An Intelligent Board of Security Countermeasure Cases in Prolog.

Frank Appiah

November 21, 2020

An Intelligent Vulnerable Prevention Specification and Factual Response in Prolog.

Prof Frank Appiah AKA FAEng PhD (KCL)
Email:appiahnsiahfrank@gmail.com
12.2020.

Abstract. This review is on intelligible security measures in vulnerable prevention knowledge specification in predicate syntax computation in representing facts and rules of countermeasures in such exploitations.

Keywords. AI, intelligent, predicates, logic, programming, syntax, knowledge, representational language.

1 Introduction.

In specifying predicate syntax computations in Prolog, there is a way to get factual response in natural alternatives - Yes or No.

On responding to vulnerable preventions at an office to weakness to security systems. There is a need to hold in facts specific to the security measures identifiable to the issues. Creating a knowledgeable database for yes responses is naturally important to the security officer or controller. On possible basis of this security requirement is a capture of countermeasure facts in this security concerns. In capturing, the international dimensions of counter -feiting by duplication, thefting by deletion replacement or insertion replacement, cyberattacking, hacking / cracking on internet / decentralized network is strong gating with incorrect measure, ungaurding on access breach, unlawful entry, uncontrolling access system codes, momenting by passby fights, intern replacement unverifiables and unvalidating information are exploitations addressed.

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

A control is a protective measure used as an action, procedure or technique. Simply, this security measure research review 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.

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 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 <u>should be used.</u>

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.

2 Predicates in Security Dimensions.

Predicate sentences are a set of specifications of facts in Prologue database with auxiliary factual responses in natural environment being interpreted in validable and verifiable way.

Here, sentences in the security dimensions are captured as knowledge base on countermeasure facts in Prolog predicate syntax database.
I will capture the natural sentence along the predicate syntax. These dimensions are all captured as cm_check(a, o) predicate syntaxes. Note a dot on every predicate to indicate end term and begin another asserting in the database.

- cm_check(invalid,info).
- cm_check(intern,replacement).
- cm_check(passby,riot).
- cm_check(access,system_codes).
- cm_check(unlawful,entry).
- cm_check(guard,access_breach).
- cm_check(strong_gating,incorrect).
- cm_check(hack,internet).
- cm_check(crack,decentralized).
- cm_check(cyberattack,home_net).
- cm_check(cyberattack,office_net).
- cm_check(insertion,office).
- cm_check(replacement,office).
- cm_check(thefting,deletion).
- cm_check(thefting,replacement).
- cm_check(countermeasure,duplication).

- cm_check(counterfeit,duplication).

There are 17 *cm_check* facts in the set of specifications of the database. Reading the predicate sentences will create sentential languages like:

- A countermeasure check on information is invalidity.
- A document replacement is a check on countermeasure on intern.
- A riot check is countermeasure on passby.
- System code access should be countermeasure check.
- Unlawful entry should be check in countermeasure.
- Strong gating is incorrect and should be check in countermeasures.
- Hacking on internet is checked in countermeasure.
- Countermeasures should be checked on cracking decentralized networks.
- Cyberattacking should be countermeasure checked.
- Electronic office insertion should be checked on countermeasure.
- Electronic office replacement should be checked on countermeasure.
- Electronic deletion is thefting that should be checked in countermeasure.
- Electronic replacement is thefting that should be checked in countermeasure.
- Duplication of countermeasure should be checked in countermeasure.

The next set of facts are based on the list of security measures. The sentences in predicate syntax includes the following :

- cm_problem(unceiled,secret).
- cm_problem(unlaid,secret).
- cm_problem(risky,secret).
- cm_problem(middle_aiding,deletion).
- cm_problem(middle_aiding,insertion).
- cm_problem(middle_aiding,replacement).
- cm_problem(invalid,document).
- cm_problem(unverified,document).

There are about 8 *cm_problem* predicates in the security measure of all worlds. Reading the predicate sentences will create sentential languages like

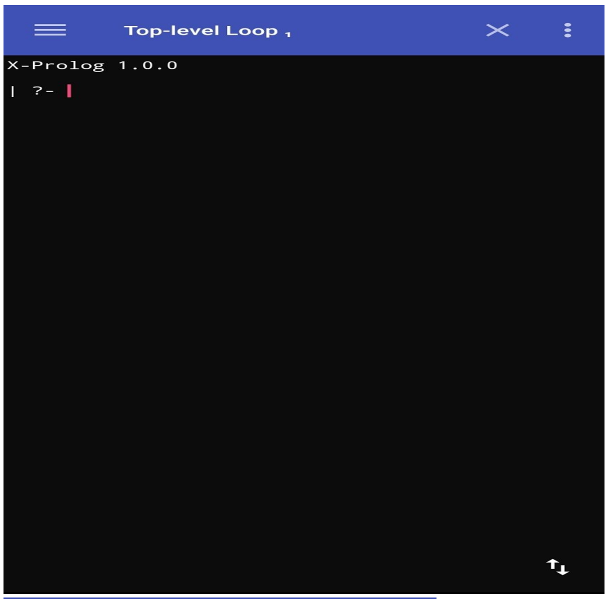
- Secret information can be a risky problem.
- Secret information in its unceiled form can be a problem.
- Secret information unlaid is a problem in countermeasure
- Aiding deletion of information is a problem.
- Aiding insertion of information is a problem.
- Aiding replacement of information is a problem.
- Invalidating a document is a problem.
- Unverified document is a problem.

