

CS 6600 Take Home Final Fall 2021

Matthew Whitesides

1) Considering the Voting System BLP model.

- a) The BLP model of the system does not currently function according to the Voting Process? The fact that “TM records v and produces c ” violates the *-Property of the BLP model by allowing the subject TM with a security level of **Medium** to write to the object c with a security level of **Low**.

First we can define our initial ACM state seen in Table 1.

To show this we can define our formal model as follows:

$$S = \{V, TM, E, M\}, O = \{b, c, v\}, P = \{r, w\}, C = \{High, Medium, Low\}, K = \{ALL\}$$

$$f_c(S) \in \{(Low\{ALL\}), (Medium\{ALL\}), (High\{ALL\})\}$$

$$f_o(O) \in \{(Low\{ALL\}), (Medium\{ALL\}), (High\{ALL\})\}$$

First V takes write access over b :

$$b_1 = \{(V, b, w)\}$$

$$f_c, 1(V) = \{(Low\{ALL\})\}$$

$$f_o, 1(b) = \{(Low\{ALL\})\}$$

$$V_1 = (b_1, m_1, f_1) \in V$$

Allowed in the actions (where $w_1 is V writestob$) and (V_1 is the initial state with the ACM m_1 change of w_1 V has write access over b):

$$X = w_1$$

$$Y = yes$$

$$Z = V_1$$

Next the TM requests read access over b (where $r_1 is TM readsfromb$) and (V_2 is the state V_1 with the ACM m_2 change of r_1 TM has read access over b):

$$b_2 = \{(TM, b, r)\}$$

$$f_c, 2(TM) = \{(Medium\{ALL\})\}$$

$$f_o, 2(b) = \{(Low\{ALL\})\}$$

$$V_2 = (b_2, m_2, f_2) \in V$$

Therefore:

$$X = w_1, r_1$$

$$Y = yes, yes$$

$$Z = V_1, V_2$$

Next the TM requests write access over E (where $w_2 is TM writestoE$) and (V_3 is the state V_2 with the ACM m_3 change of w_2 TM has write access over E):

$$b_3 = \{(TM, E, w)\}$$

$$f_c, 3(TM) = \{(Medium\{ALL\})\}$$

$$f_o, 3(E) = \{(High\{ALL\})\}$$

TABLE I
ELECTRONIC VOTING SYSTEM ACM

	V	TM	E	M	b	c	v
V	r,w	w	∅	∅	r,w	r	r
TM	r	r,w	w	∅	r	w	r
E	∅	∅	r,w	∅	r	r	∅
M	∅	∅	∅	r,w	∅	∅	∅
b	∅	∅	∅	∅	r,w	∅	∅
c	∅	∅	∅	∅	∅	r,w	∅
v	∅	∅	∅	∅	∅	∅	r,w

$$V_3 = (b_3, m_3, f_3) \in V$$

Therefore:

$$X = w_1, r_1, w_2$$

$$Y = yes, yes, yes$$

$$Z = V_1, V_2, V_3$$

Now the issue comes when the TM requests write access over c (where w_3 is TM writes to c):

$$b_4 = \{(TM, b, r)\}$$

$$f_c, 4(TM) = \{(Medium\{ALL\})\}$$

$$f_o, 4(c) = \{(Low\{ALL\})\}$$

$$V_3 = (b_3, m_3, f_3) \in V$$

This write down is not allowed in the BLP model therefore:

$$X = w_1, r_1, w_2, w_3$$

$$Y = yes, yes, yes, no$$

$$Z = V_1, V_2, V_3, V_3$$

We would have to place the TM in the Low-security category to fix this. That way, it could still write to E and read from the others but not have the issue of writing down. If we did that, our next steps would be:

