NAME : E.BLESSING CHARLES
NO : 2019103012
R BATCH
CRYPTOGRAPHY AND NETWORK SECURITY

FUZZING

AFL FUZZER

CPP CODE

A simple HTTP protocol request line parser and print the info of the requestline .

```
#include<iostream>
#include<bits/stdc++.h>
#include<fstream>


using namespace std ;

void parse_file(char *filename, vector<string> &lines){
   string line;

   ifstream input_file(filename);
   if (!input_file.is_open()) {
       cerr << "Could not open the file - '"
            << filename << "'" << endl;
       exit(-1);
   }

   while (getline(input_file, line)){
       lines.push_back(line);
   }
   input_file.close();
}

bool parse_http(vector<string> &lines){
```

```
if(lines.size() > 1){
    cout << "Request Line only accepted" ;
    exit(-1);
}

/* A simple request line parse for http protocol*/

unordered_map<string , string> headers ;
unordered_set<string> methods = {"GET" , "POST" , "DELETE" , "PUT"} ;
unordered_set<string> httpversions = {"HTTP/0.9" , "HTTP/1.0" , "HTTP/1.1"};

// parse the request line in http
// eg :    GET / HTTP/1.1
istringstream ss(lines[0]);
string word ;

cout << "[+] Parsing Requestline : " << lines[0] << endl ;
string method , path , httpname ;

int idx = 0 ;
vector<string> req_line ;
while(ss >> word){
    req_line.push_back(word) ;
}
if(req_line.size() > 3){
    cout << "invalid requestline " << endl ;
    return false ;
}

for(auto &word : req_line){
    if(idx == 0){
        if(methods.find(word) == methods.end()){
            // unknown method
            cout << "unknown method" << endl ;
            return false ;
        }
        method = word ;
    }
    else if(idx == 1){
```

```
            if(word[0] != '/'){
                cout << "invalid path";
                return false ;
            }
            path = word ;
        }
        else if(idx == 2){
            if(httpversions.find(word) == httpversions.end()){
                // unknown method
                cout << "unknown method" ;
                return false ;
            }
            httpname = word ;
        }
        else{
            // invalid state
            cout << "invalid request line" ;
            return false ;
        }
        idx++ ;
    }

    cout << "METHOD : " << method << endl ;
    cout << "path : " << path << endl ;
    cout << "httpname : " << httpname << endl ;

    return true ;
}

int main(int argc , char **argv){

    if(argc < 2){
        cout << "enter http filename" ;
        return -1 ;
    }

    vector<string> lines ;

    parse_file(argv[1],lines);
    int *ptr  ;
```

```
    ptr = NULL ;

    if(parse_http(lines)){
        cout << "successfully parsed http" ;
        // lets crash by deferencing a null pointer
        *ptr = 5 ;
    }
    else{
        cout << "failed to parse" ;
    }
}
```

The input should match the rfc abnf grammar for the requestline otherwise the code will exit . The afl fuzzer will starts with a seed value and starts fuzzing the code what possible ways we can crash the binary by covering all the paths in the code . AFL fuzzer is a feedback based fuzzer , by analyzing how the given input , change the output of the program and use it as a seed for the next input to the program . AFL fuzzer do both code coverage and path coverage in a given binary , so we can completely analyze the binary for various mutated inputs . Code Coverage refers to the amount of code that was triggered by a particular test case. Path Coverage refers to the number of potential sequence of code statements (or paths) that were triggered by a test case.

Seed Value :

