딥러닝으로 취약점을 찾아보자

VulnViz: 취약점 분석의 시각적 접근

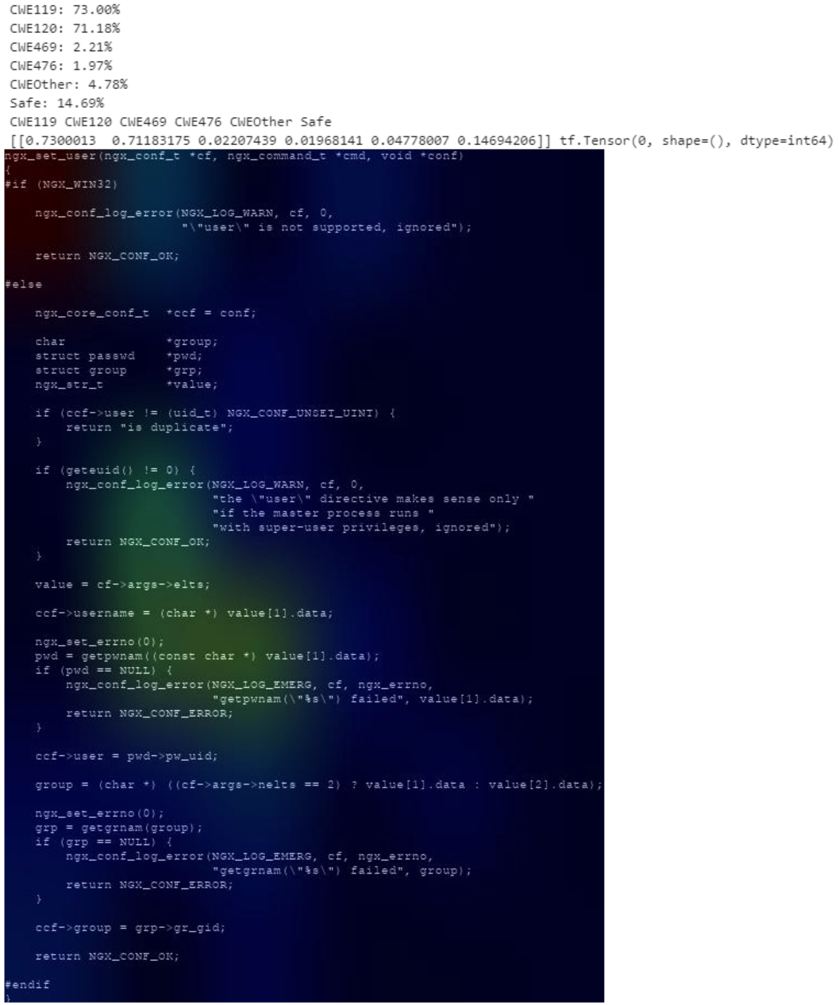
발표자소개

김성우 cd80@HypwnLab http://hypwnlab.com

cd80@DeepSec

관심분야: 취약점 분석에 기계학습 응용

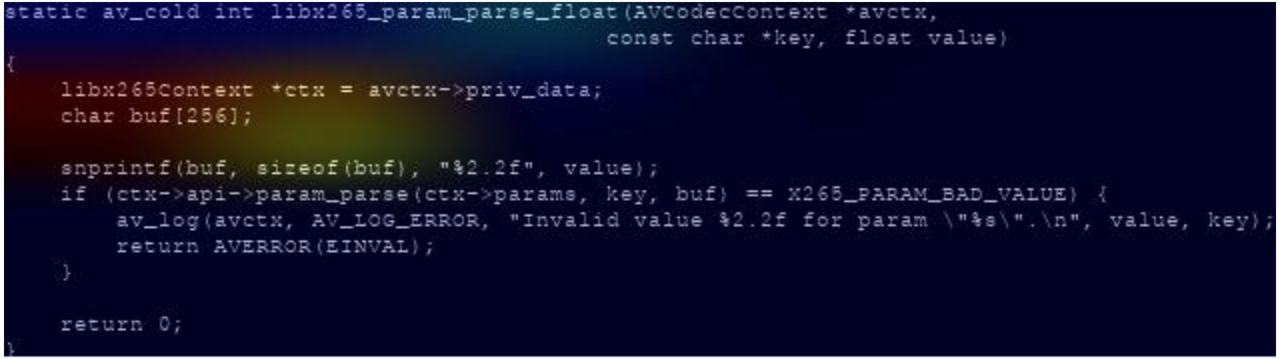
하고 싶었던 것



```
CWE119: 73.00%
CWE120: 71.18%
CWE469: 2.21%
CWE476: 1.97%
CWEOther: 4.78%
Safe: 14.69%
CWE119 CWE120 CWE469 CWE476 CWEOther Safe
[[0.7300013  0.71183175 0.02207439 0.01968141 0.04778007 0.14694206]] tf.Tensor(0, shape=(), dtype=int64)
```

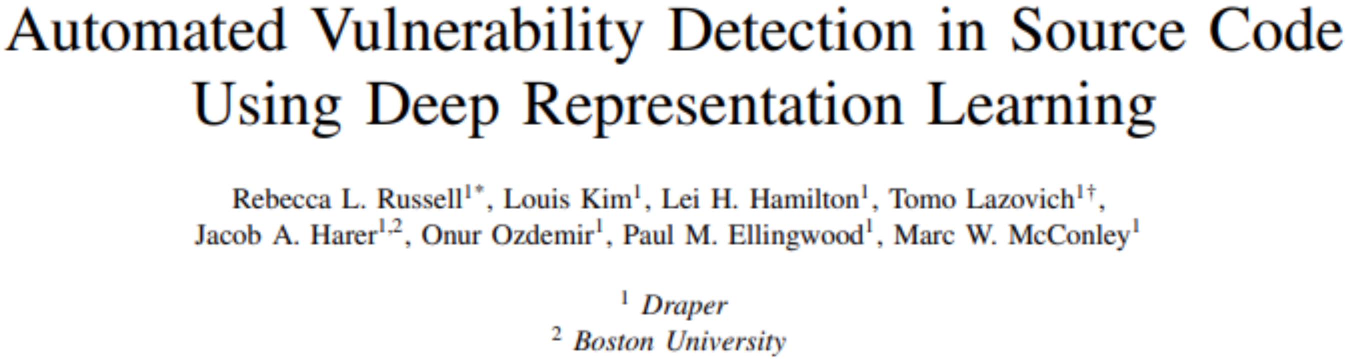
```
ngx_set_user(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
#if (NGX_WIN32)

    ngx_conf_log_error(NGX_LOG_WARN, cf, 0,
                       "\"user\" is not supported, ignored");

    return NGX_CONF_OK;

#else

    ngx_core_conf_t  *ccf = conf;

    char             *group;
    struct passwd    *pwd;
    struct group     *grp;
    ngx_str_t        *value;

    if (ccf->user != (uid_t) NGX_CONF_UNSET_UINT) {
        return "is duplicate";
    }

    if (geteuid() != 0) {
        ngx_conf_log_error(NGX_LOG_WARN, cf, 0,
                           "the \"user\" directive makes sense only "
                           "if the master process runs "
                           "with super-user privileges, ignored");
        return NGX_CONF_OK;
    }

    value = cf->args->elts;

    ccf->username = (char *) value[1].data;

    ngx_set_errno(0);
    pwd = getpwnam((const char *) value[1].data);
    if (pwd == NULL) {
        ngx_conf_log_error(NGX_LOG_EMERG, cf, ngx_errno,
                           "getpwnam(\"%s\") failed", value[1].data);
        return NGX_CONF_ERROR;
    }

    ccf->user = pwd->pw_uid;

    group = (char *) ((cf->args->nelts == 2) ? value[1].data : value[2].data);

    ngx_set_errno(0);
    grp = getgrnam(group);
    if (grp == NULL) {
        ngx_conf_log_error(NGX_LOG_EMERG, cf, ngx_errno,
                           "getgrnam(\"%s\") failed", group);
        return NGX_CONF_ERROR;
    }

    ccf->group = grp->gr_gid;

    return NGX_CONF_OK;

#endif
}
```

하고 싶었던 것



```
static av_cold int libx265_param_parse_float(AVCodecContext *avctx,
                                             const char *key, float value)
{
    libx265Context *ctx = avctx->priv_data;
    char buf[256];

    snprintf(buf, sizeof(buf), "%2.2f", value);
    if (ctx->api->param_parse(ctx->params, key, buf) == X265_PARAM_BAD_VALUE) {
        av_log(avctx, AV_LOG_ERROR, "Invalid value %2.2f for param \"%s\".\n", value, key);
        return AVERROR(EINVAL);
    }

    return 0;
}
```

Motivation

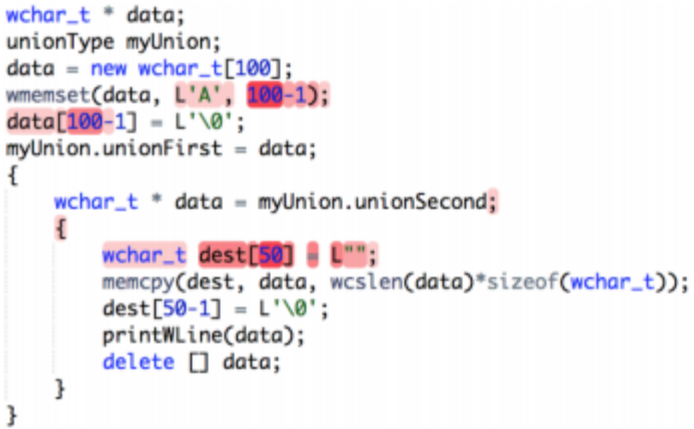


Automated Vulnerability Detection in Source Code
Using Deep Representation Learning

Rebecca L. Russell[1*], Louis Kim[1], Lei H. Hamilton[1], Tomo Lazovich[1†],
Jacob A. Harer[1,2], Onur Ozdemir[1], Paul M. Ellingwood[1], Marc W. McConley[1]

[1] *Draper*
[2] *Boston University*

Motivation



```
wchar_t * data;
unionType myUnion;
data = new wchar_t[100];
wmemset(data, L'A', 100-1);
data[100-1] = L'\0';
myUnion.unionFirst = data;
{
    wchar_t * data = myUnion.unionSecond;
    {
        wchar_t dest[50] = L"";
        memcpy(dest, data, wcslen(data)*sizeof(wchar_t));
        dest[50-1] = L'\0';
        printWLine(data);
        delete [] data;
    }
}
```

Dataset

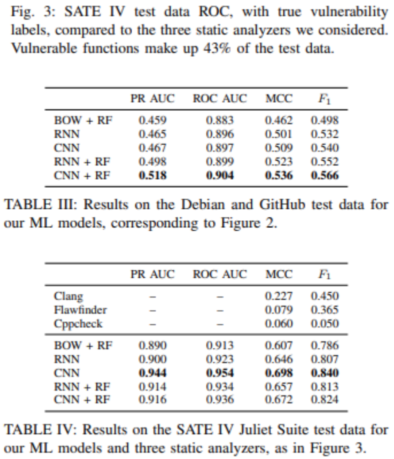


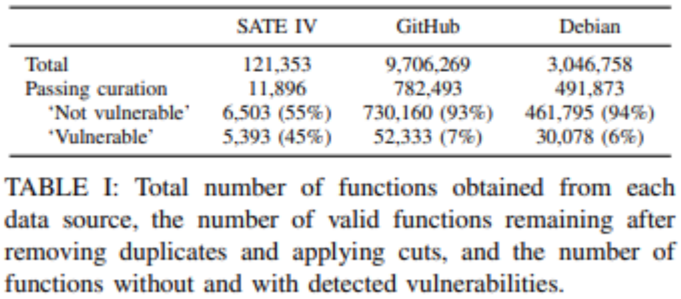
Fig. 3: SATE IV test data ROC, with true vulnerability labels, compared to the three static analyzers we considered. Vulnerable functions make up 43% of the test data.