$$b_4 = \{(TM, b, r)\}$$

$$f_c, 4(TM) = \{(Low\{ALL\})\}$$

$$f_o, 4(c) = \{(Low\{ALL\})\}$$

$$V_4 = (b_4, m_4, f_4) \in V$$

$$X = w_1, r_1, w_2, w_3$$

$$Y = yes, yes, yes, yes$$

$$Z = V_1, V_2, V_3, V_4$$

- b) Given that all voters are at the same security level and no categories exist, all voters are at the same security level as all voters. Therefore all V' would need to do to determine what V voted for was obtain their c , which given the security logic of this model, nothing is stopping them. Any voter can read any receipt/code and E trusts all queries made to it for a given c . How V' obtains V 's c is a question, but there's nothing about the model that stops it. Our change in part to lower the security level of the TM does not solve this issue. It potentially only worsens security as it may inadvertently enable voters to read from the TM, which is not good but allows it to still function according to the BLP model.

2) Considering an ND model.

- a) To show that TM's behavior is MSDND secure to E , we need to show that the Medium level TM cannot deduce anything about the set of High-level inputs to E .

TM has two inputs into E , "Report" and "Collection" there exists no valuation function on either of those inputs therefore, we can show the following claims for MSDND security.

TABLE II
REPORT INFORMATION IS MSDND SECURE.

1.	$\sim c = true$	Report information c is not normal.
2.	$w \models V^0 c(w) = true$	E cannot verify the c has been modified.
3.	$I_{1,0} r$	TM reports c to E .
4.	$B_1 I_{1,0} \sim c$	E believes the c from TM.
5.	$T_{1,0} \sim c$	E trusts the c .
6.	$B_1 I_{1,0} \sim c \wedge T_{1,0} \sim c \rightarrow B_1 \sim c$	E believes the c is correct.
7.	$w \models V_1 c(w) = true$	The verification procedure for c 's reported to the E always return true.

Similarly in respect to the vote v from the collection action is MSDND secure.

TABLE III
COLLECTION INFORMATION IS MSDND SECURE.

1.	$\sim v = true$	Collection information v is not normal.
2.	$w \models V^2 v(w) = true$	E cannot verify the v has been modified.
3.	$I_{1,0}r$	TM reports v to E.
4.	$B_1 I_{1,0} \sim v$	E believes the v from TM.
5.	$T_{1,0} \sim v$	E trusts the v .
6.	$B_1 I_{1,0} \sim v \wedge T_{1,0} \sim v \rightarrow B_1 \sim v$	E believes the v is correct.
7.	$w \models V_3 v(w) = true$	The verification procedure for v 's reported to the E always return true.

These claims show that for all traces TM has over E, we are MSDND secure, which is bad for the system as we hopefully trust the TM, but nothing is verifying the integrity of the data coming from it.

This nondeducibility can also be shown to be ND secure using the projection outputs of the various actions by various v and c 's coming from the TM as E outputs nothing regardless of the inputs from TM.

- i) $proj(TM, report(c)) = (\{E = \{\emptyset\}\})$
 $proj(TM, collection(v)) = (\{E = \{\emptyset\}\})$
- ii) $proj(TM, report(c')) = (\{E = \{\emptyset\}\})$
 $proj(TM, collection(v')) = (\{E = \{\emptyset\}\})$

- b) Like in part a, to show formally that V's behavior is MSDND secure to E we can show a claim using V's query action over E.

TABLE IV
QUERY INFORMATION IS MSDND SECURE.

1.	$\sim c = true$	Query information c is not normal.
2.	$w \models V^0 c(w) = true$	E cannot verify the c has been modified.
3.	$I_{1,0}r$	V reports c to E.
4.	$B_1 I_{1,0} \sim c$	E believes the c from V.
5.	$T_{1,0} \sim c$	E trusts the c .
6.	$B_1 I_{1,0} \sim c \wedge T_{1,0} \sim c \rightarrow B_1 \sim c$	E believes the c is correct.
7.	$w \models V_1 c(w) = true$	The verification procedure for c 's reported to the E always return true.

This nondeducibility can also be shown to be ND secure using the projection outputs of multiple queries by voters resulting in the same output traces from E.

