An Intelligent Board of Security Countermeasure Cases in Prolog.

Frank Appiah
An Intelligent Board of Security Countermeasure Cases in Prolog.

Prof Frank Appiah AKA FAEng PhD (KCL)
Email: appiahnsiahfrank@gmail.com

Abstract. This report is an approach in defining the execution cases of an intelligent board of a security director's countermeasure cases with a reasoner in Prolog.

Keywords. reasoner, intelligent, storyboard, logic, program, symbolic, case.

1 Introduction.

Dimensions of counterfeiting is a security identification tool to protect your computer with information and help to inform other users of all security measures.

The international dimensions of counterfeiting by duplication, thefting by deletion replacement or insertion replacement, cyberattacking, hacking / cracking on internet / decentralized network is strong gating with incorrect measure, unguarding on access breach, unlawful entry, uncontrolling access system codes, momenting by passby fights, intern replacement unverifiables and unvalidating information are exploitations that needs to be addressed.

Vulnerability is a weakness in the security system. A threat is blocked by control of vulnerability.

The dimensions above are used to reason about counter measures in security system. A control is a protective measure used as an action, procedure or technique. Simply, this security measure research report is addressing the following:

- Creation of security controls in unceiled secret information.
- Laying out risk of unceiled secret information and ways of dealing with it.
- Certain on ways of document process - sing with digitized image water-marking.
- Middle aging of counterfeiting by duplication with deletion replacement and insertion replacement.
- A decentralized network with marginal error on control printing with water marking process.
- A counterattack measure in validating and verification of authentic document.

If it is possible or necessary watermarking secret Ceil should be used to prevent ruining access control. Then it should be used.

The security measures are again used in the cases of reasoning. If counterfeiting by duplication creates a methods of recovery in risky information. Then, it should be recover after incident.

If a security officer or engineer can address duplication copy in cases an attacker deletes and insert a counterfeit copy to be used by the document marker thereby making information lose confidentiality or integrity. Then, it should be engineered for counter measuring. If in a decentralized scenario, the document marker will be able to authenticate as usual to able to have access to the digital document to make a copy for further processing. Then, it should create authenticated access. If the technique of authentication and authorization can be by password or biometry (fingers, iris, height etc). Then it should create technology for culturing and socializing the security process of characterization.

If file transfer protocol gives the decentralized manner of network access with secure means. Then it should create confidentiality and availability in the security process. If Unceil secret paper creates vulnerabilities and embarrassment in ruining the authenticity of document. Then it should leave the security room of vulnerabilities. If security agents unchase theft document in a vulnerable situation. Then it should be way to dismissal from the work place.

If it is hard and difficult to physically timestamp all documents at a security site. Then watermarking by stamping should be the way to countermeasure. If vulnerabilities prevention is a means to countermeasure a counterfeit information.

Then finally a Ceil by watermarking should be used.
2 Director Assessment.

A security director or officer or engineer addressing duplication copy in cases an attacker deletes and insert a counterfeit copy to be used by the document marker thereby making information lose confidentiality or integrity.

In the middle ages of counterfeiting by duplication, a copy of existing image is kept with the security officer or engineer on deletion replacement or insertion replacement. In the castle of counterfeiting by duplication, a different but approved image is quickly inserted into the document processing of the watermarking paper. Then it is casted into decentralized networks with a marginal error on the previous information dissemination from the control printer software. The fortress of counterfeiting by duplication a security officer will counterattack with an invalid document fight in the sense of seizing and requesting a reprint of information to process new.

In previous work[4], a Director will initiate the main rule of the xProlog application which is coded as below:

```prolog
main :-
1. nl,
2. write('Security Director Program.'), nl,
3. write('>   Enter a selection followed by a period.'), nl,
4. write('>   1. Yes, countermeasure'), nl,
5. write('>   2. Exit'), nl, nl,
6. read(Choice), nl,
7. assess_opt(Choice), nl,
8. main.
```