	PR AUC	ROC AUC	MCC	F_1
BOW + RF	0.459	0.883	0.462	0.498
RNN	0.465	0.896	0.501	0.532
CNN	0.467	0.897	0.509	0.540
RNN + RF	0.498	0.899	0.523	0.552
CNN + RF	**0.518**	**0.904**	**0.536**	**0.566**

TABLE III: Results on the Debian and GitHub test data for our ML models, corresponding to Figure 2.

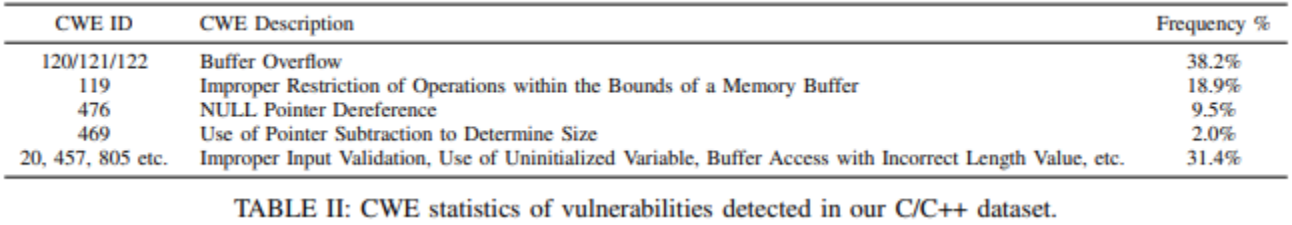
	PR AUC	ROC AUC	MCC	F_1
Clang	–	–	0.227	0.450
Flawfinder	–	–	0.079	0.365
Cppcheck	–	–	0.060	0.050
BOW + RF	0.890	0.913	0.607	0.786
RNN	0.900	0.923	0.646	0.807
CNN	**0.944**	**0.954**	**0.698**	**0.840**
RNN + RF	0.914	0.934	0.657	0.813
CNN + RF	0.916	0.936	0.672	0.824

TABLE IV: Results on the SATE IV Juliet Suite test data for our ML models and three static analyzers, as in Figure 3.



	SATE IV	GitHub	Debian
Total	121,353	9,706,269	3,046,758
Passing curation	11,896	782,493	491,873
'Not vulnerable'	6,503 (55%)	730,160 (93%)	461,795 (94%)
'Vulnerable'	5,393 (45%)	52,333 (7%)	30,078 (6%)

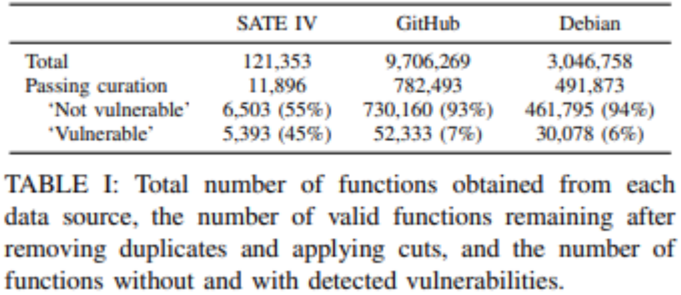
TABLE I: Total number of functions obtained from each data source, the number of valid functions remaining after removing duplicates and applying cuts, and the number of functions without and with detected vulnerabilities.



CWE ID	CWE Description	Frequency %
120/121/122	Buffer Overflow	38.2%
119	Improper Restriction of Operations within the Bounds of a Memory Buffer	18.9%
476	NULL Pointer Dereference	9.5%
469	Use of Pointer Subtraction to Determine Size	2.0%
20, 457, 805 etc.	Improper Input Validation, Use of Uninitialized Variable, Buffer Access with Incorrect Length Value, etc.	31.4%

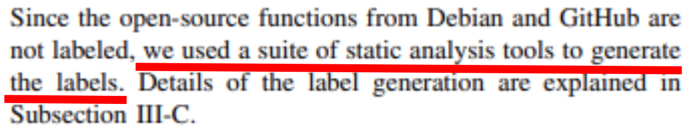
TABLE II: CWE statistics of vulnerabilities detected in our C/C++ dataset.

Dataset

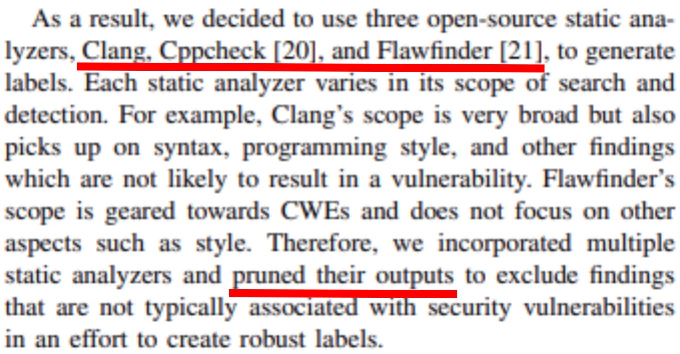


	SATE IV	GitHub	Debian
Total	121,353	9,706,269	3,046,758
Passing curation	11,896	782,493	491,873
'Not vulnerable'	6,503 (55%)	730,160 (93%)	461,795 (94%)
'Vulnerable'	5,393 (45%)	52,333 (7%)	30,078 (6%)

TABLE I: Total number of functions obtained from each data source, the number of valid functions remaining after removing duplicates and applying cuts, and the number of functions without and with detected vulnerabilities.

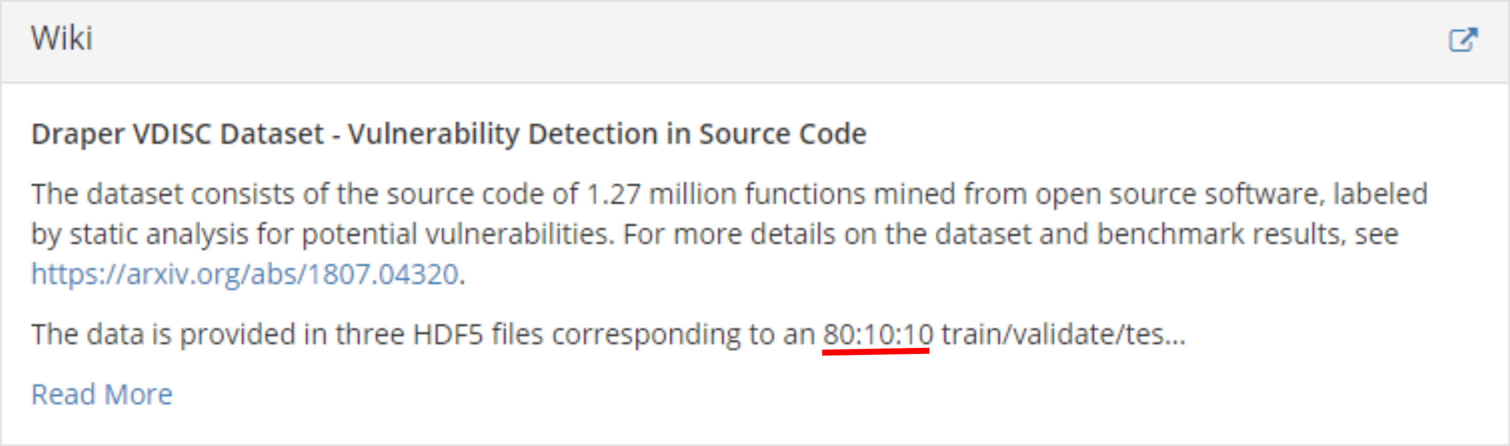


Since the open-source functions from Debian and GitHub are not labeled, we used a suite of static analysis tools to generate the labels. Details of the label generation are explained in Subsection III-C.



As a result, we decided to use three open-source static analyzers, Clang, Cppcheck [20], and Flawfinder [21], to generate labels. Each static analyzer varies in its scope of search and detection. For example, Clang's scope is very broad but also picks up on syntax, programming style, and other findings which are not likely to result in a vulnerability. Flawfinder's scope is geared towards CWEs and does not focus on other aspects such as style. Therefore, we incorporated multiple static analyzers and pruned their outputs to exclude findings that are not typically associated with security vulnerabilities in an effort to create robust labels.

Dataset



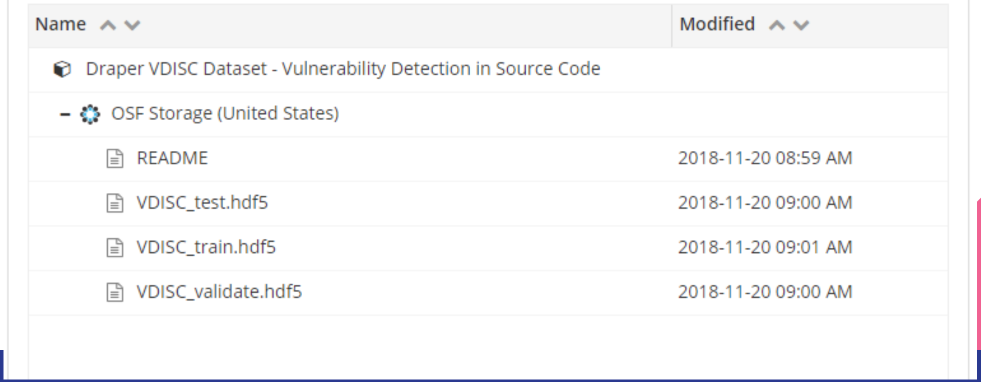
Wiki

Draper VDISC Dataset - Vulnerability Detection in Source Code

The dataset consists of the source code of 1.27 million functions mined from open source software, labeled by static analysis for potential vulnerabilities. For more details on the dataset and benchmark results, see https://arxiv.org/abs/1807.04320.