- i) $proj(V, query(c)) = (\{E = \{\emptyset\}\})$
- ii) $proj(V, query(c')) = (\{E = \{\emptyset\}\})$

Therefore regardless of the input query's from V we have the same outputs from E making V's behavior nondeducible to E.

- c) Show, formally, that TM's behavior is:

- i) To show that TM is MSDND and ND secure to V we have the following available actions and outputs.

TABLE V
INSERT INFORMATION IS MSDND SECURE.

1.	$\sim c = true$	Insert information b is not normal.
2.	$w \models V^0 b(w) = true$	TM cannot verify the b has been modified.
3.	$I_{1,0}r$	V reports b to TM.
4.	$B_1 I_{1,0} \sim b$	TM believes the b from V.
5.	$T_{1,0} \sim b$	TM trusts the v .
6.	$B_1 I_{1,0} \sim b \wedge T_{1,0} \sim b \rightarrow B_1 \sim b$	TM believes the b is correct.
7.	$I_{0,1}c$	TM reports c to V.
8.	$w \models V_1 b(w) = true$	The verification procedure for b 's reported to the TM always return true.

We can also show ND security by the projections outputs that TM has from multiple V's actions.

$$proj(V_0, insert(b)) = (\{TM = \{c_0\}\})$$

$$proj(V_1, insert(b)) = (\{TM = \{c_0, c_1\}\})$$

And to show this cannot be deduced from another V's input we can project the same result given different Vs.

$$\begin{aligned} \text{proj}(V_2, \text{insert}(b)) &= (\{TM = \{c_0\}\}) \\ \text{proj}(V_3, \text{insert}(b)) &= (\{TM = \{c_0, c_1\}\}) \end{aligned}$$

- ii) TM to V is not Non-Interference secure in that they are in two separate security levels and V can pull different outputs from the TM given various lower level inputs from V.

For example to show this formally we can see the projection of a voter using various codes gives a different output from E depending on the input c (where C_s is the query command for that voter's c).

$$\text{proj}(V_0, C_s, \sigma_0) = ? \text{Proj}(V, \pi V_1(C_s), \sigma_0)$$

Gives us the following outputs of:

$$\begin{aligned} \text{proj}(V_0, C_s, \sigma_0) &= \{true, v_0\} \\ \text{proj}(V, \pi V_1(C_s), \sigma_0) &= \{true, v_1\} \\ \{true, v_0\} &\neq \{true, v_1\} \end{aligned}$$

This be even more clearly seen if V queries any code that does not exist in the E.

$$\begin{aligned} \text{proj}(V_0, C_s(c), \sigma_0) &= \{true, v_0\} \\ \text{proj}(V_0, C_s(c'), \sigma_0) &= \{false, \emptyset\} \\ \{true, v_0\} &\neq \{false, \emptyset\} \end{aligned}$$

- d) We can conclude that TM and V are nondeducibility secure regarding E, which is not a good thing from the systems perspective. There exists no verification on E's part that the reports and collections that come in are valid, as it cannot determine a difference between the ballots and codes being input into it. Also, the database currently has no output information, so there's nothing for the human validators to get any info from E if needed (which those people would probably need to be at the higher security level). However, in reality, this part of the system is relatively secure even if it's MSDND secure, as long as the TM is not interfered with, but if it were, you'd have no way of knowing.

The more significant issue, in my opinion, is the non-NI security from the voters to the E, as they can seemingly endlessly query the E and get back query information. Depending on how secure the codes that come back from the TM are, an attacker may deduce what votes are in the election database by trying out various codes. This attack would never be detected since the E has no output info or verification procedures.

- 3) A Clark Wilson style integrity verification procedures integrated into the monitor could solve our MSDND security issue from the TM to E. First, the E could use the monitor to verify the output from the TM before it gets to E. It also could enable E to output information after it has received input to introduce some deducibility.

We can set up our Clark Wilson IVP model with the monitor M like so:

ACKNOWLEDGMENT

The author would like to thank Professor Bruce McMillin with the Department of Computer Science, Missouri University of Science and Technology.