It will first move the cursor to a newline, coded as blue. It will display a text "Security Director Program" on the monitor, coded on line 2 and a newline is called. A selection is read to be assessed with the assess_opt rule of the program on line 7. The assessment is a forward chain of cdd abbreviation rules, count of 12. After each rule has successfully run then the next cdd rule and final cdd then runs the main rule again. This is how the security director works. It identifies each Vulnerability and ask you to counter measure until all assessments are done at the office. This new
director uses a case-based approach to inform the director of routine checks to be made. This case-based approach uses a main rule as shown below:

```prolog
main :-
    nl,
    tab(4),write('Security Cases Reasoner.'), nl,
    tab(2),write('Developed by: Frank Appiah'), nl, nl,
    write('>   Enter a selection followed by a period.'), nl,
    write('>   1. Thefting Case'), nl,
    write('>   2. Insertion Replacement Case'), nl,
    write('>   3. Cyberattack Case'), nl,
    write('>   4. Hacking-Crack Case'), nl,
    write('>   5. Strong Gating Case'), nl,
    write('>   6. Access Breach Case'), nl,
    write('>   7. Unlawful Entry Case'), nl,
    write('>   8. Uncontrolled System Codes Case'), nl,
    write('>   9. Passby Fight Case'), nl,
    write('>  10. Intern Replacement Case'), nl,
    write('>  11. Invalid Information Case'), nl,
    write('>  12. Exit'), nl, nl,
    read(Choice),
    cdd(Choice), main.
    initialization('main').
```

The main is initialized just like the other counterpart. It then writes to display about 14 messages of instruction and a selection menu. The call instructions are each separated by a comma. The read instructions is a system call by the logic program. It allows input to be read into the running task. The case based approach is implemented with cdd(N) head rule where N is the number of case rules to execute or run. Here, there are about 12 of such rules. Same rule name is used but the passing head value is an increment of 1. The following are the cdd head rules:

- cdd(1),
- cdd(2),
- cdd(3),
- ....
- cdd(12).

An execution of the cdd head will in turn run the main in a loop. This is an interpreter application kind of evaluate return to loop program. The cdd(12) rule is a quit body in the application. A successful execution run did show the following in the previous program:

```
Security Director Program.
> Enter a selection followed by a period.
> 1. Yes, countermeasure
> 2. Exit.
```
This now shows as this:

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.

1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit

Code: [Security Case Reasoner]

cdd(1) :-
  nl,
  write('Identify thefting by deletion replacement.'), nl,
  write('>   Yes, countermeasure on thefting by deletion'), nl.

Code: [Security Director [4]- Prolog Code in Appendix 1].

cddtheft(1) :-
  nl,
  write('Identify thefting by deletion replacement.'), nl,
  write('>   Enter a selection followed by a period.'), nl,
  write('>   1. Yes, countermeasure on thefting by deletion'), nl,
  write('>   2. Exit'), nl, nl,
  read(Choice),
  cddrep(Choice),
  main(Choice).

The head rule cdd(1) is the new used to achieve the same information as cddtheft head rule. This prolog application is a mobile application that a security officer or engineer can use to run security routine with a team in countermeasure strategies.

The logic program is programmed in XProlog Android on Honor model from Huawei corporation. The benchmark set from AI Expert on this device is shown in Appendix 3.
Conclusion.

This report has provided a case based approach in programming an intelligent board of directors security countermeasure in logic programming, Prolog. This is original in reasoning on information security measures as if-then reasoning cases. System design of Security Case Reasoner is in terms of source code(input) of the logic program and run outputs that are shown in this report. The contrast between these two logic programs is essential in computing itself.

Further Reading


Appendix 1.

Prolog Code.

cdd(1) :-
    nl,
    write('Identify thefting by deletion replacement.'), nl,
    write('> Yes, countermeasure on thefting by deletion'), nl.

cdd(2) :-
    write('Identify insertion replacement at office.'), nl,
    write('> Yes, countermeasure on insertion replacement'), nl.

cdd(3) :-
    write('Identify cyberattack on network at office or home.'), nl,
    write('> Yes, countermeasure on cyberattacking.'), nl.

cdd(4) :- write('Identify hacking /cracking on internet/decentralized network.'), nl,
    write('> Yes, strong countermeasure on hacking/cracking.'), nl.

cdd(5) :- write('Identify if it is strong gating with incorrect measure.'), nl,
write("> Yes, countermeasure on strong gating."), nl.

cdd(6) :- write("Identify if unguarding on access breach."), nl,
write("> Yes, countermeasure by guarding on access breach."), nl.

