which reflects the tree's greatest specificity level and the level where the metric is assigned.

2.5. VALIDATE THREAT TREE

Since there are no well known metrics, metric validation is essential to the voting system risk assessment process. One way to approach validation is through comparing independent representations. With voting system threat trees, if metrics have suitable computational properties, we can use redundancy by comparing expert assessment against computed values.

To accomplish this validation, an analyst would employ a five stage analysis.

1. Select a metric that that can be assigned based on expert opinion
2. Create an algorithm for computing a parent node's metric based on the child metric values8.
3. Apply expert metric evaluation rules to every node in the tree
4. Compute the metric value for each goal node and
5. For non-terminal nodes, compare the value assigned in Step 3 to the value that is vertically computed from its subordinate nodes in Step 4.

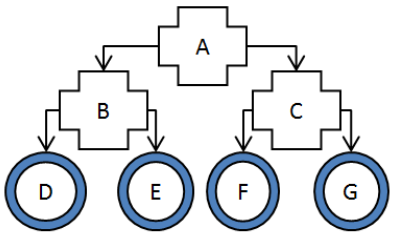

Figure 3. Simple, Generic Threat Tree.

To illustrate, consider the simple [hypothetical] threat tree in Figure 3 with the nodes:

A: Threaten voting equipment

B: Create malware

C: Install the malware

D: Design attack

E: Gain necessary knowledge

F: Determine sleepover location

G: Gain access to sleepover location at an appropriate time.

We now conduct the five stage analysis:

1. Select cost metric C
2. Compute the cost of a parent as the sum of the cost of the children
3. For instructional purposes, assume that the analyst opinion review assigns the cost of each node to be:

   (1) C(A) = 75, C(B) = 10, C(C) = 100, C(D) = 5, C(E) = 5, C(F) = 50, C(G) = 100
4. We compute the cost of the non-terminal nodes is:

   (2) C(A) = 160, C(B) = 10, C(C) = 150
5. Comparison of evaluations (3) and (4) reveals an inconsistency between the expert analysis and computed analysis at the highest level, which would not be surprising. It also reveals an inconsistency between the expert evaluation at the intermediate level for node C, suggesting reanalysis of assigned values for nodes F and G, or consideration of re-examining node C's decomposition.

2.6. PRUNE THREAT TREE

The goal of pruning the threat tree is to strike a balance between abstraction and detail. The tree must have sufficient detail to be useful and understandable by the analyst. However, too much detail creates a model that is unnecessarily complex. Complexity creates excessive cognitive load for the analyst (reducing understandability) while potentially make quantitative analysis of the tree's metrics intractable (reducing usefulness).

For example, in the simplified threat tree depicted in Figure 2, assume that step E (Gain necessary knowledge) was originally decomposed into two additional OR steps: "H: Interview insider" OR "I: Review software components". Perhaps the analyst constructing the threat tree, after validating the tree's metrics, determined that considering whether the attacker interviewed a vendor employee OR obtained a copy of a software component for private review was extraneous to understanding the likelihood and impact of the attack. Therefore, to reduce the complexity of the tree, make the tree more understandable and usable, these two steps were pruned from the threat tree.

3. SUMMARY

In this paper, we propose a voting system risk assessment process that leverages three characteristics of threat trees: the ability to (1) Descriptively name nodes as threat goals and steps (2) Graphically express logical relationships between nodes and (3) Define attack goal and step semantic properties as nodal attributes. Collectively these three characteristics allow the abstraction and precision that are necessary to reason comparatively about fundamentally different threats.

The provision of a voting system risk taxonomy and schema facilitates the comparison and validation of independent risk evaluations. That is, because the taxonomy provides a common syntax for categorizing threats and the schema provides a means of expressing logical hypothesis in consistent terms, the risk assessment of independent analysts can be compared in a logical and quantifiable manner. Further, because this process is based on abstract, extendable and common structures, it can be effective for facilitating group risk assessment. Rather than comparing independent risk evaluations after the fact, analysts can work collectively through each phase of the process.

Future research should include a vetting or validation of the schema and taxonomy by voting systems domain experts.

