ARMY RESEARCH LABORATORY

A ModSAF RDR File Interface for the U.S. Army Research Laboratory Vulnerability Server

by Erik Greenwald

ARL-TN-231 **November 2004**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TN-231 **November 2004**

A ModSAF RDR File Interface for the U.S. Army Research Laboratory Vulnerability Server

Erik Greenwald
Survivability/Lethality Analysis Directorate, ARL

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2004	Final	October 2003–November 2003

4. TITLE AND SUBTITLE
A ModSAF RDR File Interface for the U.S. Army Research Laboratory Vulnerability Server

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Erik Greenwald

5d. PROJECT NUMBER
AH80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRD-ARL-SL-BE
Aberdeen Proving Ground, MD 21005-5068

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-231

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
With the move to High-Level Architecture and the Modeling Architecture for Technology Research EXperimentation (MATREX) suite, the U.S. Army Research Laboratory Vulnerability/Lethality Server required the ability to read the new database format for vulnerability probabilities. Support was provided through a component to read these database files, find the stored probabilities, and compute the probability packing suitable for selecting an end condition given a random number. The component implements a fast parameter-driven reentrant interface, as well as a compatibility interface based on the server's global values. The new ModSAF "reader" (RDR) file format parser and lookup component provides a high-performance and simple way to extract vulnerability probabilities from MATREX RDR files. While the software was written to support the server software package, the interface to the core RDR parser was made in a generic fashion. This report describes the design and implementation of the component, how to interface it, and a performance analysis as compared to the alternative individual unit of action table component.

15. SUBJECT TERMS
MATREX, distributed simulation, lethality, vulnerability, OneSAF, ModSAF

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Erik Greenwald
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	16	19b. TELEPHONE NUMBER (*Include area code*) 410-278-6255

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

War game simulations typically compute damage results by looking up probabilities of high-level damage conditions in a table of precomputed results. The U.S. Army Research Laboratory (ARL) Lethality/Vulnerability Server (the lethality server) provides a simulation with results based on several variables such as the range from the threat source to the target, angle between the target facing and threat path, whether the target is defiladed, etc.

With the move to ModSAF and the Modeling Architecture for Technology Research EXperimentation (MATREX) suite, the server required the ability to read the new database format for vulnerability probabilities. Support was provided via a component to read these database files, find the stored probabilities, and compute the probability packing suitable for selecting an end condition given a random number. The component implements a fast parameter-driven reentrant interface as well as a compatibility interface based on the server global values, such as range, defilation, dispersion, and angle. The purpose of this report is to describe the component written in support of the ARL lethality server for MATREX.

2. Formats

The lethality server uses database tables stored in files for the actual threat and weapon pairing. The new ModSAF/MATREX data set uses a new format to store these tables. The server required a new module to load the updated table format, but required the old format to still be accessible for tables that have not been updated. Minor optimizations were also performed on the system, such as rewriting the probability-packing algorithm to use less computationally expensive mathematical operations.

2.1 Individual Unit of Action (IUA) Tables

Old IUA files are simple flat file databases. The first line contains information such as how many ranges, dates, etc. The actual database begins after the header and each line is a unique tuple containing range, cover, dispersion, kill type, and the eight angle-bound probabilities, as depicted in figure 1 (from server source “sample.iua”). The first four digits compose the key for the table. The range requires the program to read further into the file than the actual entry to verify that another fitting range does not exist further down. The other three elements (cover, dispersion, and kill type) are simple matches.

```
7 9998 616  PKH 2040                                                    09/11/90
        0 1   1  1   0.276  0.284  0.666  0.843  0.867  0.870  0.952  0.534
        0 1   1  2   0.488  0.359  0.702  0.891  0.918  0.913  0.949  0.630
        0 1   1  3   0.488  0.359  0.702  0.891  0.918  0.925  0.953  0.630
        0 1   1  4   0.049  0.008  0.009  0.000  0.010  0.488  0.937  0.030
        0 1   2  1   0.239  0.349  0.547  0.723  0.713  0.765  0.774  0.472
```

Figure 1. Old IUA table format.