cdd(7) :- write("Identify if there is an unlawful entry."), nl,
write("> Yes, countermeasure on unlawful entry."), nl.

cdd(8) :- write("Identify if it is an uncontrolling access system codes."), nl,
write("> Yes, countermeasure on controlling access system codes."), nl.

cdd(9) :- write("Momenting by passby fights is recalled."), nl,
write("> Yes, countermeasure on passby riot is checked."), nl.

cdd(10) :-
write("Identify if it caused by intern replacement unverifiables and unvalidatiables."), nl,
write("> Yes, countermeasure by checking on intern replacement."), nl.

cdd(11) :-
write("Identify if it is invalidating information"), nl,
write("> 1. Yes, countermeasure check on invalid information."), nl.

%cdd(_) :-
% write("Unknown operation").

cdd(12) :- write("Quiting..."), halt.

main :-
  nl,
  tab(4), write("Security Cases Reasoner."), nl,
  tab(2), write("Developed by: Frank Appiah"), nl, nl,
  write("> Enter a selection followed by a period."), nl,
  write("> 1. Thefting Case"), nl,
  write("> 2. Insertion Replacement Case"), nl,
  write("> 3. Cyberattack Case"), nl,
  write("> 4. Hacking-Crack Case"), nl,
  write("> 5. Strong Gating Case"), nl,
  write("> 6. Access Breach Case"), nl,
  write("> 7. Unlawful Entry Case"), nl,
  write("> 8. Uncontrolled System Codes Case"), nl,
  write("> 9. Passby Fight Case"), nl,
  write("> 10. Intern Replacement Case"), nl,
  write("> 11. Invalid Information Case"), nl,
  write("> 12. Exit"), nl, nl,
  read(Choice),
  cdd(Choice), main.

initialization("main").
Appendix 2: Execution Runs.

Security Cases Reasoner.  
Developed by: Frank Appiah

1. Enter a selection followed by a period. 
   1. Thefting Case 
   2. Insertion Replacement Case 
   3. Cyberattack Case 
   4. Hacking-Crack Case 
   5. Strong Gating Case 
   6. Access Breach Case 
   7. Unlawful Entry Case 
   8. Uncontrolled System Codes Case 
   9. Passby Fight Case 
  10. Intern Replacement Case 
  11. Invalid Information Case 
  12. Exit

1. Identify thefting by deletion replacement.
   Yes, countermeasure on thefting by deletion

Security Cases Reasoner.  
Developed by: Frank Appiah

1. Enter a selection followed by a period. 
   1. Thefting Case 
   2. Insertion Replacement Case 
   3. Cyberattack Case 
   4. Hacking-Crack Case 
   5. Strong Gating Case 
   6. Access Breach Case 
   7. Unlawful Entry Case 
   8. Uncontrolled System Codes Case 
   9. Passby Fight Case 
  10. Intern Replacement Case 
  11. Invalid Information Case 
  12. Exit

2. Identify insertion replacement at office. 
   Yes, countermeasure on insertion replacement

Security Cases Reasoner.  
Developed by: Frank Appiah

1. Enter a selection followed by a period. 
   1. Thefting Case 
   2. Insertion Replacement Case 
   3. Cyberattack Case 
   4. Hacking-Crack Case 
   5. Strong Gating Case 
   6. Access Breach Case 
   7. Unlawful Entry Case 
   8. Uncontrolled System Codes Case 
   9. Passby Fight Case 
  10. Intern Replacement Case 
  11. Invalid Information Case
Identify cyberattack on network at office or home.
> Yes, countermeasure on cyberattacking.

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.
1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit

Identify hacking /cracking on internet/decentralized network.
> Yes, strong countermeasure on hacking/cracking.

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.
1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit

Identify if it is strong gating with incorrect measure.
> Yes, countermeasure on strong gating.

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.
1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
6. Identify if guarding on access breach.
   > Yes, countermeasure by guarding on access breach.

Security Cases Reasoner.
Developed by: Frank Appiah

7. Identify if there is an unlawful entry.
   > Yes, countermeasure on unlawful entry.

Security Cases Reasoner.
Developed by: Frank Appiah

8. Identify if it is an uncontrolling access system codes.
   > Yes, countermeasure on controlling access system codes.

Security Cases Reasoner.
Developed by: Frank Appiah
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit

9. Momenting by passby fights is recalled.
Yes, countermeasure on passby riot is checked.

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.
1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit

10. Identify if it caused by intern replacement unverifiables and unvalidatiables.
Yes, countermeasure by checking on intern replacement.