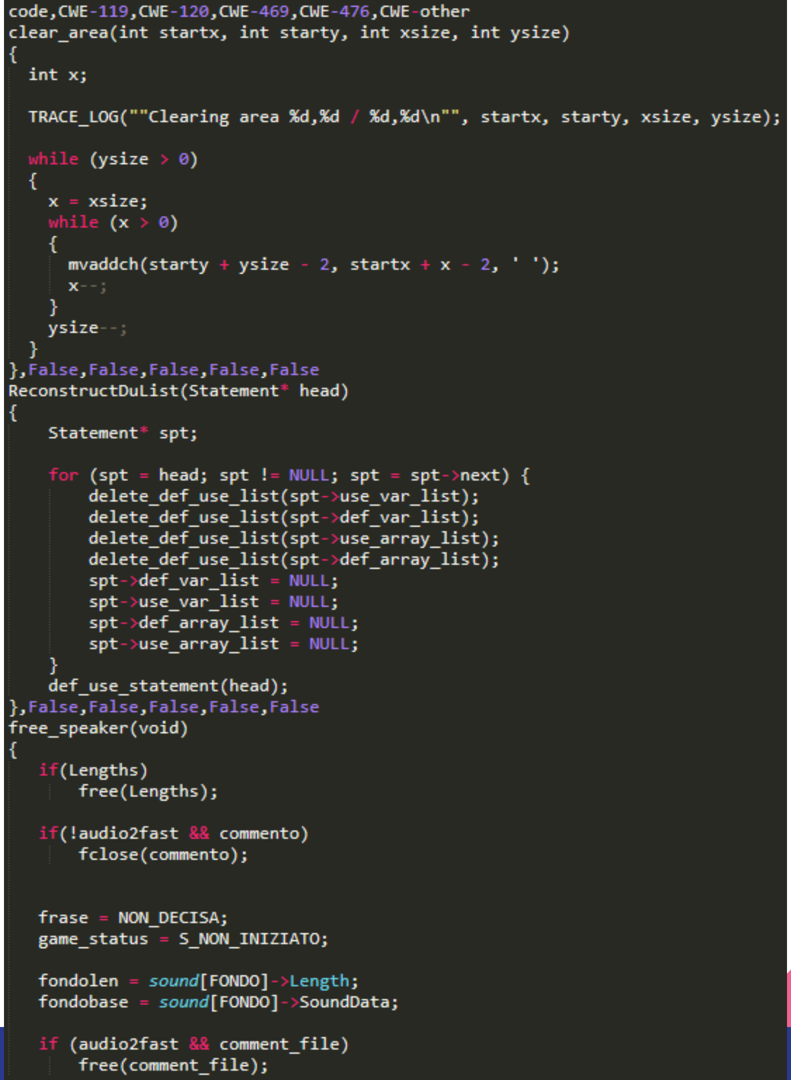
The data is provided in three HDF5 files corresponding to an 80:10:10 train/validate/tes...

Read More



Name	Modified
Draper VDISC Dataset - Vulnerability Detection in Source Code	
– OSF Storage (United States)	
README	2018-11-20 08:59 AM
VDISC_test.hdf5	2018-11-20 09:00 AM
VDISC_train.hdf5	2018-11-20 09:01 AM
VDISC_validate.hdf5	2018-11-20 09:00 AM

Dataset



```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
 int x;

 TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

 while (ysize > 0)
 {
  x = xsize;
  while (x > 0)
  {
   mvaddch(starty + ysize - 2, startx + x - 2, ' ');
   x--;
  }
  ysize--;
 }
},False,False,False,False,False
ReconstructDuList(Statement* head)
{
  Statement* spt;

  for (spt = head; spt != NULL; spt = spt->next) {
    delete_def_use_list(spt->use_var_list);
    delete_def_use_list(spt->def_var_list);
    delete_def_use_list(spt->use_array_list);
    delete_def_use_list(spt->def_array_list);
    spt->def_var_list = NULL;
    spt->use_var_list = NULL;
    spt->def_array_list = NULL;
    spt->use_array_list = NULL;
  }
  def_use_statement(head);
},False,False,False,False,False
free_speaker(void)
{
   if(Lengths)
     free(Lengths);

   if(!audio2fast && commento)
     fclose(commento);


   frase = NON_DECISA;
   game_status = S_NON_INIZIATO;

   fondolen = sound[FONDO]->Length;
   fondobase = sound[FONDO]->SoundData;

   if (audio2fast && comment_file)
     free(comment_file);
```

Dataset

multi-label

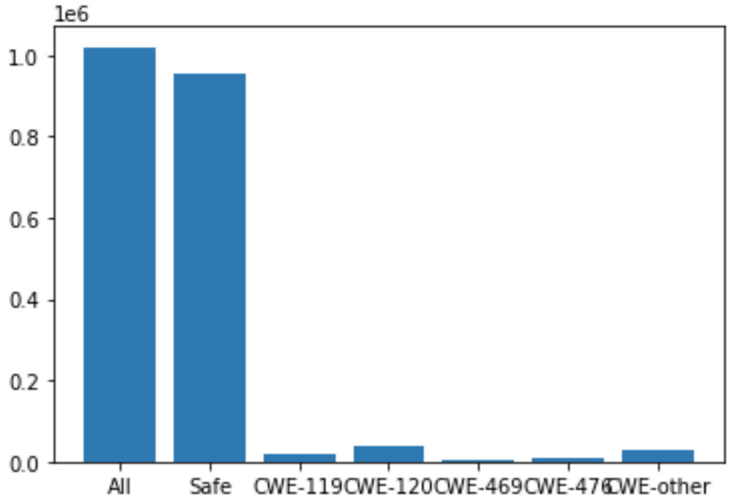
CWE119 CWE120 CWE469 CWE476 CWEOther

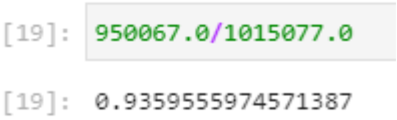
binary-class

True False

imbalanced

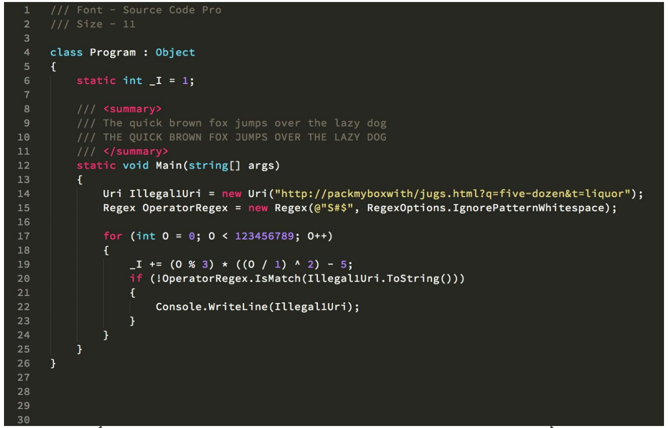
```
[1019471, 953567, 19286, 38019, 2095, 9694, 27959]
```



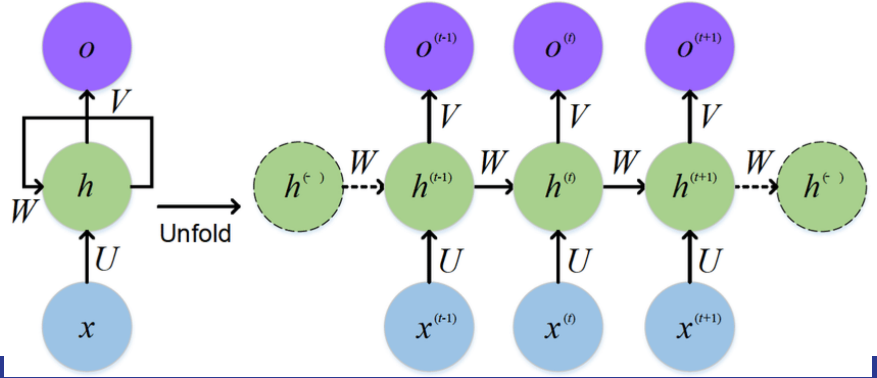


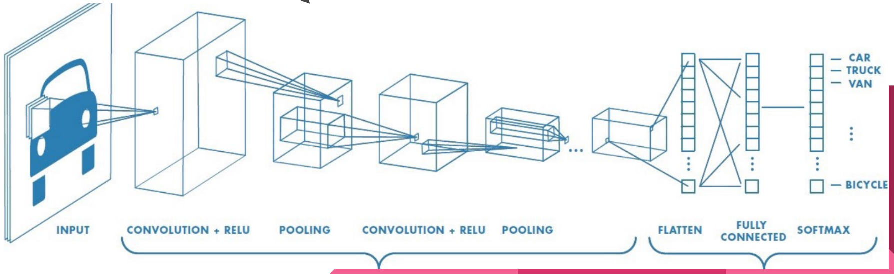
```
[19]: 950067.0/1015077.0

[19]: 0.9359555974571387
```

Model Selection - What is source code?

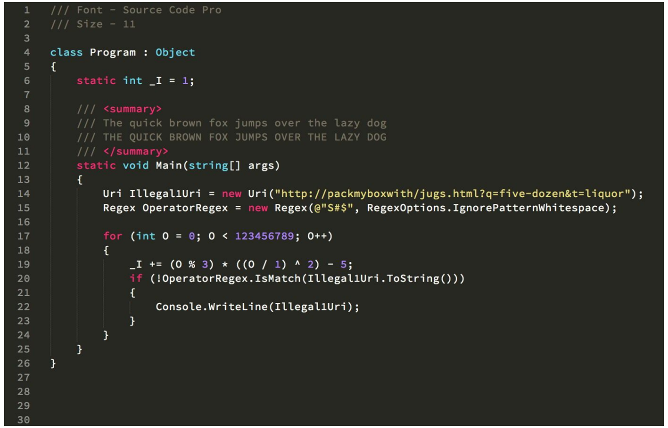


```
/// Font - Source Code Pro
/// Size - 11

class Program : Object
{
    static int _I = 1;

    /// <summary>
    /// The quick brown fox jumps over the lazy dog
    /// THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
    /// </summary>
    static void Main(string[] args)
    {
        Uri Illegal1Uri = new Uri("http://packmyboxwith/jugs.html?q=five-dozen&t=liquor");
        Regex OperatorRegex = new Regex(@"S#$", RegexOptions.IgnorePatternWhitespace);

        for (int O = 0; O < 123456789; O++)
        {
            _I += (O % 3) * ((O / 1) ^ 2) - 5;
            if (!OperatorRegex.IsMatch(Illegal1Uri.ToString()))
            {
                Console.WriteLine(Illegal1Uri);
            }
        }
    }
}
```