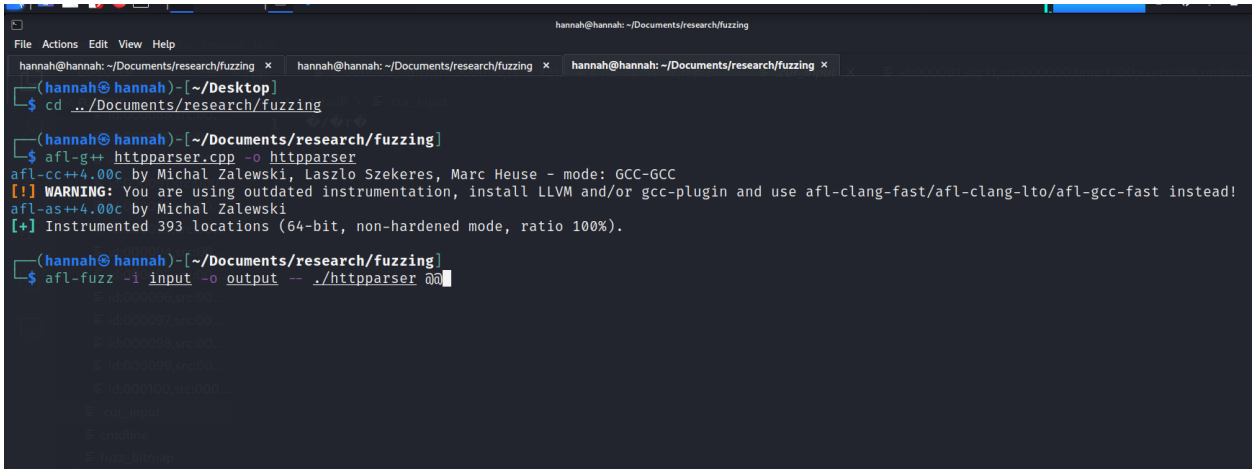
For the initial seed we are going to give HTE / HPPT/5.4 as the input .

VALID HTTP Requestline Syntax according to RFC

Request-line = method path http-name
Method = “GET” , “POST” , “PUT” , “DELETE”
Path = “/”<ascii-char>
Http-name = HTTP/ DIGIT “.” DIGIT

The afl fuzzer will start from the initial given seed value and it will try to match the defined grammar or any crashes occured are also reported .