3 Factual Response Assessment.

Yes/No assessment is a factual response in directing on existence of facts in Security domain. The predicate sentences described above is to run in consult top-level loop mode. This means that a Director will initiate the main assertion of the xProlog application based on loading a database file containing the security facts which are coded as predicate syntax. In an unloaded loop, response on one of the cm_check fact yields:

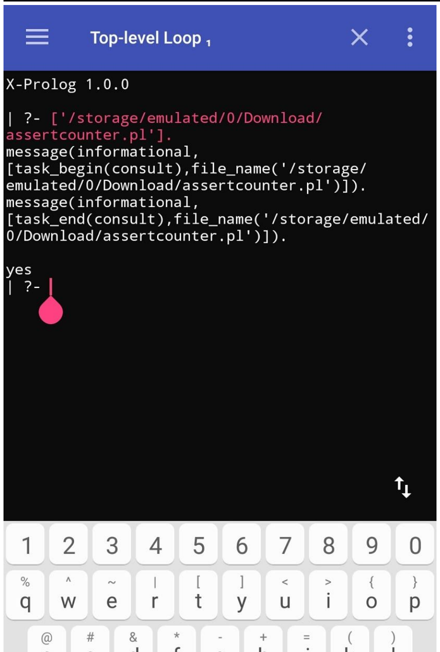
```
X-Prolog 1.0.0

| ?- cm_check(invalid,info).
message(error, error(existence_error(procedure,cm_check/2),call(user:cm_check(invalid,info)))).
```

It shows an error message and it claims an existence error with the predicate that takes two arguments in calling the cm_check predicate.



```
X-Prolog 1.0.0
| ?-
```

It is now time to consult the database. This is done by Clicking on the Run button at the top of a running XProlog application. This is a mobile version on my Android phone. Then you will select the Top-level loop sidebar menu. A new window will be pop up showing a black screen on the prompt. It is as shown. Next is to select consult on the top dot menu. This will open the file system view to enable you to select the prolog file with (pl) extension. This show the prompt as below:



```
X-Prolog 1.0.0

| ?- ['/storage/emulated/0/Download/
assertcounter.pl'].
message(informational,
[task_begin(consult),file_name('/storage/
emulated/0/Download/assertcounter.pl')]).
message(informational,
[task_end(consult),file_name('/storage/emulated/
0/Download/assertcounter.pl')]).

yes
| ?-
```

The prompt response is yes after the consultation. That means we are ready to go checking facts in database. Let the oracle consult us good wishes in this check.

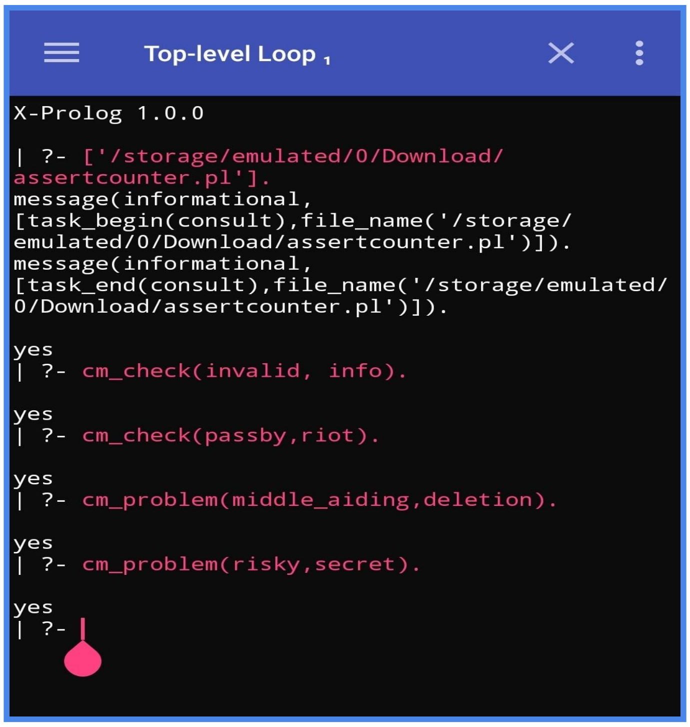
```
cm_check(invalid, info).
```

Response:

```
| ?- cm_check(invalid, info).
yes
| ?-
```

The consult of the fact has responded yes. The security measure database has confirm the claim of this fact. The countermeasure team will have place a check on the fact on occurrence.

