Lecture 25, June 1, 2026

ECS 235B, Foundations of Computer and Information Security

Spring Quarter 2026

Example: Visual Audit Browser

- Frame Visualizer
  - Generates graphical representation of logs
- Movie Maker
  - Generates sequence of graphs, each event creating a new graph suitably modified
- Hypertext Generator
  - Produces page per user, page per modified file, summary and index pages
- Focused Audit Browser
  - Enter node name, displays node, incident edges, and nodes at end of edges

Example Use

- File changed
  - Use focused audit browser
    - Changed file is initial focus
    - Edges show which processes have altered file
  - Focus on suspicious process
    - Iterate through nodes until method used to gain access to system determined
- Question: is masquerade occurring?
  - Auditor knows audit UID of attacker

Tracking Attacker

- Use hypertext generator to get all audit records with that UID
  - Now examine them for irregular activity
  - Frame visualizer may help here
  - Once found, work forward to reconstruct activity
- For non-technical people, use movie maker to show what happened
  - Helpful for law enforcement authorities especially!

Example: MieLog

- Computes counts of single words, word pairs
  - Auditor defines “threshold count”
  - MieLog colors data with counts higher than threshold
- Display uses graphics and text together
  - Tag appearance frequency area: colored based on frequency (*e.g.*, red is rare)
  - Time information area: bar graph showing number of log entries in that period of time; click to get entries
  - Outline of message area: outline of log messages, colored to match tag appearance frequency area
  - Message in text area: displays log entry under study

Example Use

- Auditor notices unexpected gap in time information area
  - No log entries during that time!?!?
- Auditor focuses on log entries before, after gap
  - Wants to know why logging turned off, then turned back on
- Color of words in entries helps auditor find similar entries elsewhere and reconstruct patterns

Key Points

- Logging is collection and recording; audit is analysis
- Need to have clear goals when designing an audit system
- Auditing should be designed into system, not patched into system after it is implemented
- Browsing through logs helps auditors determine completeness of audit (and effectiveness of audit mechanisms!)

Review: What Is "Secure"?

- Adding a generic right *r* where there was not one is "leaking"
  - In what follows, a right leaks if it was not present *initially*
  - Alternately: not present *in the previous state* (not discussed here)
- If a system *S*, beginning in initial state s_0, cannot leak right *r*, it is *safe with respect to the right r*
  - Otherwise it is called *unsafe with respect to the right r*

Safety Question

- Is there an algorithm for determining whether a protection system *S* with initial state s_0 is safe with respect to a generic right *r*?
  - Here, “safe” = “secure” for an abstract model

Mono-Operational Commands

- Answer: *yes*
- Sketch of proof:

  Consider minimal sequence of commands c_1, …, c_k to leak the right.

  - Can omit **delete**, **destroy** (with some rewriting)
  - Can merge all **create**s into one

  Worst case: insert every right into every entry; with s subjects and o objects initially, and n rights, upper bound is $k \leq n(s+1)(o+1)+1$

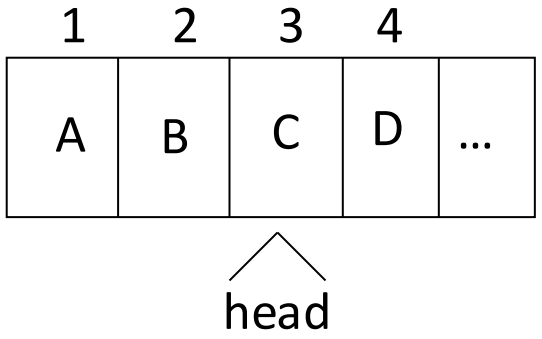
General Case

- Answer: *no*
- Sketch of proof:

  Reduce halting problem to safety problem

  Turing Machine review:
  - Infinite tape in one direction
  - States *K*, symbols *M*; distinguished blank *b*
  - Transition function $\delta(k, m) = (k', m', \text{L})$ means in state *k*, symbol *m* on tape location replaced by symbol *m′*, head moves to left one square, and enters state *k′*
  - Halting state is q_f; TM halts when it enters this state