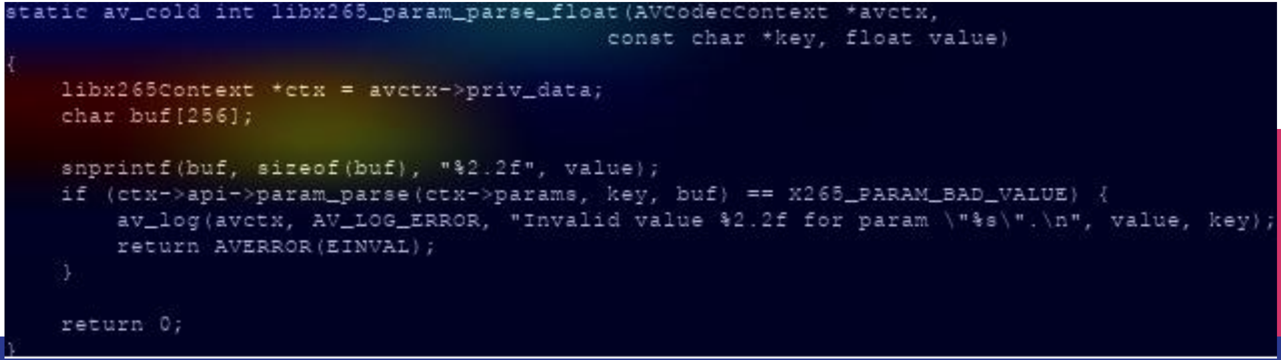




Model Selection - Source code as an Image

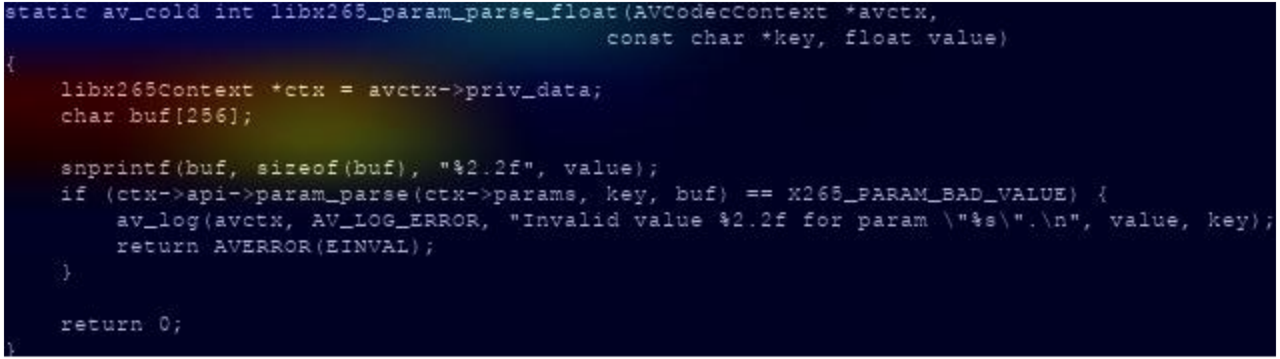


```
/// Font - Source Code Pro
/// Size - 11

class Program : Object
{
    static int _I = 1;

    /// <summary>
    /// The quick brown fox jumps over the lazy dog
    /// THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
    /// </summary>
    static void Main(string[] args)
    {
        Uri Illegal1Uri = new Uri("http://packmyboxwith/jugs.html?q=five-dozen&t=liquor");
        Regex OperatorRegex = new Regex(@"S#$", RegexOptions.IgnorePatternWhitespace);

        for (int O = 0; O < 123456789; O++)
        {
            _I += (O % 3) * ((O / 1) ^ 2) - 5;
            if (!OperatorRegex.IsMatch(Illegal1Uri.ToString()))
            {
                Console.WriteLine(Illegal1Uri);
            }
        }
    }
}
```



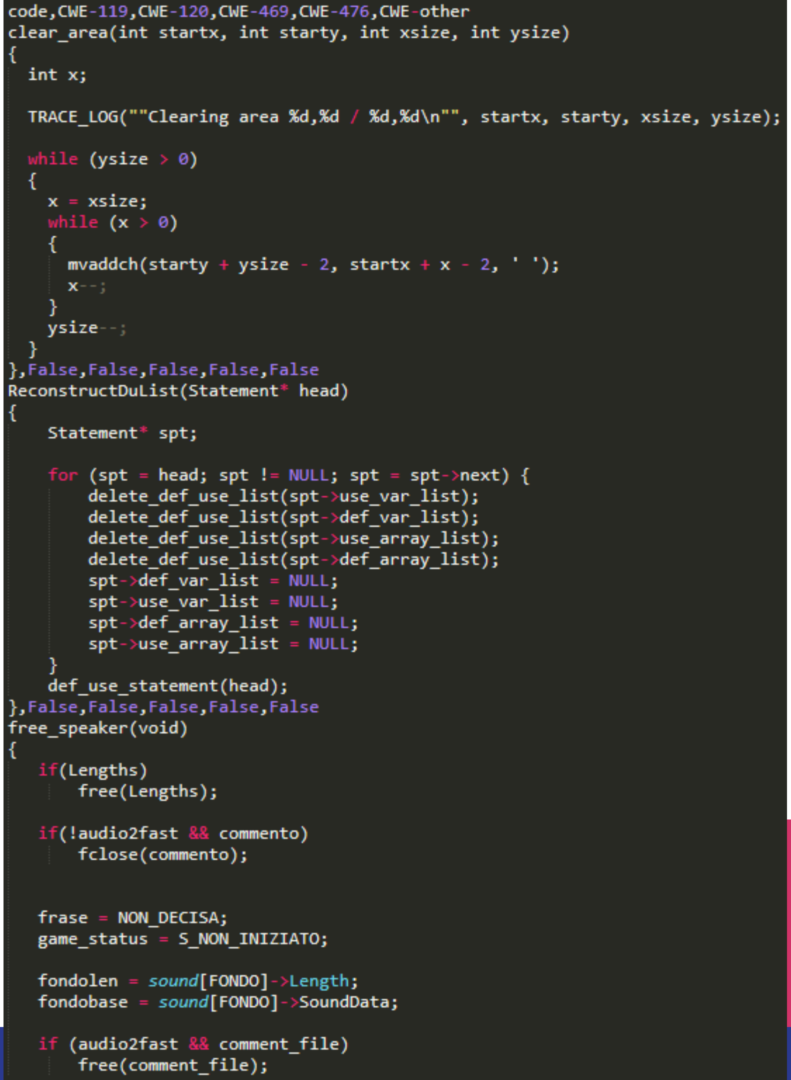
```
static av_cold int libx265_param_parse_float(AVCodecContext *avctx,
                                             const char *key, float value)
{
    libx265Context *ctx = avctx->priv_data;
    char buf[256];

    snprintf(buf, sizeof(buf), "%2.2f", value);
    if (ctx->api->param_parse(ctx->params, key, buf) == X265_PARAM_BAD_VALUE) {
        av_log(avctx, AV_LOG_ERROR, "Invalid value %2.2f for param \"%s\".\n", value, key);
        return AVERROR(EINVAL);
    }

    return 0;
}
```

Model Selection - Source code as an Image



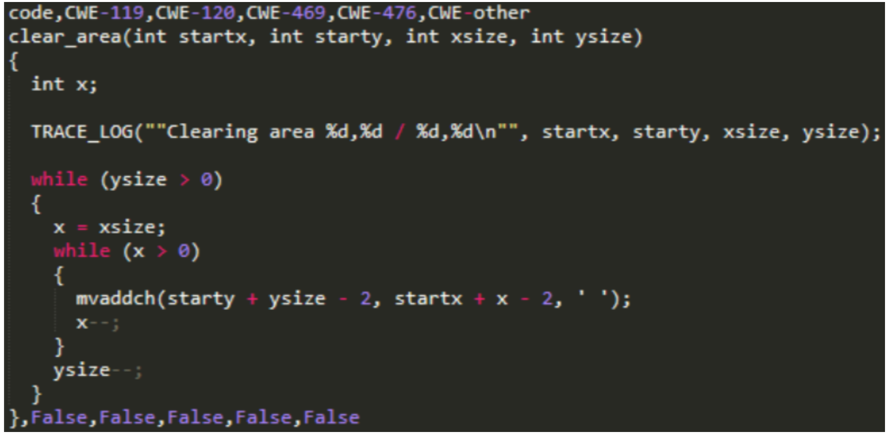
```
static av_cold int libx265_param_parse_float(AVCodecContext *avctx,
                                              const char *key, float value)
{
    libx265Context *ctx = avctx->priv_data;
    char buf[256];

    snprintf(buf, sizeof(buf), "%2.2f", value);
    if (ctx->api->param_parse(ctx->params, key, buf) == X265_PARAM_BAD_VALUE) {
        av_log(avctx, AV_LOG_ERROR, "Invalid value %2.2f for param \"%s\".\n", value, key);
        return AVERROR(EINVAL);
    }

    return 0;
}
```

Dataset Processing

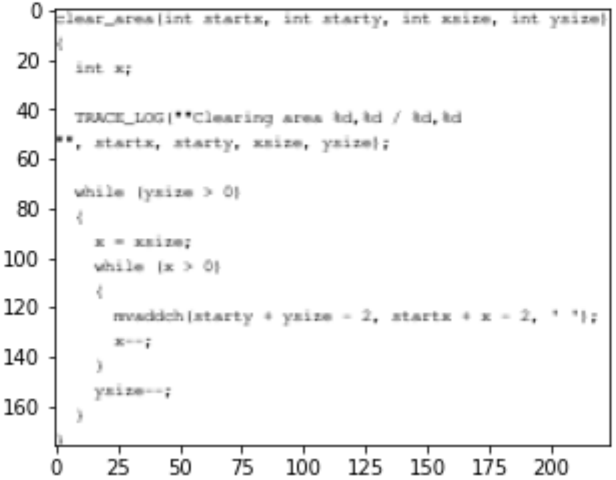


```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
},False,False,False,False,False
ReconstructDuList(Statement* head)
{
    Statement* spt;

    for (spt = head; spt != NULL; spt = spt->next) {
        delete_def_use_list(spt->use_var_list);
        delete_def_use_list(spt->def_var_list);
        delete_def_use_list(spt->use_array_list);
        delete_def_use_list(spt->def_array_list);
        spt->def_var_list = NULL;
        spt->use_var_list = NULL;
        spt->def_array_list = NULL;
        spt->use_array_list = NULL;
    }
    def_use_statement(head);
},False,False,False,False,False
free_speaker(void)
{
   if(Lengths)
       free(Lengths);

   if(!audio2fast && commento)
       fclose(commento);


   frase = NON_DECISA;
   game_status = S_NON_INIZIATO;

   fondolen = sound[FONDO]->Length;
   fondobase = sound[FONDO]->SoundData;

   if (audio2fast && comment_file)
       free(comment_file);
```

Dataset Processing

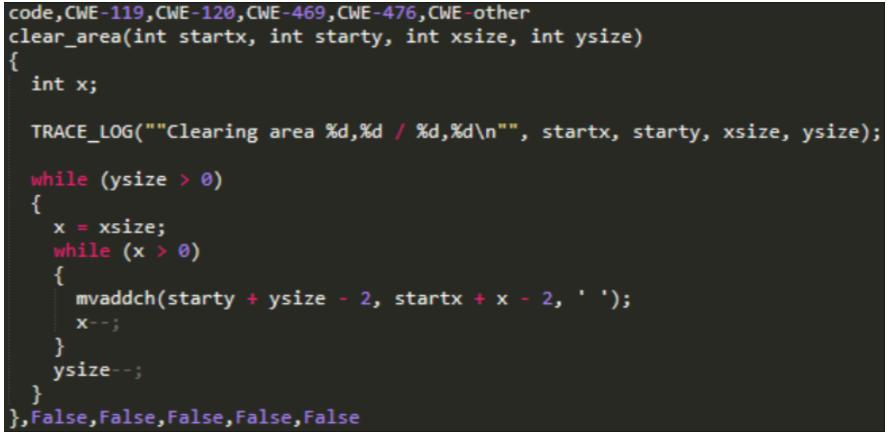


```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
},False,False,False,False,False
```