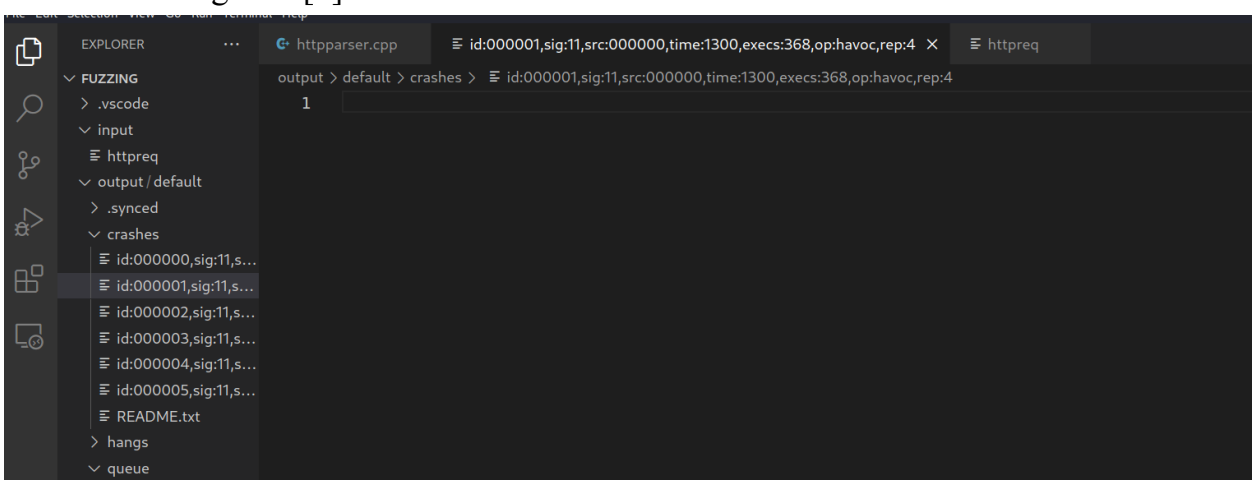
AFL RUNNING



CRASHES REPORTED

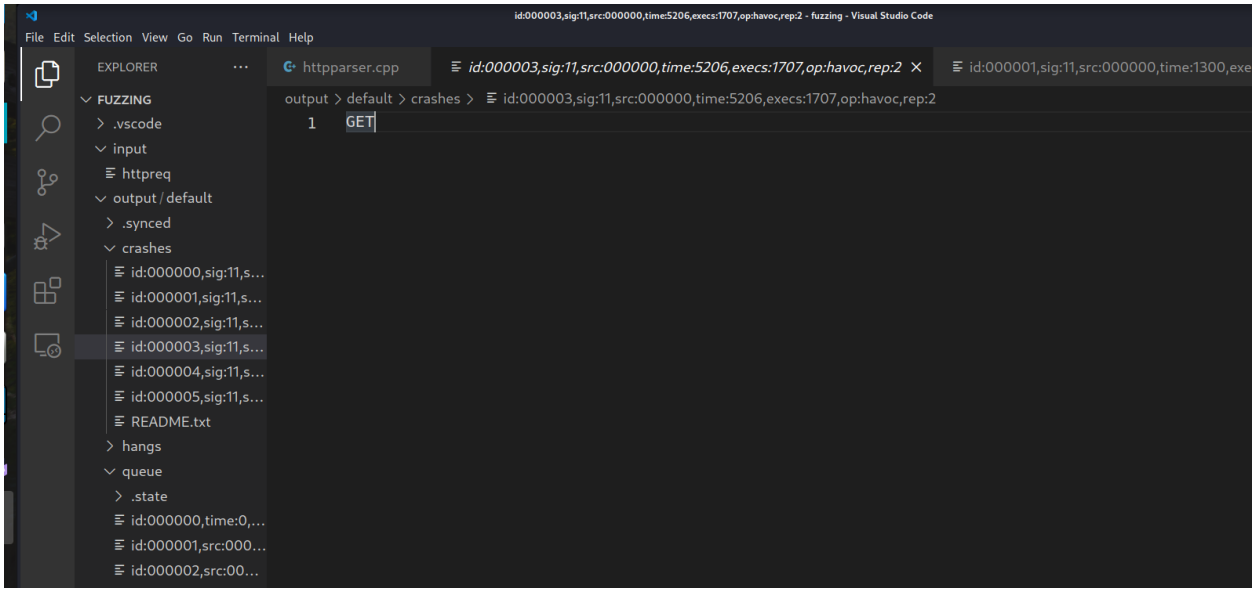
All crashed for a given input will be in the output/crashes folder

1. Empty input , as in parse_http function there is no check on minimum size of lines vector so by accessing lines[0] will crash .

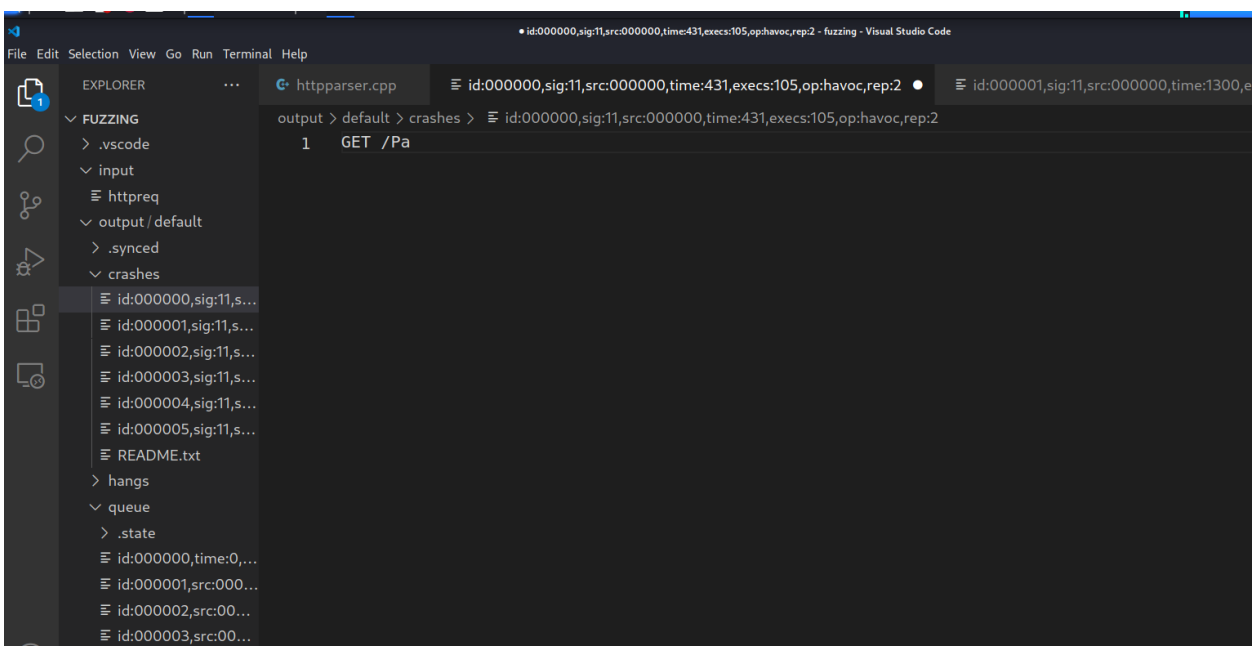


2. There should be three strings required for the requestline as there is no check for it in the code , do deferencing will result in crash .