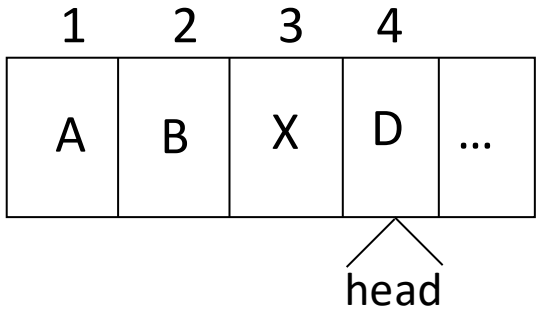
Mapping

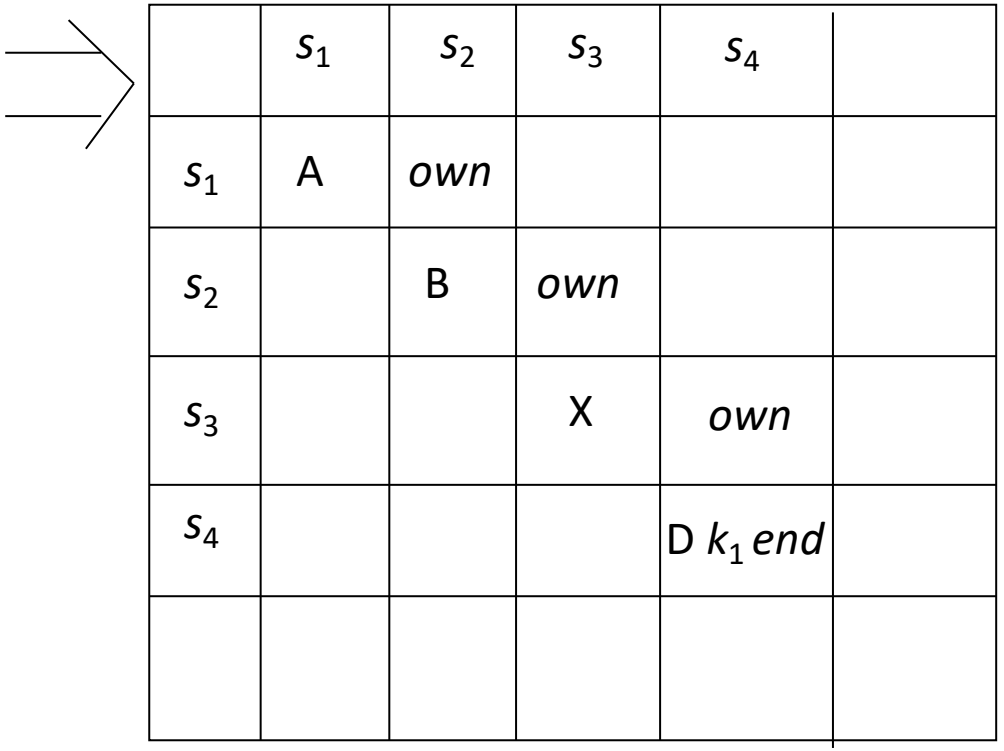


Current state is *k*

	s_1	s_2	s_3	s_4	
s_1	A	*own*			
s_2		B	*own*		
s_3			C *k*	*own*	
s_4				D *end*	

Mapping





	s_1	s_2	s_3	s_4	
s_1	A	*own*			
s_2		B	*own*		
s_3			X	*own*	
s_4				D k_1 *end*	

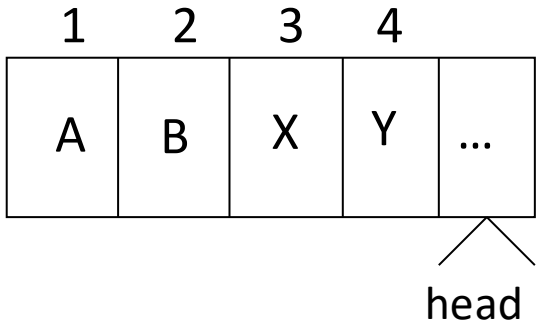
After $\delta(k, C) = (k_1, X, R)$ where k is the current state and k_1 the next state