4. ACKNOWLEDGEMENT

This work was supported in part by the Election Assistance Commission under grant EAC-RDV08-R-001.

5. REFERENCES

Alaska, (2008) Election Security Project, *Division of Elections*, January 18, 2008. Retrieved on June 2010 from http://www.elections.alaska.gov/election_security.php.

Black Box Voting, (2005) The Black Box Report, SECURITY ALERT: Critical Security Issues with Diebold Optical Scan Design. Retrieved June 2010 from http://www.blackboxvoting.org/BBVreport.pdf.

Brennan Center (2006) The Machinery of Democracy: Protecting Elections in an Electronic World, *Brennan Center Task Force on Voting System Security*, Lawrence Norden, Chair.

California Secretary of State, (2007) UC Final Reports for the Top-to-Bottom Review (July-Aug. 2007). Retrieved on June 2010 from http://www.sos.ca.gov/elections/elections_vsr.htm.

Clifton, E. (1999) Fault Tree Analysis - A History. *Proceedings of the 17th International Systems Safety Conference*.

Epstein, J. (2007) Improving Kentucky's Electronic Voting System Certifications. *Letter to Kentucky Attorney General*, September 27, 2007. Retrieved on June 2010 from http://ag.ky.gov/NR/rdonlyres/1B3F7428-0728-4E83-AADB-51343C13FA29/0/votingexpertletter.pdf.

Evans, S., Heinbuch, D., Kyle, E., & Prokowski, J. (2004) Risk-based Systems Security Engineering: Stopping attacks with intention, *IEEE Security & Privacy* 2(6) 59-62.

Gardner, R., Yasinsac, A., Bishop, M., Kohno, T., Hartley, Z., Kerski, J., Gainey, D., Walega, R., Hollander, E., & Gerke, M. (2007) Software Review and Security Analysis of the Diebold Voting Machine Software", Final Report For the Florida Department of State, July 27, 2007. Retrieved on June 2010 from http://election.dos.state.fl.us/pdf/SAITreport.pdf.

Kohno, T., Stubblefield, A., Rubin, A., & Wallach, D. (2004) Analysis of an Electronic Voting System. *IEEE Symposium on Security and Privacy*, May 9-12, 27-40.

Schneier, B. (1999) Attack Trees. *Dr. Dobb's Journal*, December, 24(12).

Uppal, V. (2007) The Importance of Threat Modeling, *IRM Research White Paper*. Retrieved June 2010 from http://www.irmplc.com/downloads/whitepapers/Threat_Modelling.pdf

Yasinsac, A, Wagner, D., Bishop, M., Baker, T., Medeiros, D., Tyson, G., Shamos, M., & Burmester, M. (2007) Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware, Final Report, Security and Assurance in Information Technology Laboratory, Florida State University, February 23. Retrieved on June 2010 from http://election.dos.state.fl.us/pdf/FinalAudRepSAIT.pdf.

Yasinsac A. & Bishop, M. (2008) The Dynamics of Counting and Recounting Votes, *IEEE Security and Privacy Magazine*, 6(3) 22-29.