The second check is on the fact:



```
X-Prolog 1.0.0

| ?- ['/storage/emulated/0/Download/
assertcounter.pl'].
message(informational,
[task_begin(consult),file_name('/storage/
emulated/0/Download/assertcounter.pl')]).
message(informational,
[task_end(consult),file_name('/storage/emulated/
0/Download/assertcounter.pl')]).

yes
| ?- cm_check(invalid, info).

yes
| ?- cm_check(passby,riot).

yes
| ?- cm_problem(middle_aiding,deletion).

yes
| ?- cm_problem(risky,secret).

yes
| ?-
```

Response:

```
| ?- cm_check(passby,riot).
yes
```

There is a countermeasure check on the database to find out if passby on riot is a fact. The response is yes. This review shows that not only that the consult helps in building the database without asserting before or after but it also statically response to hold the facts on predicate cm_check.

I will now run a few cm_problem on the Prolog Interpreter.

```
cm_problem(middle_aiding,deletion).
```

Response:

```
| ?- cm_problem(middle_aiding,deletion).

yes
```

Again, this is interpreted on the prompt:

```
cm_problem(risky,secret).
```

Response.

```
| ?- cm_problem(risky,secret).

yes
```

Finally on two response request :

```
- cm_problem(middle_aiding,replacement).
cm_problem(invalid,document).
```

Response:

```
| ?- cm_problem(middle_aiding,replacement).
cm_problem(invalid,document).

yes
| ?-
yes
| ?-
```

The rest of responses are shown in the appendix 2 and the screenshot view above.

Conclusion.

This is a review on intelligent vulnerable prevention specification and yes/no assessment response. A total of about 28 facts were interpreted. There were about three categorical predicate sentences in this review only. These predicates were enumerated and along with sentences in natural language processed. This prolog interpretation is a mobile interpreter 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.

Further Reading

[1] Frank Appiah (2020). Security Controls or Countermeasures: Vunerabilities Prevention, Easychair Preprint 4410.

[2] XProlog (2020). XProlog Android Package, Playstore. Programming IDE. Online Accessed.
[3] Android Mobile (2020). Huawei Device Honor Model. Programming with Footprint Minicomputers. Huawei Corporation.

Appendix 1. Prolog Code.

FACTS: countermeasure.pl

```
cm_check(invalid,info).
cm_check(intern,replacement).
cm_check(passby,riot).
cm_check(access,system_codes).
cm_check(unlawful,entry).
cm_check(guard,access_breach).
cm_check(strong_gating,incorrect).
cm_check(hack,internet).
cm_check(crack,decentralized).
cm_check(cyberattack,home_net).
cm_check(cyberattack,office_net).
cm_check(insertion,office).
cm_check(replacement,office).
cm_check(thefting,deletion).
cm_check(thefting,replacement).
cm_check(countermeasure,duplication).
cm_check(counterfeit,duplication).

cm_protect(action,procedure,technique).
cm_protect(vulnerable,weakness,security).

cm_problem(unceiled,secret).
cm_problem(unlaid,secret).
cm_problem(risky,secret).
cm_problem(middle_aiding,deletion).
cm_problem(middle_aiding,insertion).
cm_problem(middle_aiding,replacement).
cm_problem(invalid,document).
cm_problem(unverified,document).
```

Appendix 2: Execution Runs.

X-Prolog 1.0.0

```
| ?- ['/storage/emulated/0/Download/assertcounter.pl'].
```

```
message(informational,
[task_begin(consult),file_name('/storage/emulated/0/Download/assertcounter.pl')]).
message(informational,
[task_end(consult),file_name('/storage/emulated/0/Download/assertcounter.pl')]).

yes
| ?- cm_check(invalid, info).

yes
| ?- cm_check(passby,riot).

yes
| ?- cm_problem(middle_aiding,deletion).

yes
| ?- cm_problem(risky,secret).

yes
| ?- cm_problem(middle_aiding,replacement).

yes
| ?- cm_problem(middle_aiding,replacement).
cm_problem(invalid,document).

yes
| ?-
yes
| ?- cm_check(guard,access_breach).
cm_check(strong_gating,incorrect).

yes
| ?-
yes
| ?- cm_check(intern,replacement).

yes
| ?- cm_check(cyberattack,home_net).
cm_check(cyberattack,office_net).
cm_check(insertion,office).
cm_check(replacement,office).

yes
| ?-
yes
| ?-
yes
| ?-
yes
| ?- cm_protect(action,procedure,technique).

yes
| ?- cm_protect(action,procedure,technique).
```

```
cm_protect(vulnerable,weakness,security).

yes
| ?-
yes
| ?-
```

Appendix 3. Benchmark Set from AI Expert magazine.

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

28,110 msec