Command Mapping

- $\delta(k, C) = (k_1, X, R)$ at intermediate becomes

```
command c_{k,C}(s_3, s_4)
if own in A[s_3, s_4] and k in A[s_3, s_3] and C in A[s_3, s_3]
then
  delete k from A[s_3, s_3];
  delete C from A[s_3, s_3];
  enter X into A[s_3, s_3];
  enter k_1 into A[s_4, s_4];
end
```

Mapping



After $\delta(k_1, D) = (k_2, Y, R)$ where k_1 is the current state and k_2 the next state

	s_1	s_2	s_3	s_4	s_5
s_1	A	*own*			
s_2		B	*own*		
s_3			X	*own*	
s_4				Y	*own*
s_5					*b* k_2 end

Command Mapping

- $\delta(k_1, \text{D}) = (k_2, \text{Y}, \text{R})$ at end becomes

```
command crightmost_{k,C}(s_4,s_5)
if end in A[s_4,s_4] and k_1 in A[s_4,s_4]
        and D in A[s_4,s_4]
then
  delete end from A[s_4,s_4];
  delete k_1 from A[s_4,s_4];
  delete D from A[s_4,s_4];
  enter Y into A[s_4,s_4];
  create subject s_5;
  enter own into A[s_4,s_5];
  enter end into A[s_5,s_5];
  enter k_2 into A[s_5,s_5];
end
```

Rest of Proof

- Protection system exactly simulates a TM
  - Exactly 1 *end* right in ACM
  - 1 right in entries corresponds to state
  - Thus, at most 1 applicable command
- If TM enters state q_f, then right has leaked
- If safety question decidable, then represent TM as above and determine if q_f leaks
  - Implies halting problem decidable, which we know is false
- Conclusion: safety question undecidable

Other Results

- Set of unsafe systems is recursively enumerable
- Without **create** primitive, safety question is complete in **P-SPACE**
- Without **destroy**, **delete** primitives, then safety question is undecidable
  - Systems are called “monotonic”
- Safety question for biconditional protection systems is decidable
- Safety question for monoconditional, monotonic protection systems is decidable
- Safety question for monoconditional protection systems with **create**, **enter**, **delete** (and no **destroy**) is decidable.

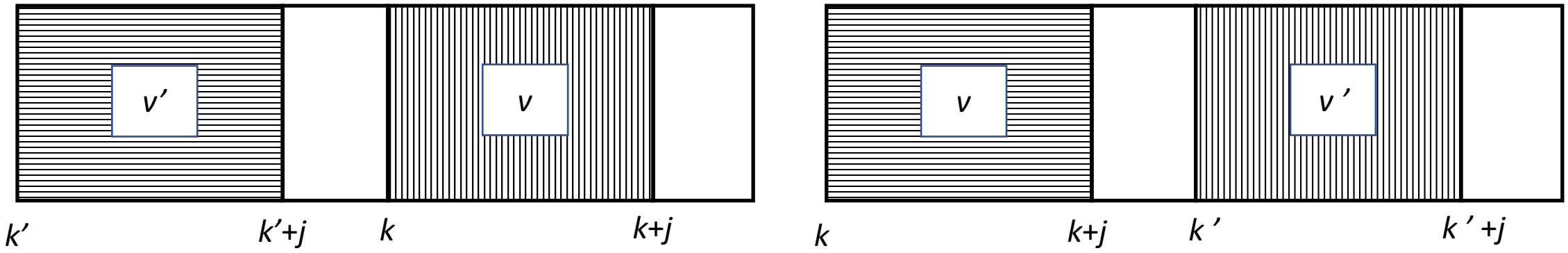
Theory of Viruses

- Is there a single algorithm that detects computer viruses precisely?
  - Need to define viruses in terms of Turing machines
  - See if we can map the halting problem into that algorithm

Step 1: Virus

- T Turing machine
  - s_v distinguished state of T
- V sequence of symbols on machine tape
- For every $v \in V$, when T lies at the beginning of v in tape square k, suppose that after some number of instructions are executed, a sequence $v' \in V$ lies on the tape beginning at location k', where either $k+|v| \leq k'$ or $k'+|v| \leq k$.
- (T, V) is a *viral set* and the elements of V are computer viruses.