2.2 ModSAF Vulnerability Data Trees

The new format for storing IUA tables is the ModSAF "reader" (RDR) file format. The numbers are stored in a tree composed of lists instead of being a flat database of rows and columns. Much of the data is extracted from the lists structure. Lists are delimited with Lisp style parenthesis. Comments begin with a semicolon and are terminated at the end of the line. Figure 2 is a snippet from an RDR format file. The Backus Naur Form (BNF) grammar for this file format is listed in figure 3, and the lexicon in figure 4 (from MATREX v0.5 drop 2). These figures were composed from the "YACC"* and "Lex"† files, respectively.

```
;;
(IUA
(9999
 (
  (
   ( 0.000 0.000 0.200 0.200 0.200 0.500 0.500 0.000 )
   ( 0.100 0.100 0.300 0.300 0.300 0.500 0.500 0.000 )
   ( 0.100 0.100 0.300 0.300 0.300 0.500 0.500 0.000 )
   ( 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )
...
))))
```

Figure 2. ModSAF RDR list format.

The RDR parser walks the list of lists and generates a nearly duplicate arrangement in the computer's memory. This allows smaller segments of memory to be allocated by the software so "out of memory" errors are less likely than with large, table-style allocations that require big pieces of contiguous memory. The tree structure being stored in memory also allows O($log_2 n$) lookup speeds, as opposed to the typical O(n) lookup speed of a basic linear table scan.

* YACC - Yet Another Compiler Compiler, an LALR (Look-Ahead Left to Right) parser generator. Used to convert a CFG (Context Free Grammar) to a machine useable hierarchy.

† Lex - A lexical analyzer generator. Converts sequences of characters to machine useable atomic tokens.

```
<Table>       ::= "("<STR> <RangeSet>")"
<RangeSet>  ::= λ | <Ranges> <RangeSet>
<Ranges>     ::= "("<INT> <Showing> <Showing>")"
<Showing>   ::= "("<Kill> < Kill> < Kill>")" |
                    "("<R> <R> <R> <R> <R> <R> <R> <R> <R> <R> <R> <R> ")"
<R>            ::= "("<Kill> < Kill> < Kill> < Kill> ")"
<Kill>         ::= "("<FLT> < FLT > < FLT > < FLT > < FLT > < FLT > < FLT > < FLT > ")"
```

Figure 3. RDR grammar.

```
<INT>   ::=   [0-9][0-9]*
<STR>   ::=   [A-Za-z\_][A-Za-z\_0-9-]*
<FLT>   ::=   [0-9]*\.[0-9]+
;.*       ::=
```

Figure 4. RDR lexicon.

Simple integration into the existing server code base was necessary. A "glue" component was written to translate server-style function calls to RDR module recursive reentrant function calls. The RDR reader has a unified interface that allows the data itself to help dictate interpretation.

This allowed the "first call" function set in the glue component to be very thin, simple wrappers. One function was inserted into the lookup path to collect the global variables and adjust them to the formats and units used in the RDR files.

3. Application Program Interface

A new table reader component was added to access the new RDR format. The source to this new component resides in the "$VLS_HOME[†]/src/TblRdrs/rdr/" directory of ARL's lethality server project. Source to the new RDR component can be provided alone by request. The languages/tools used to generate this new component are C, Lex, and YACC. With a new unified interface, two sets of functions were created: native interface functions and compatibility interface functions. The interface between the new RDR component and the server is shown in figure 5.

```
void *rdr_load (FILE *in);
float *rdr_ground (void *table, int showing, float angle, float range, float dispersion);
void rdr_clear (void *tbl);
```

These functions are defined in section 3.1.

[†] "$VLS_HOME is a shell environment variable containing the "Lserver" directory.