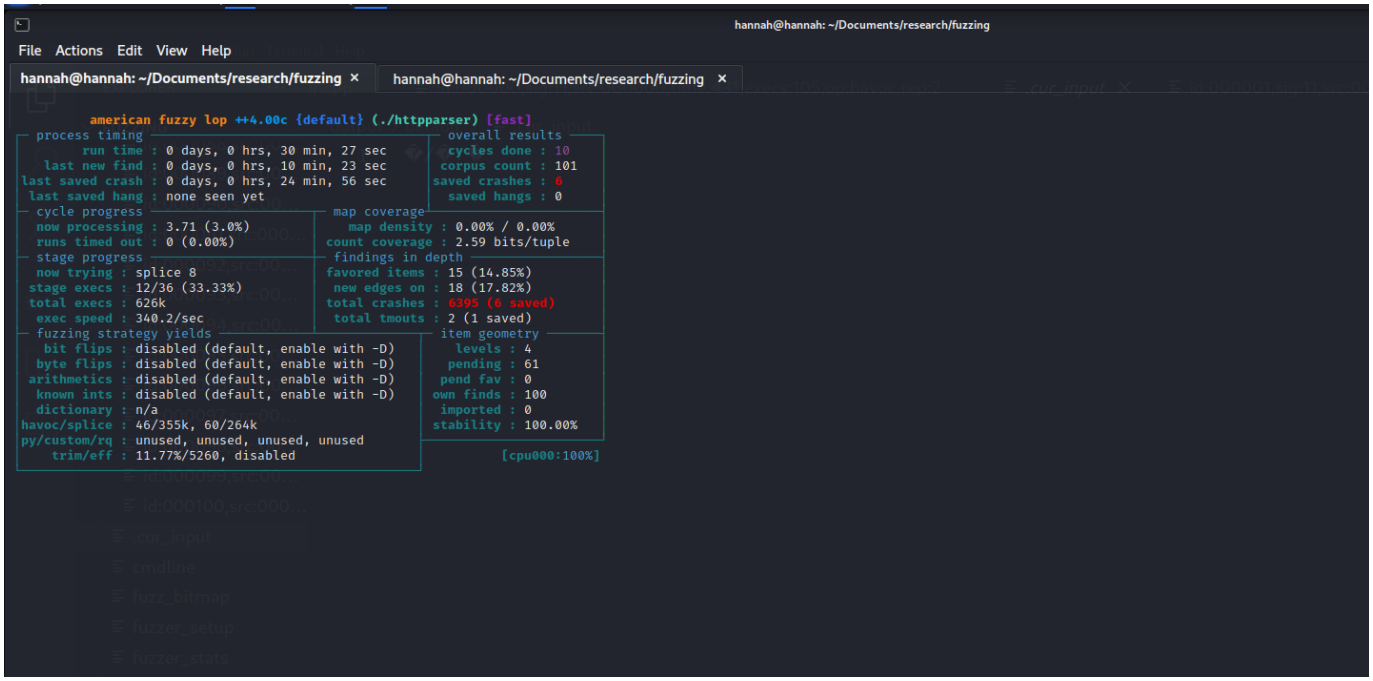
Request only with http method result in crash



Request without httpname will result in a crash



AFL FUZZING SCREEN



LIBFUZZER

In Libfuzzer , instead of main function we need to provide int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) function , where the libfuzzer start fuzzing by giving input in *data character array . LibFuzzer is linked with the library under test, and feeds fuzzed inputs to the library , the fuzzer then tracks which areas of the code are reached, and generates mutations on the corpus of input data in order to maximize the code coverage.