In A Picture



- Virus *v* can copy another element of *V* either before or after itself on the tape
  - May not overwrite itself
  - Before at left, after at right

Theorem

- It is undecidable whether an arbitrary program contains a computer virus
- Terms in Proof:
  - *T* defines Turing machine
  - *V* defines sequence of tape symbols
  - $A, B \in M$ (tape symbols)
  - $q_i \in K$ for $i \geq 1$ (states)
  - *a*, *b*, *i*, *j* non-negative integers
  - δ: $K \times M \rightarrow K \times M \times \{L,R,-\}$ (transition function; – is no motion)

Overview of Argument

- Arbitrary *T*, sequence *S* of symbols on tape
- Construct second Turing machine *T′*, tape *V*, such that when *T* halts on *S*, *V* and *T′* create copy of *S* on tape
- *T′* replicates *S* iff *T* halts on *S*
  - Recall replicating program is a computer virus
- So there is a procedure deciding if (*T′*, *V*) is a viral set iff there is a procedure that determines if *T* halts on *S*
  - That is, if the halting problem is solvable

Proof

- Abbreviation for δ:

$$\delta(q_a, y) = (q_a, y, L) \text{ when } y \neq A$$

  means all definitions of δ where:
  - First element (current state) is q_a
  - Second element (tape symbol) is anything other than A
  - Third element is L (left head motion)

Abbreviations

- $LS(q_a, x, q_b)$
  - In state q_a, move head left until square with symbol x
  - Enter state q_b
  - Head remains over symbol x
- $RS(q_a, x, q_b)$
  - In state q_a, move head right until square with symbol x
  - Enter state q_b
  - Head remains over symbol x

Abbreviations

- $LS(q_a, x, q_b)$
  $\delta(q_a, x) = (q_b, x, -)$
  $\delta(q_a, y) = (q_a, y, L)$ when $y \neq x$
- $RS(q_a, x, q_b)$
  $\delta(q_a, x) = (q_b, x, -)$
  $\delta(q_a, y) = (q_a, y, R)$ when $y \neq x$

Abbreviation

- *COPY*(q_a, *x*, *y*, *z*, q_b)
  - In state q_a, move head right until square with symbol *x*
  - Copy symbols on tape until next square with symbol *y*
  - Place copy after first symbol *z* following *y*
  - Enter state q_b

Idea of Actions

- Put marker (A) over initial symbol
- Move to where to write it (*B*)
- Write it and mark location of next symbol (move *B* down one)
- Go back and overwrite marker *A* with symbol
- Iterate until *V* copied
  - Note: *A*, *B* symbols that do not occur in V

Abbreviation

$RS(q_a, x, q_{a+i})$

$\delta(q_{a+i}, x) = (q_{a+i+1}, A, -)$

- Move head over x, replace with marker A

$RS(q_{a+i+1}, y, q_{a+i+2})$

$RS(q_{a+i+2}, z, q_{a+i+3})$

- Skip to where segment is to be copied

$\delta(q_{a+i+3}, z) = (q_{a+i+4}, z, R)$

$\delta(q_{a+i+4}, u) = (q_{a+i+5}, B, -)$ for any $u \in M$

- Mark next square with B

More

- $LS(q_{a+i+5}, A, q_{a+i+6})$
- $\delta(q_{a+i+6}, A) = (q_{a+i+7}, x, -)$
  - Put x (clobbered by A) back
- $\delta(q_{a+i+7}, s_j) = (q_{a+i+5j+10}, A, R)$ for $s_j \neq y$
- $\delta(q_{a+i+7}, y) = (q_{a+i+8}, y, R)$
  - Overwrite symbol being copied (if last, enter new state)
- $RS(q_{a+i+5j+10}, B, q_{a+i+5j+11})$
- $\delta(q_{a+i+5j+11}, B) = (q_{a+i+5j+12}, s_j, R)$
  - Make copy of symbol