Security Cases Reasoner.
Developed by: Frank Appiah

Enter a selection followed by a period.
1. Thefting Case
2. Insertion Replacement Case
3. Cyberattack Case
4. Hacking-Crack Case
5. Strong Gating Case
6. Access Breach Case
7. Unlawful Entry Case
8. Uncontrolled System Codes Case
9. Passby Fight Case
10. Intern Replacement Case
11. Invalid Information Case
12. Exit


<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Iterations</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>tail_call_atom_atom</td>
<td>50,000</td>
<td>6.20</td>
</tr>
<tr>
<td>binary_call_atom_atom</td>
<td>50,000</td>
<td>9.60</td>
</tr>
<tr>
<td>Function</td>
<td>50,000</td>
<td>10,000</td>
</tr>
<tr>
<td>-------------------------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>cons_list</td>
<td>50,000</td>
<td>6.80</td>
</tr>
<tr>
<td>walk_list</td>
<td>50,000</td>
<td>5.20</td>
</tr>
<tr>
<td>walk_list_rec</td>
<td>50,000</td>
<td>6.60</td>
</tr>
<tr>
<td>args(1)</td>
<td>50,000</td>
<td>6.20</td>
</tr>
<tr>
<td>args(2)</td>
<td>50,000</td>
<td>8.80</td>
</tr>
<tr>
<td>args(4)</td>
<td>50,000</td>
<td>14.80</td>
</tr>
<tr>
<td>args(8)</td>
<td>50,000</td>
<td>25.20</td>
</tr>
<tr>
<td>args(16)</td>
<td>50,000</td>
<td>46.40</td>
</tr>
<tr>
<td>cons_term</td>
<td>50,000</td>
<td>8.40</td>
</tr>
<tr>
<td>walk_term</td>
<td>50,000</td>
<td>7.40</td>
</tr>
<tr>
<td>walk_term_rec</td>
<td>50,000</td>
<td>7.00</td>
</tr>
<tr>
<td>shallow_backtracking</td>
<td>50,000</td>
<td>5.20</td>
</tr>
<tr>
<td>deep_backtracking</td>
<td>50,000</td>
<td>17.80</td>
</tr>
<tr>
<td>trail_variables</td>
<td>50,000</td>
<td>15.40</td>
</tr>
<tr>
<td>medium_unify</td>
<td>50,000</td>
<td>0.80</td>
</tr>
<tr>
<td>deep_unify</td>
<td>10,000</td>
<td>1.00</td>
</tr>
<tr>
<td>integer_add</td>
<td>10,000</td>
<td>9.00</td>
</tr>
<tr>
<td>floating_add</td>
<td>10,000</td>
<td>10.00</td>
</tr>
<tr>
<td>arg(1)</td>
<td>50,000</td>
<td>22.80</td>
</tr>
<tr>
<td>arg(2)</td>
<td>50,000</td>
<td>23.20</td>
</tr>
<tr>
<td>arg(4)</td>
<td>50,000</td>
<td>20.60</td>
</tr>
<tr>
<td>arg(8)</td>
<td>50,000</td>
<td>18.40</td>
</tr>
<tr>
<td>arg(16)</td>
<td>50,000</td>
<td>18.60</td>
</tr>
<tr>
<td>index</td>
<td>20,000</td>
<td>8.50</td>
</tr>
<tr>
<td>assert_unit</td>
<td>10,000</td>
<td>98.00</td>
</tr>
<tr>
<td>access_unit</td>
<td>10,000</td>
<td>92.00</td>
</tr>
<tr>
<td>slow_access_unit</td>
<td>10,000</td>
<td>94.00</td>
</tr>
<tr>
<td>setof</td>
<td>10,000</td>
<td>43.00</td>
</tr>
<tr>
<td>pair_setof</td>
<td>10,000</td>
<td>67.00</td>
</tr>
<tr>
<td>double_setof</td>
<td>10,000</td>
<td>676.00</td>
</tr>
<tr>
<td>bagof</td>
<td>10,000</td>
<td>27.00</td>
</tr>
</tbody>
</table>

28,110 msec