CPP Code

```
#include<iostream>
#include<bits/stdc++.h>
#include<fstream>


using namespace std ;

string convertToString(const char* a, int size)
```

```
{
   int i;
   string s = "";
   for (i = 0; i < size; i++) {
       s = s + a[i];
   }
   return s;
}


extern "C" int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {

   string input = convertToString((char *)Data  , Size);

   /* A simple request line parse for http protocol*/

   unordered_map<string , string> headers ;
   unordered_set<string> methods = {"GET" , "POST" , "DELETE" , "PUT"} ;
   unordered_set<string> httpversions = {"HTTP/0.9" , "HTTP/1.0" ,
"HTTP/1.1"};

   // parse the request line in http
   // eg :    GET / HTTP/1.1
   istringstream ss(input);
   string word ;

   string method , path , httpname ;

   int idx = 0 ;
   vector<string> req_line ;
   while(ss >> word){
       req_line.push_back(word) ;
   }
   if(req_line.size() > 3){
```

```
        return -1 ;
    }

    for(auto &word : req_line){
        if(idx == 0){
            if(methods.find(word) == methods.end()){
                // unknown method
                return -1 ;
            }
            method = word ;
        }
        else if(idx == 1){
            if(word[0] != '/'){
                return -1 ;
            }
            path = word ;
        }
        else if(idx == 2){
            if(httpversions.find(word) == httpversions.end()){
                // unknown method
                return -1 ;
            }
            httpname = word ;
        }
        else{
            // invalid state
            return -1 ;
        }
        idx++ ;
    }
    return 0 ;
}
```

Compiling the code with clang++

```
th3h04x@ThomasThecaT:~/Documents/fuzzing
./http-parser input        th3h04x@ThomasThecaT:~/Documents/.dsa/tree
th3h04x@ThomasThecaT:~/Documents/fuzzing 166x43
➜ fuzzing clang++ -g -fsanitize=address,fuzzer http-parser.cpp -o http-parser
```