More

$\delta(q_{a+i+5j+12}, u) = (q_{a+i+5j+13}, B, -)$

- Mark where next symbol goes

$LS(q_{a+i+5j+13}, A, q_{a+i+5j+14})$

$\delta(q_{a+i+5j+14}, A) = (q_{a+i+7}, s_j, R)$

- Copy back symbol

$RS(q_{a+i+8}, B, q_{a+i+9})$

$\delta(q_{a+i+9}, B) = (q_b, y, -)$

- Write terminal symbol

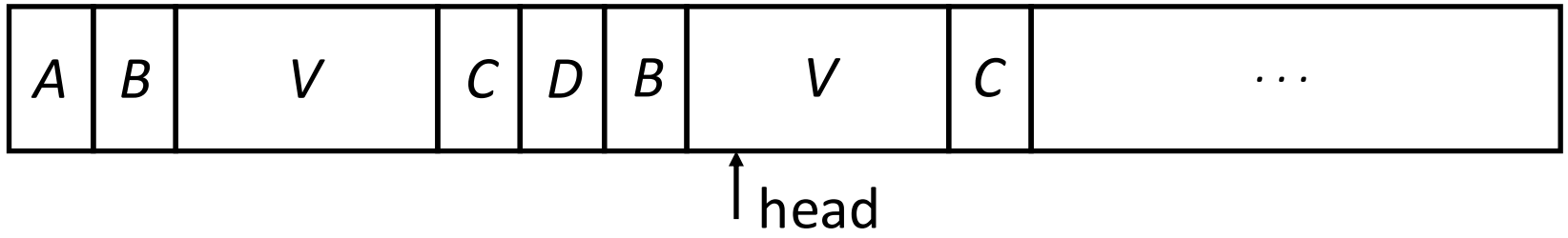
Construction of T', V'

- Symbols of T': $M' = M \cup \{ A, B, C, D \}$
- States of T' :

  $K' = K \cup \{ q_a, q_b, q_c, q_d, q_e, q_f, q_g, q_h, q_H \}$
- q_a initial state of T'
- q_H halting state of T'
- *SIMULATE*(q_f, T, q_h)
  - Simulate execution of T on tape with head at current position, q_f, q_h in K' correspond to initial, terminal state of T

T'

- Let $V' = (A, B, V, C, D)$.
- Idea: copy V after D, run T on V, and if it finishes, copy V over results
- Then if T halts, (T', V) a viral set by definition



Running T in T'

$\delta(q_a, A) = (q_b, A, -)$

$\delta(q_a, y) = (q_H, y, -)$ for $y \neq A$

- Beginning, halting transitions

$COPY(q_b, B, C, D, q_c)$

- Copy V after D

$LS(q_c, A, q_d)$

$RS(q_d, D, q_e)$

$\delta(q_e, D) = (q_e, D, R)$

- Position head so T executes copy of V

Running T in T'

$\delta(q_e, B) = (q_f, B, R)$

- Position head after B at beginning of copy of V

SIMULATE(q_f, T, q_h)

- T runs on copy of V (execution phase)

LS(q_h, A, q_g)

- T finishes; go to beginning of T' tape

COPY(q_g, A, D, D, q_H)

- Copy initial contents of V over results of running T on V (reproduction phase)

Analysis

- If *T* halts on *V*, definition of “viral set” and “virus” satisfied
- If *T* never halts on *V*, *V* never recopied, and definition never satisfied
- Establishes result

More General Result

- **Theorem:** It is undecidable whether an arbitrary program contains malicious logic

Why Elections?