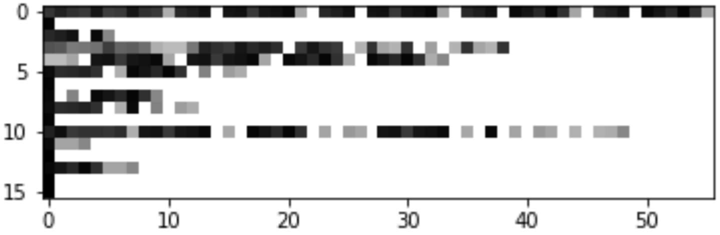


```
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d
"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
}
```

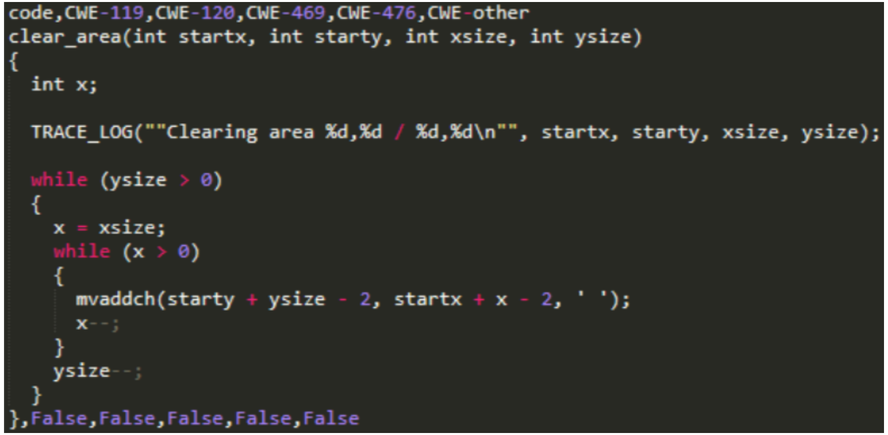
Dataset Processing



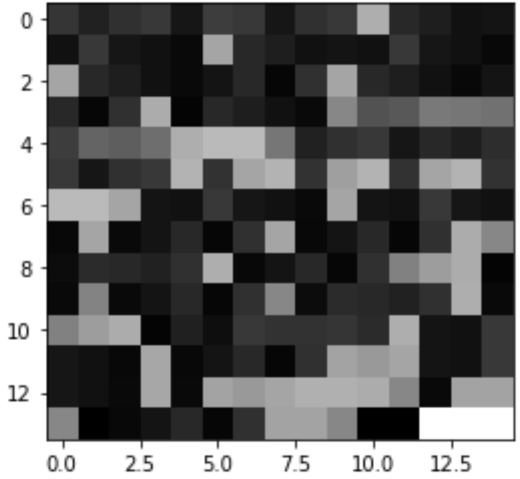
```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
},False,False,False,False,False
```



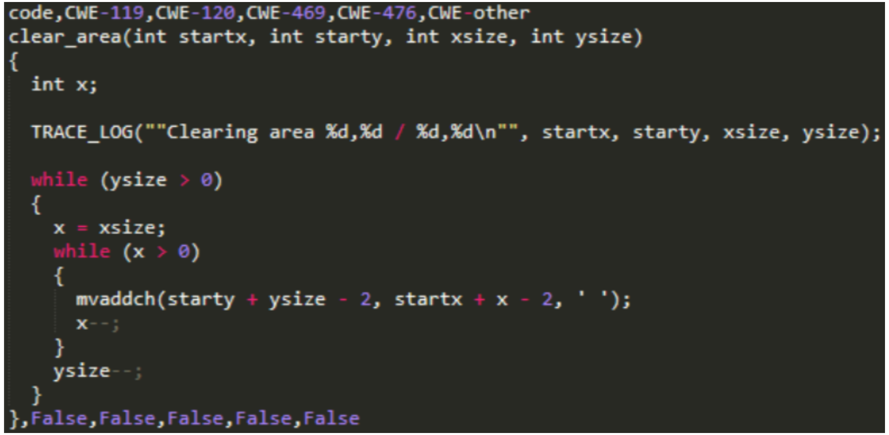
Dataset Processing



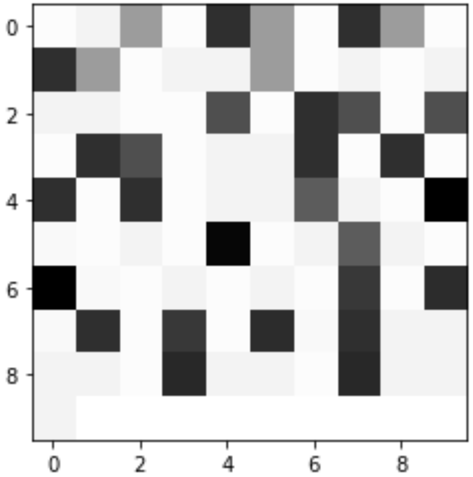
```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
},False,False,False,False,False
```



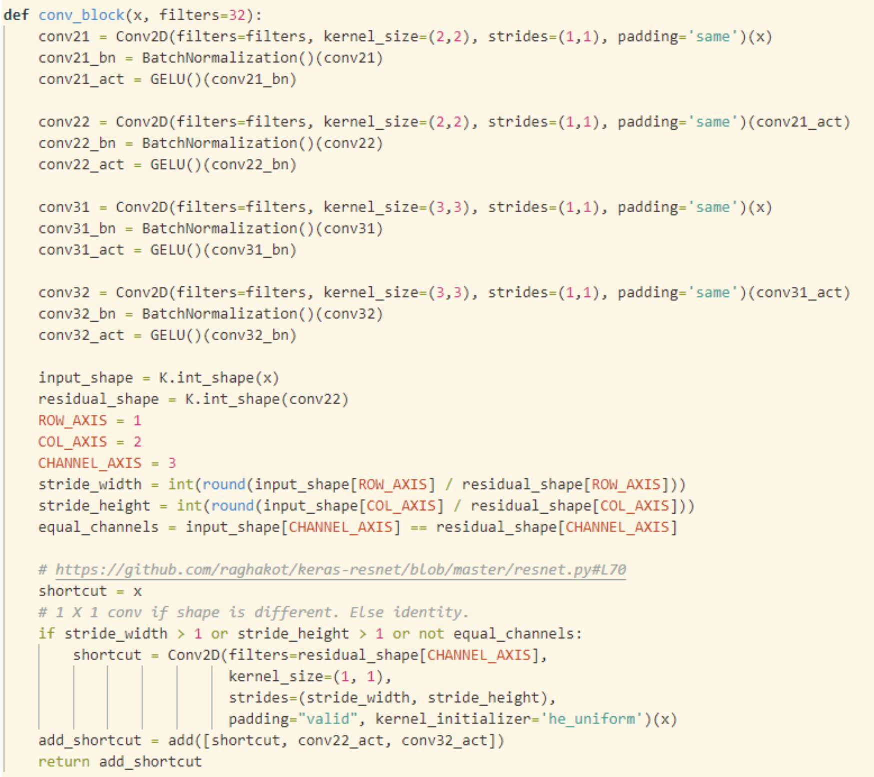
Dataset Processing



```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
  int x;

  TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

  while (ysize > 0)
  {
    x = xsize;
    while (x > 0)
    {
      mvaddch(starty + ysize - 2, startx + x - 2, ' ');
      x--;
    }
    ysize--;
  }
},False,False,False,False,False
```

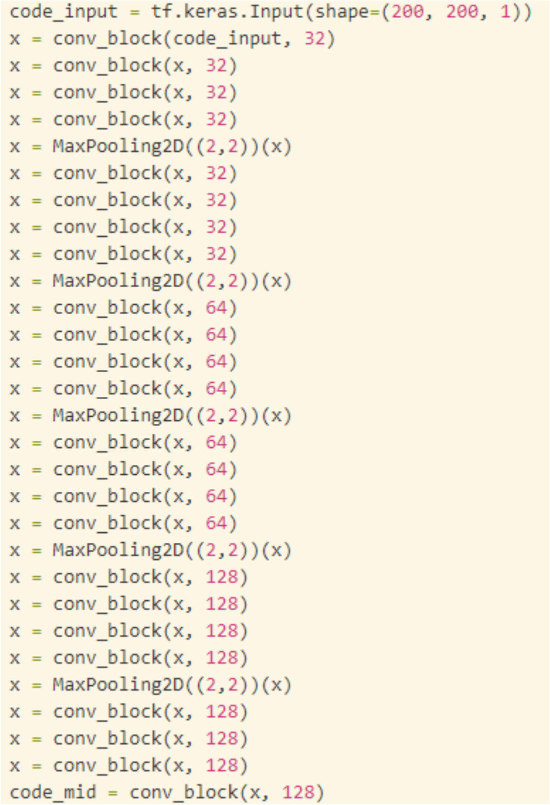


Model design

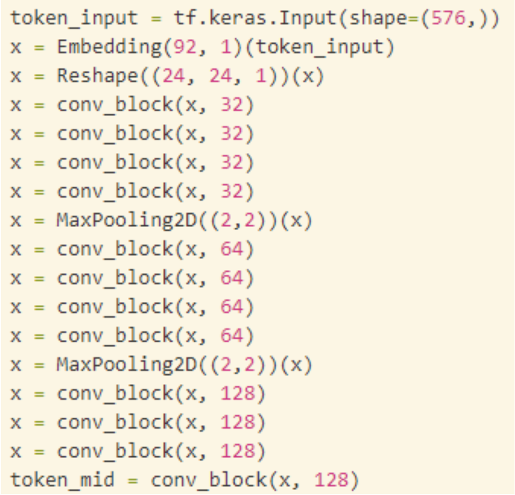


```python
def conv_block(x, filters=32):
    conv21 = Conv2D(filters=filters, kernel_size=(2,2), strides=(1,1), padding='same')(x)
    conv21_bn = BatchNormalization()(conv21)
    conv21_act = GELU()(conv21_bn)

    conv22 = Conv2D(filters=filters, kernel_size=(2,2), strides=(1,1), padding='same')(conv21_act)
    conv22_bn = BatchNormalization()(conv22)
    conv22_act = GELU()(conv22_bn)

    conv31 = Conv2D(filters=filters, kernel_size=(3,3), strides=(1,1), padding='same')(x)
    conv31_bn = BatchNormalization()(conv31)
    conv31_act = GELU()(conv31_bn)

    conv32 = Conv2D(filters=filters, kernel_size=(3,3), strides=(1,1), padding='same')(conv31_act)
    conv32_bn = BatchNormalization()(conv32)
    conv32_act = GELU()(conv32_bn)

    input_shape = K.int_shape(x)
    residual_shape = K.int_shape(conv22)
    ROW_AXIS = 1
    COL_AXIS = 2
    CHANNEL_AXIS = 3
    stride_width = int(round(input_shape[ROW_AXIS] / residual_shape[ROW_AXIS]))
    stride_height = int(round(input_shape[COL_AXIS] / residual_shape[COL_AXIS]))
    equal_channels = input_shape[CHANNEL_AXIS] == residual_shape[CHANNEL_AXIS]

    # https://github.com/raghakot/keras-resnet/blob/master/resnet.py#L70
    shortcut = x
    # 1 X 1 conv if shape is different. Else identity.
    if stride_width > 1 or stride_height > 1 or not equal_channels:
        shortcut = Conv2D(filters=residual_shape[CHANNEL_AXIS],
                          kernel_size=(1, 1),
                          strides=(stride_width, stride_height),
                          padding="valid", kernel_initializer='he_uniform')(x)
    add_shortcut = add([shortcut, conv22_act, conv32_act])
    return add_shortcut
```

