Z-Language 语言

语法规范白皮书

郑立松

Albert Zheng <lisong.zheng@gmail.com>

目　　录

一、Z-Language语言简介

Z-Language 语言是作者在 2001 年创业时为当时的创业产品 NetDefender-1 IDS 系统（黑客入侵检测系统）特别设计的一种**强类型嵌入式编程语言**，用于灵活编写和扩展 IDS 系统的入侵检测规则。2001 年彼时的 Lua 语言还很简单，且还是运行时动态解析的弱类型脚本语言，满足不了 IDS 系统对网络数据流内容进行高速检测的要求，因此作者决定自己设计一种健壮的、高速的强类型嵌入式语言，其具有如下特征：

- **Z-Language 在 2001 年时已经采用了字节码和虚拟机技术（类似 JVM），作者自己设计了一套字节码指令集和开发实现了字节码虚拟机**，因此能够实现真正的“一次编译，跨平台高速运行”。
- Z-Language **采用了“沙盒”安全保护技术**，因此可以保护 IDS 系统，避免错误编写的 Z-Language 程序摧毁 IDS 系统，或者消耗掉系统资源，例如：死循环检测、内存申请限制、函数递归层数限制等技术。
- Z-Language **提供了一个高效率的 C++嵌入式调用接口**，因此能够很容易地将 Z-Language 集成到公司的 NetDefender-1 IDS 系统中的任何网络协议处理层里；
- Z-Language **借鉴了 C 语言的语法结构，保留了 C 语言的灵活性和强大功能，但是去除了 C 语言中的不安全因素**，例如：指针、数组、显式内存申请、释放，等等。
- Z-Language **提供了多种丰富的基本数据类型和高级数据类型**，即 bool、byte、sdword、udword、sqword、uqword、double、string、rope、ipaddress、pport、table 等数据类型，**特别其中 ipaddress、pport 是专门设计的网络数据类型，rope 是一种高效率的串数据类型，而 table 是一种比数组更灵活、更安全的关联表数据类型**。
- Z-Language **支持多种高级编程语言才具有的特征**，例如：变量类型 cast、函数递归、for 和 while 循环控制、引用参数、引用变量、AND 和 OR 逻辑操作短路、C/C++风格的宏预处理，等等。
- Z-Language **支持将变量与 NetDefender-1 IDS 系统中正监测的任何 TCP Session 进行打结**，因此可以让用户在 Z-Language 程序中轻松地跟踪某个 TCP Session 的状态变化。
- Z-Language **提供了一套简洁的用于编写、扩展系统库函数的 C++接口**，在 Z-Language 代码里能够像原生代码那样调用新扩展的系统库函数。

二、数据类型

2.1、byte

无符号的 1Byte 类型，取值 0~255。

2.2、bool

无符号的 1Byte 类型，只能取值 true 或 false。

2.3、sdword

有符号的 4Byte 类型，取值-2147483648 ~ 2147483647。

2.4、udword

无符号的 4Byte 类型，取值 0 ~ 4294967295。

2.5、sqword

有符号的 8Byte 类型，取值-9223372036854775808 ~ 9223372036854775807。

2.6、uqword

无符号的 8Byte 类型，取值 0 ~ 18446744073709551615。

2.7、double

双精度浮点数。

2.8、string

string 类型被用来表示 8bit 清楚的非空结尾的 byte stream 或 byte array，长度最大为 uqword，可内含'\0'字符。

string 类型允许的操作符为"=="、"!="、"+"。

2.9、rope

rope 类型也被用来表示 8bit 清楚的非空结尾的 byte stream 或 byte array，长度最大为 uqword，可内含'\0'字符。

rope 类型允许的操作符为"=="、"!="、"+"。

rope 类型与 string 类型的区别是 rope 的"+"操作的速度极快，花费的时间始终是一个常量，因此 rope 比较适合用于大量的串进行"+"操作。

2.10、ipaddress

ipaddress 类型被用来表示 IP 主机地址或网络地址，ipaddress 类型的值格式为：

- 表示一个主机地址：{x.x.x.x}或{主机域名}
- 表示一个网络地址：{x.x.x.x/cidr-mask-bits-count}

如：{192.168.8.90}、{www.talent.com}、{192.168.8.0/24}。

ipaddress 类型允许的操作符为"=="、"!="。

2.11、pport

pport 类型被用来表示 internet protocol port，该类型占 2 个 UWORD 大小，其第一个 UWORD 为 port 类型（1=tcp、2=udp），第二个 UWORD 为 port 号。

pport 类型的值格式为 {80/tcp}、{21/udp}。

pport 类型允许的操作符为"=="、"!="、"<"、"<="、">"、">="，同时当两个 pport 类型变量的 port 类型不相同时，比较操作符总是返回 false。