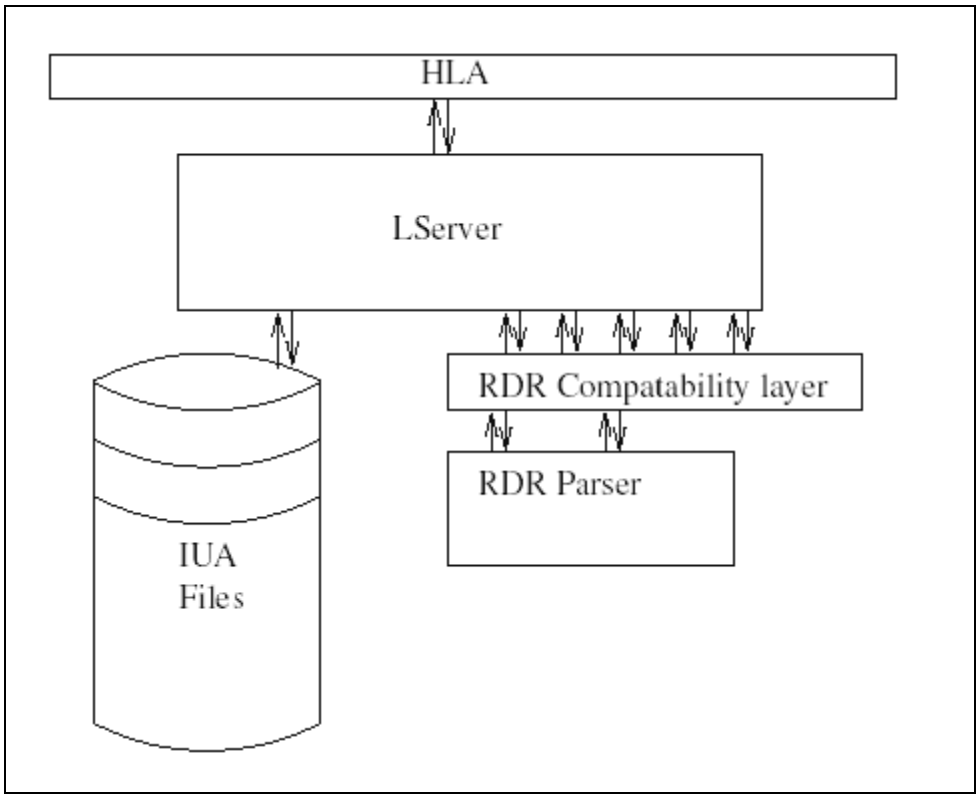

Figure 5. Component layout.

```
void * tblfmt_iua_multi_result (void *table);
void * tblfmt_iua_heat_rd (FILE * fp);
void * tblfmt_iua_he_rd (FILE * fp);
void * tblfmt_iua_ke_rd (FILE * fp);
void * tblfmt_iua_staff_rd (FILE * fp);
void * tblfmt_iua_heat_result (void *table);
void * tblfmt_iua_he_result (void *table);
void * tblfmt_iua_ke_result (void *table);
void * tblfmt_iua_staff_result (void *table);
```

These functions are defined in section 3.1.

3.1 Native Functions

The new RDR reader features a unified interface. All weapon and ground vehicle types are called in the same manner. Types were matched closely to the original IUA reader types to make integration with the server simple.

3.1.1 void *rdr_load (FILE *in);

When the server decides to load an RDR format IUA data table into memory, the rdr_load function is called to perform that task. The parameter is the file descriptor bound to the RDR file

(via fopen). The return value is a void pointer which will be passed to the reader functions. This function is reentrant by means of a semaphore which locks the critical area. If two rdr_load calls are executed simultaneously, they will be queued in sequence based on the operating systems semaphore management system.