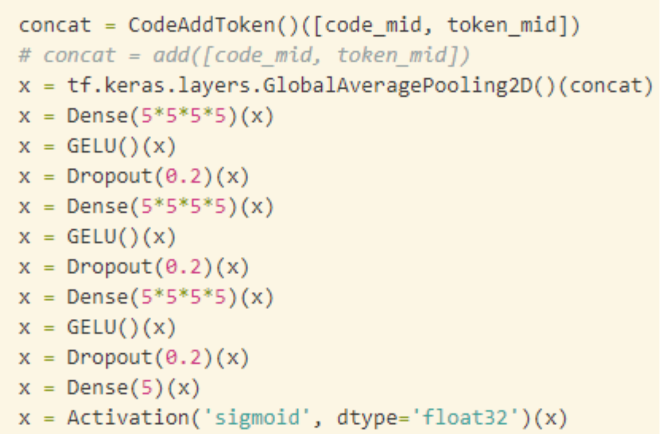
Model design



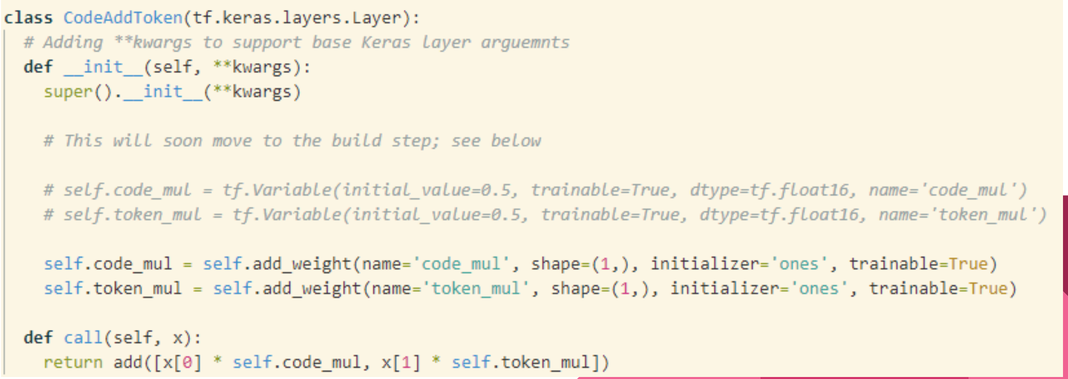
```
code_input = tf.keras.Input(shape=(200, 200, 1))
x = conv_block(code_input, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 128)
x = conv_block(x, 128)
x = conv_block(x, 128)
x = conv_block(x, 128)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 128)
x = conv_block(x, 128)
x = conv_block(x, 128)
code_mid = conv_block(x, 128)
```



```
token_input = tf.keras.Input(shape=(576,))
x = Embedding(92, 1)(token_input)
x = Reshape((24, 24, 1))(x)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = conv_block(x, 32)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = conv_block(x, 64)
x = MaxPooling2D((2,2))(x)
x = conv_block(x, 128)
x = conv_block(x, 128)
x = conv_block(x, 128)
token_mid = conv_block(x, 128)
```



```
concat = CodeAddToken()([code_mid, token_mid])
# concat = add([code_mid, token_mid])
x = tf.keras.layers.GlobalAveragePooling2D()(concat)
x = Dense(5*5*5*5)(x)
x = GELU()(x)
x = Dropout(0.2)(x)
x = Dense(5*5*5*5)(x)
x = GELU()(x)
x = Dropout(0.2)(x)
x = Dense(5*5*5*5)(x)
x = GELU()(x)
x = Dropout(0.2)(x)
x = Dense(5)(x)
x = Activation('sigmoid', dtype='float32')(x)
```



```
class CodeAddToken(tf.keras.layers.Layer):
  # Adding **kwargs to support base Keras layer arguemnts
  def __init__(self, **kwargs):
    super().__init__(**kwargs)

    # This will soon move to the build step; see below

    # self.code_mul = tf.Variable(initial_value=0.5, trainable=True, dtype=tf.float16, name='code_mul')
    # self.token_mul = tf.Variable(initial_value=0.5, trainable=True, dtype=tf.float16, name='token_mul')

    self.code_mul = self.add_weight(name='code_mul', shape=(1,), initializer='ones', trainable=True)
    self.token_mul = self.add_weight(name='token_mul', shape=(1,), initializer='ones', trainable=True)

  def call(self, x):
    return add([x[0] * self.code_mul, x[1] * self.token_mul])
```

Training

optimizer : Adam(lr=1e-3)

loss : binary focal loss

batch size : 64

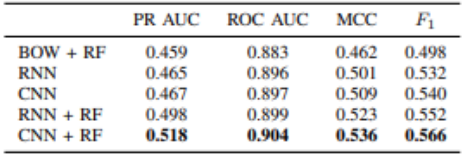
System : RTX 3090 * 2

Minutes per epoch : 7min~15min

Total epoch : 39

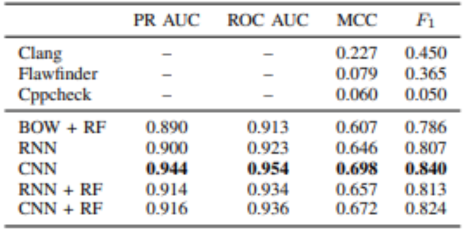
best epoch : 29

Evaluation - Xception



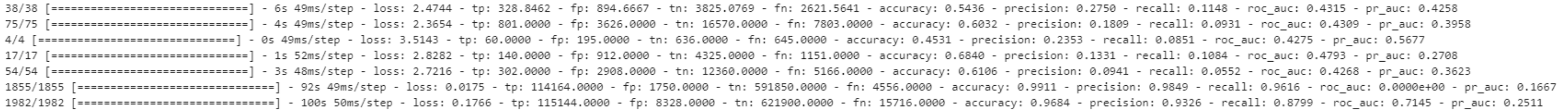
	PR AUC	ROC AUC	MCC	F_1
BOW + RF	0.459	0.883	0.462	0.498
RNN	0.465	0.896	0.501	0.532
CNN	0.467	0.897	0.509	0.540
RNN + RF	0.498	0.899	0.523	0.552
CNN + RF	**0.518**	**0.904**	**0.536**	**0.566**

TABLE III: Results on the Debian and GitHub test data for our ML models, corresponding to Figure 2.



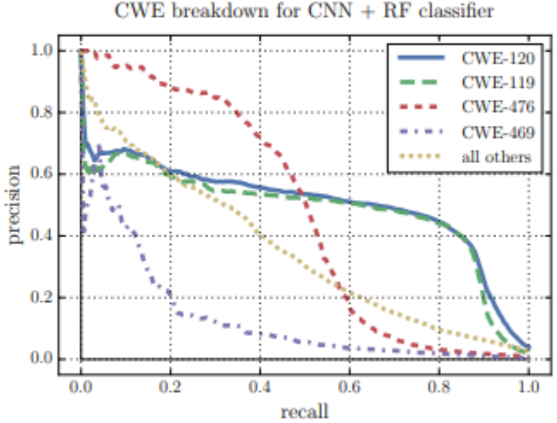
	PR AUC	ROC AUC	MCC	F_1
Clang	–	–	0.227	0.450
Flawfinder	–	–	0.079	0.365
Cppcheck	–	–	0.060	0.050
BOW + RF	0.890	0.913	0.607	0.786
RNN	0.900	0.923	0.646	0.807
CNN	**0.944**	**0.954**	**0.698**	**0.840**
RNN + RF	0.914	0.934	0.657	0.813
CNN + RF	0.916	0.936	0.672	0.824

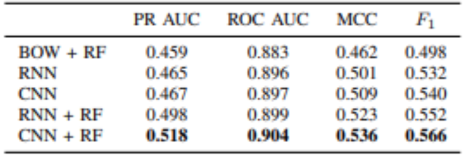
TABLE IV: Results on the SATE IV Juliet Suite test data for our ML models and three static analyzers, as in Figure 3.



```
38/38 [==============================] - 6s 49ms/step - loss: 2.4744 - tp: 328.8462 - fp: 894.6667 - tn: 3825.0769 - fn: 2621.5641 - accuracy: 0.5436 - precision: 0.2750 - recall: 0.1148 - roc_auc: 0.4315 - pr_auc: 0.4258
75/75 [==============================] - 4s 49ms/step - loss: 2.3654 - tp: 801.0000 - fp: 3626.0000 - tn: 16570.0000 - fn: 7803.0000 - accuracy: 0.6032 - precision: 0.1809 - recall: 0.0931 - roc_auc: 0.4309 - pr_auc: 0.3958
4/4 [==============================] - 0s 49ms/step - loss: 3.5143 - tp: 60.0000 - fp: 195.0000 - tn: 636.0000 - fn: 645.0000 - accuracy: 0.4531 - precision: 0.2353 - recall: 0.0851 - roc_auc: 0.4275 - pr_auc: 0.5677
17/17 [==============================] - 1s 52ms/step - loss: 2.8282 - tp: 140.0000 - fp: 912.0000 - tn: 4325.0000 - fn: 1151.0000 - accuracy: 0.6840 - precision: 0.1331 - recall: 0.1084 - roc_auc: 0.4793 - pr_auc: 0.2708
54/54 [==============================] - 3s 48ms/step - loss: 2.7216 - tp: 302.0000 - fp: 2908.0000 - tn: 12360.0000 - fn: 5166.0000 - accuracy: 0.6106 - precision: 0.0941 - recall: 0.0552 - roc_auc: 0.4268 - pr_auc: 0.3623
1855/1855 [==============================] - 92s 49ms/step - loss: 0.0175 - tp: 114164.0000 - fp: 1750.0000 - tn: 591850.0000 - fn: 4556.0000 - accuracy: 0.9911 - precision: 0.9849 - recall: 0.9616 - roc_auc: 0.0000e+00 - pr_auc: 0.1667
1982/1982 [==============================] - 100s 50ms/step - loss: 0.1766 - tp: 115144.0000 - fp: 8328.0000 - tn: 621900.0000 - fn: 15716.0000 - accuracy: 0.9684 - precision: 0.9326 - recall: 0.8799 - roc_auc: 0.7145 - pr_auc: 0.2511
```