2.12、table

table 类型是一种类似一维数组的 Hash table，其具有如下特性：

1) key 类型可以为任意简单类型，ZVM 总是先将 key 自动地转换为 string，然后再进行表操作；
2) 当采用"[]"操作符时，如果某个 key 关联的 element 并还不存在表中，则 ZVM 会自动立即按这个 key 插入一个具有缺省值的 element；

table 类型的语法如下：

```
table<SimpleType> TableName
```

例如：table<string> ts、table<udword> tudw

table 类型的变量初始化行的语法如下：

```
{ANY_TYPE_VALUE_AS_INDEX_VALUE = ELEMENT_VALUE, ...}
```

例如：table<string> ts = {1 = “abc”, “def” = “def”, {80/tcp} = “http”}

table 类型允许的操作符为"[]"，例如：ts[1]、ts[“def”]。

三、操作符

Z-Language 支持如下操作符（优先级从低到高排列）：

```
=
||
&&
|
&
^
==、!=
<、>、<=、>=
<<、>>
+、-、
*、/、%
++、--、
+（一元操作）、-（一元操作）
~、!
typecast
[]
function-call
```

四、变量

变量定义语法：

```
[extern | static] type VariableName
```

4.1、局部变量

在函数内部定义的变量为局部变量。例如：

```
string function1()
{
  sdword sdw = 8;
  table<string> t = { 1 = “1”, 2 = “2” };
  ……
}
```

4.2、局部引用变量

在函数内部定义的带有“&”修饰符的局部变量，局部引用变量必须在定义时就被初始化。例如：

```
extern table<string> & GetStringReference();

string function2()
{
  sdword sdw = 8;
  sdword & rsdw = sdw;
  table<string> & rt = GetStringReference();
  ……
}
```

4.3、全局变量

在函数外部定义的变量为全局变量。例如：

```
sdword Gsdw = 8;
table<string> Gt = { 1 = “1”, 2 = “2” };
……
```

4.4、static全局变量

在函数外部定义的带有“static”修饰符的全局变量，static 全局变量是一种特殊的全局变量，它只在自己的源程序作用域内可视的，因此可以避免与其它源程序内的同名全局变量冲突。例如：

```
static sdword SGsdw = 8;
static table<string> SGt = { 1 = “1”, 2 = “2” };
```

```
……
```

4.5、extern全局变量申明

以 extern 开头的全局变量申明。例如：

```
extern sdword Gsdw;
extern table<string> Gt;
……
```

五、函数

函数定义语法：

```
[static] return_type [&] FunctionName ‘(‘ P1_type [&] p1 ‘,’ ... ‘)’
‘{‘
  local_variables 定义;
  statements;
‘}’
```

参数的传递有两种方式：值参数、引用参数，“值参数”即是按值拷贝的方式传递参数，“引用参数”即是只传递参数的地址，“引用参数”带有“&”修饰符。

5.1、static函数定义

带有“static”修饰符的函数定义。

```
static string function1(sdword sdw, table<string> & rts)
{
  ……
}
```

5.2、extern函数申明

以 extern 开头的函数原型申明。

```
extern string function1(sdword sdw, table<string> & rts);
```

六、statement

6.1、if statement

if statement 的语法格式为：

```
if (expression)
{
  ......
}
else
{
  ......
}
```

6.2、while statement

while statement 的语法格式为：

```
while (expression)
{
  ......
  continue;  // 跳到 while 循环的开始，继续 while 循环
  ......
  break;  // 跳出 while 循环
  ......
}
```

6.3、for statement

for statement 的语法格式为：

```
for (expression1; expression2; expression3)
{
  ......
  continue;  // 跳到 for 循环判断的开始，继续 for 循环
  ......
  break;  // 跳出 for 循环
  ......
}
```

七、宏预处理

Z-Language 提供与 C/C++语言相同风格的宏预处理功能，例如：#include、#define、#if 等。

八、注释

Z-Language 采用 C 语言的“/* ..... */”和 C++语言的“// .....”风格的注释。