3.1.2 float *rdr_ground (void *table, int showing, float angle, float range, float dispersion);

The actual lookup is performed with the *rdr_ground* function. The five parameters work to provide a closed deterministic result. The first parameter is the void pointer as returned by *rdr_load*. The showing parameter should be set to either 0 for partially defiladed or 1 for fully exposed. Currently, showing is assumed to be fully exposed in the compatibility layer. The range is the distance of the shot measured in meters. The dispersion is the distance of the hit measured in feet. The discrepancy in units is due to the definition of the RDR IUA format. The return value is an array of five 32-bit floats. The semantic of each is as follows: Mobility Kill (MKill), Firepower Kill (FKill), Mobility and Firepower Kill (MFKill), Catastrophic Kill (KKill), and no effect.

3.1.3 void rdr_clear (void *tbl);

This function is not implemented. This function was intended to destroy the RDR tree generated by the *rdr_load()* function. The decision to forego implementation was made due to deadline constraints and the existing code not leveraging collection.

3.2 Compatibility Functions

Several compatibility functions are required to interface the new RDR reader and lookup component without excessive modification to the existing server software. The server side expects variables to be passed in to the lookup component via global variables *(float VLP_impact[3], double VLP_range, and float VLP_ang_aspect)*, and a different function available for the table format pertaining to each of the weapons types: kinetic energy (KE), high explosive (HE), high explosive anti tank (HEAT), and smart target activated fire and forget (STAFF). To convert the globals to the parameter schema of the new reader and adjust the data to the right units is a single function which is capable of handling any of the four input types. The weapon-class functions call the new *rdr_ground* function with the variables extracted from the global environment. The purpose of these functions is to provide a “drop-in” compatibility with the server.

3.2.1 void * tblfmt_iua_multi_result (void *table);

The *tblfmt_iua_multi_result* function is the data translation and pass-through component to the compatibility layers query functionality. The global variables (range, defilation, angle, and dispersion) are extracted and converted into the format that rdr_ground is expecting, then *rdr_ground* is called, and the results are passed out. This function is not a compatibility function per se, but is a utility function to simplify the actual compatibility functions.

3.2.2 void * tblfmt_iua_heat_rd (FILE * fp);
void * tblfmt_iua_he_rd (FILE * fp);
void * tblfmt_iua_ke_rd (FILE * fp);
void * tblfmt_iua_staff_rd (FILE * fp);

These four compatibility functions merely call the *rdr_load* function. No data translation is done on the parameters or return values.

3.2.3 void * tblfmt_iua_heat_result (void *table);
void * tblfmt_iua_he_result (void *table);
void * tblfmt_iua_ke_result (void *table);
void * tblfmt_iua_staff_result (void *table);

The four compatibility functions previously shown call *tblfmt_iua_multi_result*. The input parameter is not modified. The output parameter is casted from float* to void*.

---

4. Performance

One concern raised about the new component was how quickly it performed queries on the data set. The original IUA query accessed the data by holding the table in arrays and using the query key components as indices to look directly at the data. The new RDR query traverses a tree. While the new method allows the program to operate on systems with less contiguous memory available, an overhead is incurred by the traversal.

A simple benchmark comparison between the old style table reader and the new RDR file reader was conducted to verify that the performance of the RDR component would not be much slower than the original table query. The test application loaded a table, set up the global variables, and then called the *tblfmt_iua_ke_result* function ten million times. Central processing unit (CPU) time was queried before and after the loop using the Unix *clock* function. The difference was converted to seconds, and the actual CPU time as well as the number of queries per second was displayed.

The new RDR component was tested using the compatibility layer to consider the overhead of addressing the global variables and passing them to the actual function. The total run time on a 2.0-GHz Intel Pentium 4 computer was 9.29 s. The yield was 1076426.269221 queries per second.

The IUA table version was tested in the same fashion on the same hardware. Completion of the benchmark set took 10.42 s. The resulting yield was 959692.891246 queries per second.

The numbers in the two paragraphs previously mentioned and figure 6 are of the first benchmark run for the two approaches. Both benchmarks were executed several times to verify the numbers were not erroneous. The programs were compiled with no optimization flags set to compare

	CPU time (s)	queries per second
rdr	9.29	1076426.269221
iua	10.42	959692.891246

Figure 6. RDR vs. IUA benchmark results.

algorithmic design instead of compiler capability. Both benchmark programs approximately doubled throughput when the compiler was instructed to use aggressive speed optimizations.* Performance of the new RDR component is slightly better than the original IUA-indexed lookup.