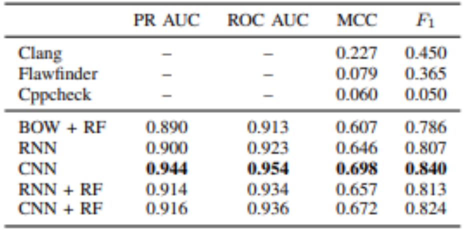
Evaluation - VulnViz Mk.2





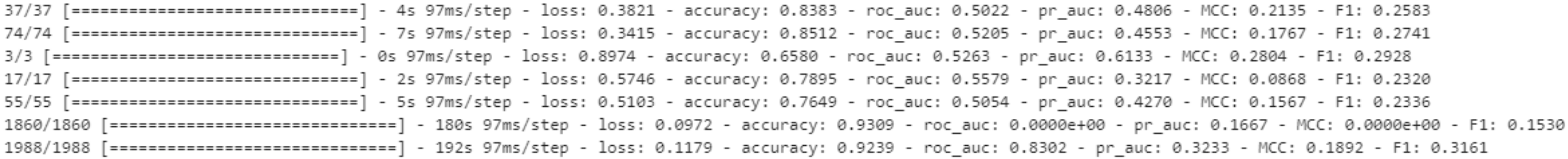
	PR AUC	ROC AUC	MCC	F_1
BOW + RF	0.459	0.883	0.462	0.498
RNN	0.465	0.896	0.501	0.532
CNN	0.467	0.897	0.509	0.540
RNN + RF	0.498	0.899	0.523	0.552
CNN + RF	**0.518**	**0.904**	**0.536**	**0.566**

TABLE III: Results on the Debian and GitHub test data for our ML models, corresponding to Figure 2.



	PR AUC	ROC AUC	MCC	F_1
Clang	–	–	0.227	0.450
Flawfinder	–	–	0.079	0.365
Cppcheck	–	–	0.060	0.050
BOW + RF	0.890	0.913	0.607	0.786
RNN	0.900	0.923	0.646	0.807
CNN	**0.944**	**0.954**	**0.698**	**0.840**
RNN + RF	0.914	0.934	0.657	0.813
CNN + RF	0.916	0.936	0.672	0.824

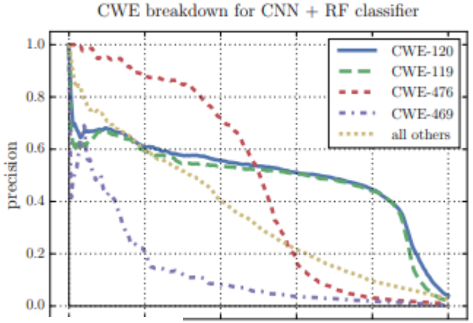
TABLE IV: Results on the SATE IV Juliet Suite test data for our ML models and three static analyzers, as in Figure 3.

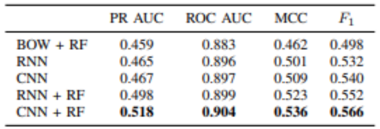


```
37/37 [==============================] - 4s 97ms/step - loss: 0.3821 - accuracy: 0.8383 - roc_auc: 0.5022 - pr_auc: 0.4806 - MCC: 0.2135 - F1: 0.2583
74/74 [==============================] - 7s 97ms/step - loss: 0.3415 - accuracy: 0.8512 - roc_auc: 0.5205 - pr_auc: 0.4553 - MCC: 0.1767 - F1: 0.2741
3/3 [==============================] - 0s 97ms/step - loss: 0.8974 - accuracy: 0.6580 - roc_auc: 0.5263 - pr_auc: 0.6133 - MCC: 0.2804 - F1: 0.2928
17/17 [==============================] - 2s 97ms/step - loss: 0.5746 - accuracy: 0.7895 - roc_auc: 0.5579 - pr_auc: 0.3217 - MCC: 0.0868 - F1: 0.2320
55/55 [==============================] - 5s 97ms/step - loss: 0.5103 - accuracy: 0.7649 - roc_auc: 0.5054 - pr_auc: 0.4270 - MCC: 0.1567 - F1: 0.2336
1860/1860 [==============================] - 180s 97ms/step - loss: 0.0972 - accuracy: 0.9309 - roc_auc: 0.0000e+00 - pr_auc: 0.1667 - MCC: 0.0000e+00 - F1: 0.1530
1988/1988 [==============================] - 192s 97ms/step - loss: 0.1179 - accuracy: 0.9239 - roc_auc: 0.8302 - pr_auc: 0.3233 - MCC: 0.1892 - F1: 0.3161
```

Evaluation - VulnViz Mk.3

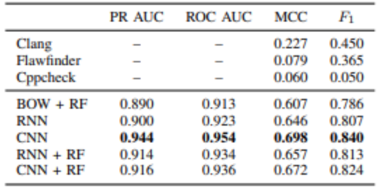
CWE breakdown for CNN + RF classifier





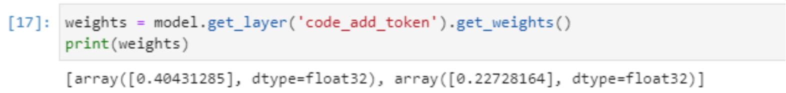
	PR AUC	ROC AUC	MCC	F_1
BOW + RF	0.459	0.883	0.462	0.498
RNN	0.465	0.896	0.501	0.532
CNN	0.467	0.897	0.509	0.540
RNN + RF	0.498	0.899	0.523	0.552
CNN + RF	**0.518**	**0.904**	**0.536**	**0.566**

TABLE III: Results on the Debian and GitHub test data for our ML models, corresponding to Figure 2.

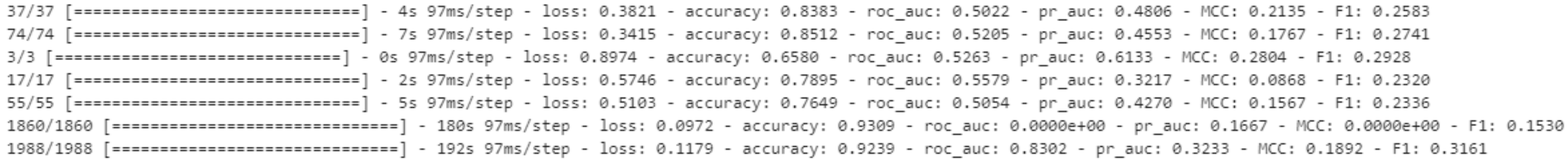


	PR AUC	ROC AUC	MCC	F_1
Clang	–	–	0.227	0.450
Flawfinder	–	–	0.079	0.365
Cppcheck	–	–	0.060	0.050
BOW + RF	0.890	0.913	0.607	0.786
RNN	0.900	0.923	0.646	0.807
CNN	**0.944**	**0.954**	**0.698**	**0.840**
RNN + RF	0.914	0.934	0.657	0.813
CNN + RF	0.916	0.936	0.672	0.824

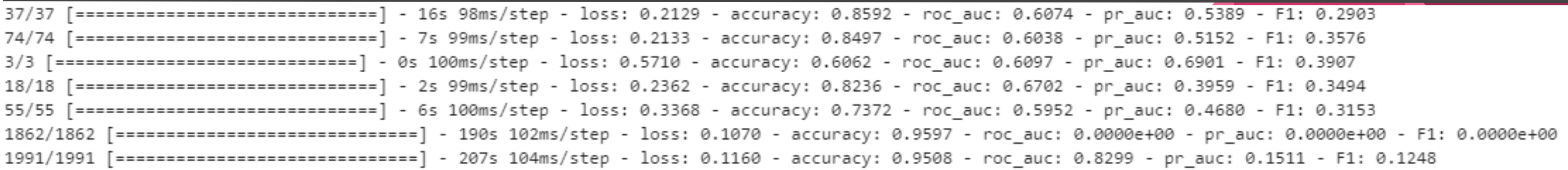
TABLE IV: Results on the SATE IV Juliet Suite test data for our ML models and three static analyzers, as in Figure 3.



```
[17]: weights = model.get_layer('code_add_token').get_weights()
      print(weights)

      [array([0.40431285], dtype=float32), array([0.22728164], dtype=float32)]
```

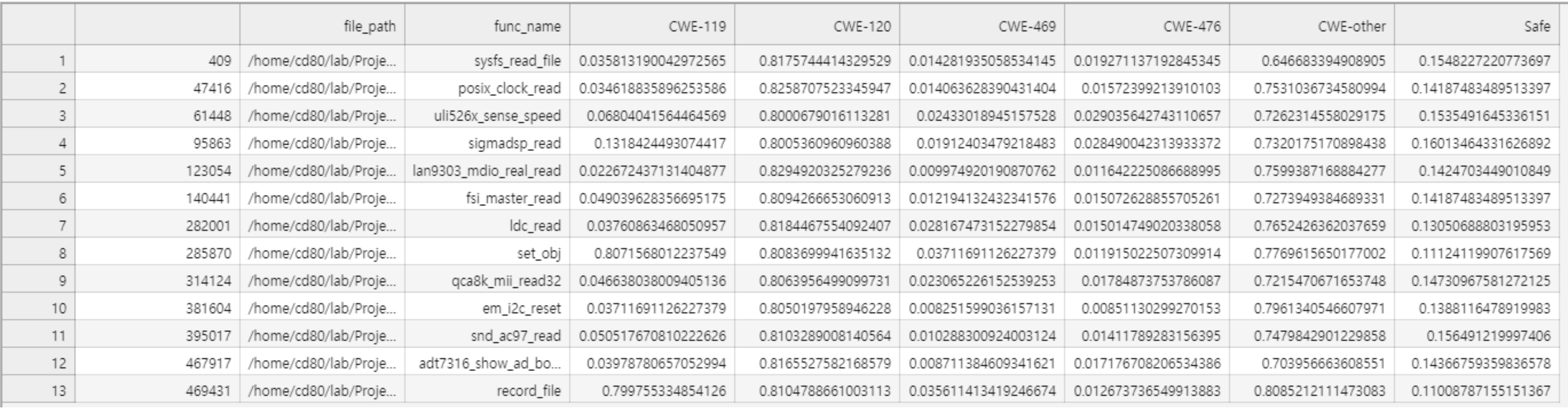


```
37/37 [==============================] - 4s 97ms/step - loss: 0.3821 - accuracy: 0.8383 - roc_auc: 0.5022 - pr_auc: 0.4806 - MCC: 0.2135 - F1: 0.2583
74/74 [==============================] - 7s 97ms/step - loss: 0.3415 - accuracy: 0.8512 - roc_auc: 0.5205 - pr_auc: 0.4553 - MCC: 0.1767 - F1: 0.2741
3/3 [==============================] - 0s 97ms/step - loss: 0.8974 - accuracy: 0.6580 - roc_auc: 0.5263 - pr_auc: 0.6133 - MCC: 0.2804 - F1: 0.2928
17/17 [==============================] - 2s 97ms/step - loss: 0.5746 - accuracy: 0.7895 - roc_auc: 0.5579 - pr_auc: 0.3217 - MCC: 0.0868 - F1: 0.2320
55/55 [==============================] - 5s 97ms/step - loss: 0.5103 - accuracy: 0.7649 - roc_auc: 0.5054 - pr_auc: 0.4270 - MCC: 0.1567 - F1: 0.2336
1860/1860 [==============================] - 180s 97ms/step - loss: 0.0972 - accuracy: 0.9309 - roc_auc: 0.0000e+00 - pr_auc: 0.1667 - MCC: 0.0000e+00 - F1: 0.1530
1988/1988 [==============================] - 192s 97ms/step - loss: 0.1179 - accuracy: 0.9239 - roc_auc: 0.8302 - pr_auc: 0.3233 - MCC: 0.1892 - F1: 0.3161
```