Appendix A. VOTING SYSTEM THREAT TAXONOMY

VSRisk = <Attack, Impact, Likelihood>
- Impact = <Magnitude**,** ContestBreadth, NumberOfContests**,** Persistence>
  - Magnitude = <**Retail, Wholesale**, CloseRace>
  - ContestBreadth = <**Federal, State, Local**>
  - NumberOfContests = <**SingleContest, MultipleArbitraryContests, MultipleContestsOfGivenType**>
  - Persistence = <**SingleElection, MultipleCycles, Perpetual**>
- Likelihood = <**Low, VeryLow, UnMeasurable, UnImaginable**>
- Attack = <VS, Command, VSRiskTo, Environment, Protocol, MaliciousIntruder+>
  - VS = <PCOS, CCOS, VBM, VBP, DRE, PBHC, IV, BMD>
  - Command = <Adjustable, Precision>
    - Adjustable = <**ChangeOnDemand, LimitedChange, FireAndForget**>
    - Precision = <**Candidate, Contest, Party**>
  - VSRiskTo = <ElectionAccuracy, VoteAttribution, VoterConfidence>
    - ElectionAccuracy = <**VoteError, AccumulationError**>
    - VoteAttribution = <**VoteBuying, VoteSelling, VoterCoersion**>
  - Environment = <Vulnerability, Phase>
    - Vulnerability = <Software**, Hardware**>
      - Software = <**VendorFirmware, COTS,** ElectionDefinition>
      - ElectionDefinition = <**BallotDef, ConfigItems**>
    - Phase = <**BeforePollsOpen, DuringVoting, AfterPollsClose**>
  - Protocol = <Objective+, AttackVector+, **Tree**>
    - Objective = <ChangeCount, **DoS,** VoteAttribution**,** DiscreditCount>
      - ChangeCount = <**BallotStuffing, BallotDeletion, VoteFlipping**>
      - VoteAttributionPurpose = <**VoteBuying, VoteSelling, VoterCoersion, GeneralIrritation**>
      - DiscreditCount = <**CountAuditMismatch, PublicAnomaly**>
    - AttackVector = <**VoterInput, SupervisorEntryDevice, RemovableMedia, Network, VendorKey**>
  - MaliciousIntruder = <Role, Skills, **Resources**>
    - Role = <**Voter, PollWorker, Auditor,** ElectionsOfficial, **OfficeAdmin>**
    - ElectionsOfficial = <Permanent, Temp>
    - Permanent = <**County, State, Vendor**>
    - Temp = <**CountyOffice, Precinct**>
    - Skills = <**HighTech, TechFamiliar, SpecificSkills, TechNovice**>

Appendix B. VOTING SYSTEM THREAT TREE SCHEMA

VSAttack = <AlterContestDecision, UndermineVoterConfidence>
AlterContestDecision = <AddVotes, DeleteVotes, FlipVotes, AlterCount>
UndermineVoterConfidence = <AlterAuditData, AlterContestTotals, DenialOfService, CreateOperationalProblems>
DeleteVotes = <DeleteAcceptedBallotsPhysical, DeleteAcceptedBallotsElectronic>
AddVotes = <StuffPhysicalBallotBox, CreateBallotImages>
schema: DeleteAcceptedBallotsPhysical.[Phase].[Control] =
GainPrivateAccessToABPs
RemoveABPsFromControlledCustody
MoveABPsToPrivateSpace
DeleteAcceptedBallotsPhysical.[Phase:AVP].[Control:none] =
GainPrivateAccessToABPs
PollWorkerAutomatic or ElectionsOfficialAutomatic or TriggerPollingPlaceFireAlarm
RemoveABPsFromControlledCustody
StealBallotBox or RemoveBallotsFromBox
ConcealContraband
MoveABPsToPrivateSpace
DeleteAcceptedBallotsPhysical.[Phase:AVP].[Control:AcceptedBallotCoC] =
GainPrivateAccessToABPs,
PollWorkerAutomatic or ElectionsOfficialAutomatic or TriggerPollingPlaceFireAlarm,
RemoveABPsFromControlledCustody(Constraint(RiskCoCDetection)),
MoveABPToPrivateSpace
Schema: DeleteAcceptedBallotsElectronic.[Phase].[Control].[HackVector]
Phase = <BVP, DVP, AVP, DR>
HackVector = <Malware, SupervisorMode, BadData, NetHack, RemovableMediaHack>
Control = <CommonControl, EControl, PControl>
CommonControl = <RandomAudit, PollWatchers, TwoPersonIntergrity>
EControl = <L&STest, EquipCoC, ParallelTesting, HashCodeTest>
PControl = <VotableBallotCoC, AcceptedBallotCoC, BallotAccounting, BallotWatermarking>
DeleteAcceptedBallotsElectronic.[Phase:Any].[Control:none].[HackVector:Malware] =
CreateMalware, InstallMalware
DeleteAcceptedBallotsElectronic.[Phase:DVP].[Control:none].[HackVector:Malware] =
CreateMalware(BVP, DVP), InstallMalware(BVP, DVP)
DeleteAcceptedBallotsElectronic.[Phase:DVP].[Control:L&ATest].[HackVector:Malware] =
CreateMalware, InstallMalware(Constraint(DefeatL&A or InstallAfterL&A))