5. Conclusion

The new RDR file parser and lookup component provides a high performance and simple way to extract vulnerability probabilities from MATREX RDR files. While the software was written to support the ARL server software package, the interface to the core RDR parser was made in a generic fashion. Source code is available in the ARL server package or by request.

* "-O3-march=pentium4 -mcpu=pentium4" on GCC 3.2.

NO. OF COPIES ORGANIZATION

1 (PDF ONLY) DEFENSE TECHNICAL INFORMATION CTR
DTIC OCA
8725 JOHN J KINGMAN RD
STE 0944
FORT BELVOIR VA 22060-6218

1 US ARMY RSRCH DEV & ENGRG CMD
SYSTEMS OF SYSTEMS INTEGRATION
AMSRD SS T
6000 6TH ST STE 100
FORT BELVOIR VA 22060-5608

1 INST FOR ADVNCD TCHNLGY
THE UNIV OF TEXAS
AT AUSTIN
3925 W BRAKER LN STE 400
AUSTIN TX 78759-5316

1 US MILITARY ACADEMY
MATH SCI CTR EXCELLENCE
MADN MATH
THAYER HALL
WEST POINT NY 10996-1786

1 DIRECTOR
US ARMY RESEARCH LAB
IMNE AD IM DR
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CI OK TL
2800 POWDER MILL RD
ADELPHI MD 20783-1197

3 DIRECTOR
US ARMY RESEARCH LAB
AMSRD ARL CS IS T
2800 POWDER MILL RD
ADELPHI MD 20783-1197

NO. OF COPIES ORGANIZATION

ABERDEEN PROVING GROUND

1 DIR USARL
AMSRD ARL CI OK TP (BLDG 4600)

NO. OF
COPIES ORGANIZATION

1 OASD C3I
J BUCHHEISTER
RM 3D174
6000 DEFENSE PENTAGON
WASHINGTON DC 20301-6000

1 OUSD(AT)/S&T AIR WARFARE
R MUTZELBURG
RM 3E139
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(AT)/S&T LAND WARFARE
A VIILU
RM 3B1060
3090 DEFENSE PENTAGON
WASHINGTON DC 20310-3090

1 UNDER SECY OF THE ARMY
DUSA OR
RM 2E660
102 ARMY PENTAGON
WASHINGTON DC 20310-0102

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZP RM 2E661
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECH
SAAL ZS RM 3E448
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 DIRECTOR FORCE DEV
DAPR FDZ
RM 3A522
460 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 US ARMY TRADOC ANL CTR
ATRC W
A KEINTZ
WSMR NM 88002-5502

1 USARL
AMSRD ARL SL EA
R FLORES
WSMR NM 88002-5513

NO. OF
COPIES ORGANIZATION

1 USARL
AMSRD ARL SL EI
J NOWAK
FORT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST COM
CSTE DTC TT T
APG MD 21005-5055

1 US ARMY EVALUATION CTR
CSTE AEC SVE
R BOWEN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SVE S
R POLIMADEI
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CTR
CSTE AEC SV L
R LAUGHMAN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

14 DIR USARL
AMSRD ARL SL
J BEILFUSS
P DEITZ
AMSRD ARL SL B
J FRANZ
M PERRY
P TANENBAUM
AMSRD ARL SL BB
D BELY
D FARENWALD
S JUARASCIO
M RITONDO
AMSRD ARL SL BD
R GROTE
AMSRD ARL SL BE
L ROACH
AMSRD ARL SL E
M STARKS
AMSRD ARL SL EC
J FEENEY
E PANUSKA

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

8 DIR USARL
AMSRD ARL SL BE
J ANDERSON
R BOWERS
L BUTLER
E FIORAVANTE
E GREENWALD (4 CPS)