Topic of current interest

- This election is important
- Most places will use some form of electronic voting (e-voting) systems
- Lots of FUD (fear, uncertainty, doubt) flying around
  - About election systems being inaccurate
  - About foreign actors rigging electronic voting machines
  - About requiring IDs to vote
  - …

Key Questions

- Does using computers in an election process:
  - Introduce new ways for attackers to compromise the election, or prevent voters from voting?
  - Stop any of the previous ways for attackers to compromise the election, or provide new ways to enable voters to vote?

Some Terms for E-Voting Systems

- BMD: Ballot Marking Device
  - Marks a paper ballot
- DRE: Direct Recording Electronic
  - Stores votes (ballots) electronically
- DRE + VVPAT: DRE + Voter Verified Paper Audit Trail
  - A DRE that also prints a paper record of the votes (ballots) cast on it
- PCOS: Precinct Count Optical Scanners
  - Used to count paper ballots at the precinct (polling station); these are stored electronically and the memory cards used to transfer results to central vote tabulator

Some Terms for Elections

- Race
  - An element on a ballot that people vote on
- Overvote
  - More votes cast by a voter in a particular race than is allowed for a voter
- Undervote
  - Fewer votes cast by a voter in a particular race than is allowed for a voter
- Example
  - Race is 3 open seats for city council, 5 candidates for those seats
  - I vote for 2 of them, not 3: that's an undervote and it counts
  - I vote for 4 of them, not 3: that's an overvote and it doesn't count

How an Election Works in Yolo County, CA

- Voters:
  - Go to polling station, give name, possibly proof of identity
  - Get ballot, enter booth, vote using marker to mark ballot
  - Put ballot in protective sleeve, leave booth
  - Drop ballot into ballot box
    - If provisional or conditional, put ballot and sleeve into envelope with voter's name, reason for the challenge (provisional) or condition (conditional) on the *outside*
- Vote-by-mail voters:
  - Fill in ballot
  - Put ballot into inner envelope
  - Put inner envelope into mailing envelope; sign the *outside* and mail it in

End of the Day

- Election officials take ballot box to County seat
- Election officials remove ballots from envelopes
  - Provisional and conditional ballots handled separately
- Ballots counted, put into bags marked with precinct and count
- Ballots removed from bag, run through automatic counters
  - Humans intervene when problems arise
  - Intermediate tallies written onto flash cards
  - Every so often, cards removed, walked to tally computer, inserted, votes counted
- Reported tallies periodically updated, given for posting to web

And Then . . .

- All places have provisional ballots
  - These are cast when it is unclear if the person is allowed to vote
  - In California, ***always*** on paper, never electronic
- California allows conditional ballots
  - These are cast by folks who register at the election (same day registration)
- Conditional and provisional ballots must be validated before being counted
- California also allows mail-in ballots arriving up to 3 days after Election Day to be counted

The Canvass

Required by California law:

- Ballots for 1% of precincts counted by hand
  - Chosen with throw of dice; if some races not in precincts selected, add more in until all covered
  - Some counties have legal authority to use risk-limiting audit as well or instead
  - In California, you *must* use paper for this (hence, all DREs have VVPATs)
- Compared to tallies from election
  - If different, must be reconciled
- Certify final counts to Secretary of State
  - Has to be done within some number of days after election

Some Election Requirements

- Voter validation (authenticated, registered, has not yet voted)
- Ballot validation (voter uses right ballot, results of marking capture intent of voter as required by law)
- Voter privacy, secrecy (no association between voter, ballot; includes preventing voter showing others how he/she voted)
- Integrity (ballots unchanged, votes tallied accurately)

Some Election Requirements

- Voting availability (voter must be able to vote, materials must be available)
- Voting reliability (voting mechanisms must work, even under adverse circumstances)
- Election manageability (process must be usable by those involved, including poll workers)
- Election transparency (audit election process, verify everything done right)