Running the fuzzer

```
➜ fuzzing ./http-parser input
INFO: Seed: 1345243893
INFO: Loaded 1 modules   (421 inline 8-bit counters): 421 [0x5bb290, 0x5bb435),
INFO: Loaded 1 PC tables (421 PCs): 421 [0x57aac0,0x57c510),
INFO:        1 files found in input
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: seed corpus: files: 1 min: 14b max: 14b total: 14b rss: 27Mb
#2      INITED cov: 273 ft: 274 corp: 1/14b exec/s: 0 rss: 28Mb
#5      NEW    cov: 273 ft: 283 corp: 2/28b lim: 14 exec/s: 0 rss: 28Mb L: 14/14 MS: 3 ShuffleBytes-ChangeBit-ChangeBit-
#6      NEW    cov: 274 ft: 292 corp: 3/37b lim: 14 exec/s: 0 rss: 28Mb L: 9/14 MS: 1 CrossOver-
#7      NEW    cov: 274 ft: 330 corp: 4/51b lim: 14 exec/s: 0 rss: 28Mb L: 14/14 MS: 1 ChangeBit-
#16     REDUCE cov: 274 ft: 330 corp: 4/48b lim: 14 exec/s: 0 rss: 28Mb L: 11/14 MS: 4 ChangeBinInt-ChangeBinInt-ChangeByte-CrossOver-
#18     REDUCE cov: 276 ft: 352 corp: 5/62b lim: 14 exec/s: 0 rss: 28Mb L: 14/14 MS: 2 ChangeBinInt-CrossOver-
#20     NEW    cov: 276 ft: 390 corp: 6/76b lim: 14 exec/s: 0 rss: 28Mb L: 14/14 MS: 2 ChangeByte-CrossOver-
#28     NEW    cov: 276 ft: 393 corp: 7/80b lim: 14 exec/s: 0 rss: 28Mb L: 4/14 MS: 3 ChangeByte-CMP-CrossOver- DE: "\x01\x00\x00\x00"-
#32     NEW    cov: 276 ft: 438 corp: 8/86b lim: 14 exec/s: 0 rss: 28Mb L: 6/14 MS: 4 InsertByte-ChangeByte-InsertByte-PersAutoDict- DE: "\x01\x00\x00\x00"-
#38     NEW    cov: 276 ft: 439 corp: 9/100b lim: 14 exec/s: 0 rss: 28Mb L: 14/14 MS: 1 CopyPart-
#44     NEW    cov: 276 ft: 442 corp: 10/103b lim: 14 exec/s: 0 rss: 28Mb L: 3/14 MS: 1 EraseBytes-
#67     NEW    cov: 276 ft: 445 corp: 11/104b lim: 14 exec/s: 0 rss: 28Mb L: 1/14 MS: 3 ChangeBit-CopyPart-CrossOver-
#88     REDUCE cov: 276 ft: 445 corp: 11/103b lim: 14 exec/s: 0 rss: 28Mb L: 5/14 MS: 1 EraseBytes-
#201    NEW    cov: 276 ft: 448 corp: 12/105b lim: 14 exec/s: 0 rss: 28Mb L: 2/14 MS: 3 EraseBytes-ShuffleBytes-EraseBytes-
#206    REDUCE cov: 276 ft: 448 corp: 12/104b lim: 14 exec/s: 0 rss: 28Mb L: 8/14 MS: 5 ChangeBit-PersAutoDict-InsertByte-InsertRepeatedBytes-EraseBytes- DE: "\
0\x00\x00"-
#240    REDUCE cov: 276 ft: 448 corp: 12/103b lim: 14 exec/s: 0 rss: 28Mb L: 4/14 MS: 4 ChangeBit-ShuffleBytes-CopyPart-EraseBytes-
#262    REDUCE cov: 276 ft: 448 corp: 12/100b lim: 14 exec/s: 0 rss: 28Mb L: 8/14 MS: 2 EraseBytes-InsertByte-
#284    NEW    cov: 278 ft: 459 corp: 13/101b lim: 14 exec/s: 0 rss: 28Mb L: 1/14 MS: 2 ChangeByte-EraseBytes-
#345    NEW    cov: 278 ft: 461 corp: 14/115b lim: 14 exec/s: 0 rss: 29Mb L: 14/14 MS: 1 CopyPart-
#461    REDUCE cov: 278 ft: 461 corp: 14/114b lim: 14 exec/s: 0 rss: 29Mb L: 3/14 MS: 1 EraseBytes-
#503    REDUCE cov: 278 ft: 461 corp: 14/113b lim: 14 exec/s: 0 rss: 29Mb L: 2/14 MS: 2 InsertByte-EraseBytes-
#504    REDUCE cov: 278 ft: 461 corp: 14/109b lim: 14 exec/s: 0 rss: 29Mb L: 4/14 MS: 1 EraseBytes-
#571    REDUCE cov: 278 ft: 461 corp: 14/108b lim: 14 exec/s: 0 rss: 29Mb L: 1/14 MS: 2 ChangeByte-EraseBytes-
#886    NEW    cov: 278 ft: 464 corp: 15/125b lim: 17 exec/s: 0 rss: 29Mb L: 17/17 MS: 5 ChangeBit-InsertRepeatedBytes-ChangeByte-InsertRepeatedBytes-CopyPart-
#999    REDUCE cov: 278 ft: 464 corp: 15/124b lim: 17 exec/s: 0 rss: 29Mb L: 16/16 MS: 3 ChangeBit-CrossOver-InsertRepeatedBytes-
#1022   REDUCE cov: 278 ft: 472 corp: 16/141b lim: 17 exec/s: 0 rss: 29Mb L: 17/17 MS: 3 ChangeBinInt-CopyPart-CrossOver-
#1203   REDUCE cov: 278 ft: 472 corp: 16/137b lim: 17 exec/s: 0 rss: 30Mb L: 4/17 MS: 1 EraseBytes-
#1316   REDUCE cov: 278 ft: 472 corp: 16/131b lim: 17 exec/s: 0 rss: 30Mb L: 8/17 MS: 3 CopyPart-ChangeBit-EraseBytes-
#1432   REDUCE cov: 278 ft: 472 corp: 16/130b lim: 17 exec/s: 0 rss: 30Mb L: 13/17 MS: 1 EraseBytes-
#1668   REDUCE cov: 278 ft: 472 corp: 16/128b lim: 17 exec/s: 0 rss: 30Mb L: 2/17 MS: 1 EraseBytes-
#1829   REDUCE cov: 278 ft: 472 corp: 16/127b lim: 17 exec/s: 0 rss: 30Mb L: 13/17 MS: 1 CrossOver-
#2041   REDUCE cov: 278 ft: 472 corp: 16/126b lim: 17 exec/s: 0 rss: 31Mb L: 16/16 MS: 2 EraseBytes-ShuffleBytes-
#2488   REDUCE cov: 278 ft: 472 corp: 16/125b lim: 21 exec/s: 0 rss: 31Mb L: 12/16 MS: 2 CMP-CrossOver- DE: "\xa0_\x16u\xa8\x0c\xc6\x01"-
#2607   NEW    cov: 278 ft: 481 corp: 17/146b lim: 21 exec/s: 0 rss: 31Mb L: 21/21 MS: 4 InsertByte-ShuffleBytes-CrossOver-CrossOver-
#3004   REDUCE cov: 278 ft: 481 corp: 17/145b lim: 21 exec/s: 0 rss: 32Mb L: 3/21 MS: 2 ShuffleBytes-EraseBytes-
```