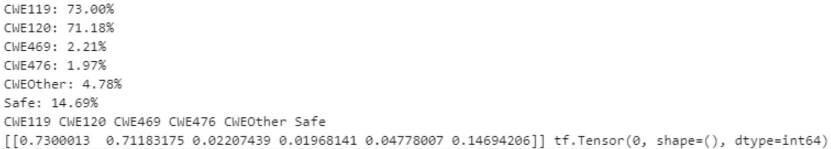
```
37/37 [==============================] - 16s 98ms/step - loss: 0.2129 - accuracy: 0.8592 - roc_auc: 0.6074 - pr_auc: 0.5389 - F1: 0.2903
74/74 [==============================] - 7s 99ms/step - loss: 0.2133 - accuracy: 0.8497 - roc_auc: 0.6038 - pr_auc: 0.5152 - F1: 0.3576
3/3 [==============================] - 0s 100ms/step - loss: 0.5710 - accuracy: 0.6062 - roc_auc: 0.6097 - pr_auc: 0.6901 - F1: 0.3907
18/18 [==============================] - 2s 99ms/step - loss: 0.2362 - accuracy: 0.8236 - roc_auc: 0.6702 - pr_auc: 0.3959 - F1: 0.3494
55/55 [==============================] - 6s 100ms/step - loss: 0.3368 - accuracy: 0.7372 - roc_auc: 0.5952 - pr_auc: 0.4680 - F1: 0.3153
1862/1862 [==============================] - 190s 102ms/step - loss: 0.1070 - accuracy: 0.9597 - roc_auc: 0.0000e+00 - pr_auc: 0.0000e+00 - F1: 0.0000e+00
1991/1991 [==============================] - 207s 104ms/step - loss: 0.1160 - accuracy: 0.9508 - roc_auc: 0.8299 - pr_auc: 0.1511 - F1: 0.1248
```

Final Result

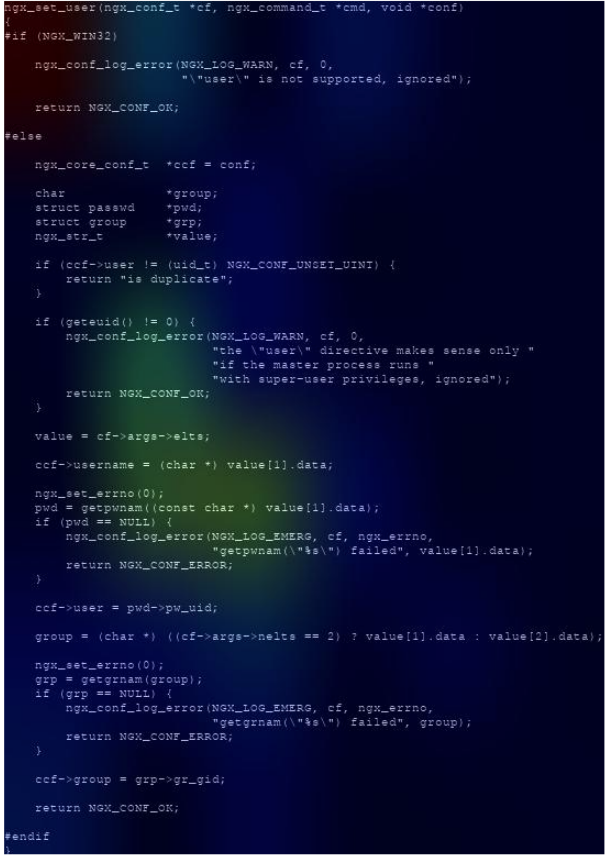


		file_path	func_name	CWE-119	CWE-120	CWE-469	CWE-476	CWE-other	Safe
1	409	/home/cd80/lab/Proje...	sysfs_read_file	0.035813190042972565	0.8175744414329529	0.014281935058534145	0.019271137192845345	0.646683394908905	0.1548227220773697
2	47416	/home/cd80/lab/Proje...	posix_clock_read	0.034618835896253586	0.8258707523345947	0.014063628390431404	0.01572399213910103	0.7531036734580994	0.14187483489513397
3	61448	/home/cd80/lab/Proje...	uli526x_sense_speed	0.06804041564464569	0.8000679016113281	0.02433018945157528	0.029035642743110657	0.7262314558029175	0.1535491645336151
4	95863	/home/cd80/lab/Proje...	sigmadsp_read	0.1318424493074417	0.8005360960960388	0.01912403479218483	0.028490042313933372	0.7320175170898438	0.16013464331626892
5	123054	/home/cd80/lab/Proje...	lan9303_mdio_real_read	0.022672437131404877	0.829492032525279236	0.009974920190870762	0.011642225086688995	0.7599387168884277	0.1424703449010849
6	140441	/home/cd80/lab/Proje...	fsi_master_read	0.049039628356695175	0.8094266653060913	0.012194132432341576	0.015072628855705261	0.7273949384689331	0.14187483489513397
7	282001	/home/cd80/lab/Proje...	ldc_read	0.03760863468050957	0.8184467554092407	0.028167473152279854	0.015014749020338058	0.7652426362037659	0.13050688803195953
8	285870	/home/cd80/lab/Proje...	set_obj	0.8071568012237549	0.8083699941635132	0.03711691126227379	0.011915022507309914	0.7769615650177002	0.11124119907617569
9	314124	/home/cd80/lab/Proje...	qca8k_mii_read32	0.046638038009405136	0.8063956499099731	0.023065226152539253	0.01784873753786087	0.7215470671653748	0.14730967581272125
10	381604	/home/cd80/lab/Proje...	em_i2c_reset	0.03711691126227379	0.8050197958946228	0.008251599036157131	0.00851130299270153	0.7961340546607971	0.1388116478919983
11	395017	/home/cd80/lab/Proje...	snd_ac97_read	0.050517670810222626	0.8103289008140564	0.010288300924003124	0.01411789283156395	0.7479842901229858	0.15649121997406
12	467917	/home/cd80/lab/Proje...	adt7316_show_ad_bo...	0.03978780657052994	0.8165527582168579	0.008711384609341621	0.017176708206534386	0.703956663608551	0.14366759359836578
13	469431	/home/cd80/lab/Proje...	record_file	0.799755334854126	0.8104788661003113	0.035611413419246674	0.012673736549913883	0.8085212111473083	0.11008787155151367

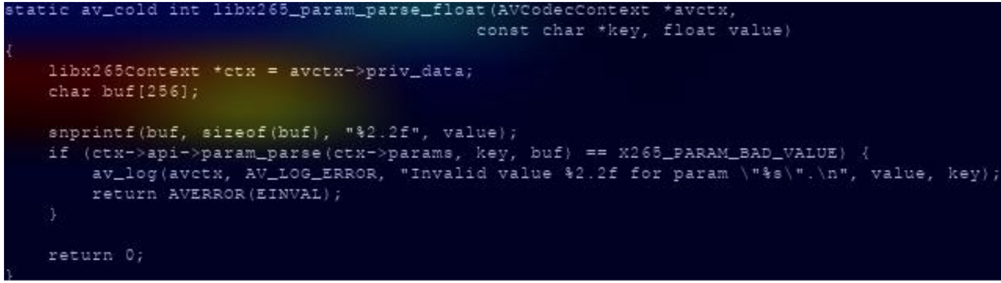
CAM



```
CWE119: 73.00%
CWE120: 71.18%
CWE469: 2.21%
CWE476: 1.97%
CWEOther: 4.78%
Safe: 14.69%
CWE119 CWE120 CWE469 CWE476 CWEOther Safe
[[0.7300013  0.71183175 0.02207439 0.01968141 0.04778007 0.14694206]] tf.Tensor(0, shape=(), dtype=int64)
```

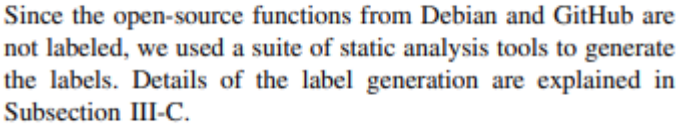


```
ngx_set_user(ngx_conf_t *cf, ngx_command_t *cmd, void *conf)
{
#if (NGX_WIN32)

    ngx_conf_log_error(NGX_LOG_WARN, cf, 0,
                       "\"user\" is not supported, ignored");

    return NGX_CONF_OK;

#else

    ngx_core_conf_t  *ccf = conf;

    char             *group;
    struct passwd    *pwd;
    struct group     *grp;
    ngx_str_t        *value;

    if (ccf->user != (uid_t) NGX_CONF_UNSET_UINT) {
        return "is duplicate";
    }

    if (geteuid() != 0) {
        ngx_conf_log_error(NGX_LOG_WARN, cf, 0,
                           "the \"user\" directive makes sense only "
                           "if the master process runs "
                           "with super-user privileges, ignored");
        return NGX_CONF_OK;
    }

    value = cf->args->elts;

    ccf->username = (char *) value[1].data;

    ngx_set_errno(0);
    pwd = getpwnam((const char *) value[1].data);
    if (pwd == NULL) {
        ngx_conf_log_error(NGX_LOG_EMERG, cf, ngx_errno,
                           "getpwnam(\"%s\") failed", value[1].data);
        return NGX_CONF_ERROR;
    }

    ccf->user = pwd->pw_uid;

    group = (char *) ((cf->args->nelts == 2) ? value[1].data : value[2].data);

    ngx_set_errno(0);
    grp = getgrnam(group);
    if (grp == NULL) {
        ngx_conf_log_error(NGX_LOG_EMERG, cf, ngx_errno,
                           "getgrnam(\"%s\") failed", group);
        return NGX_CONF_ERROR;
    }

    ccf->group = grp->gr_gid;

    return NGX_CONF_OK;

#endif
}
```

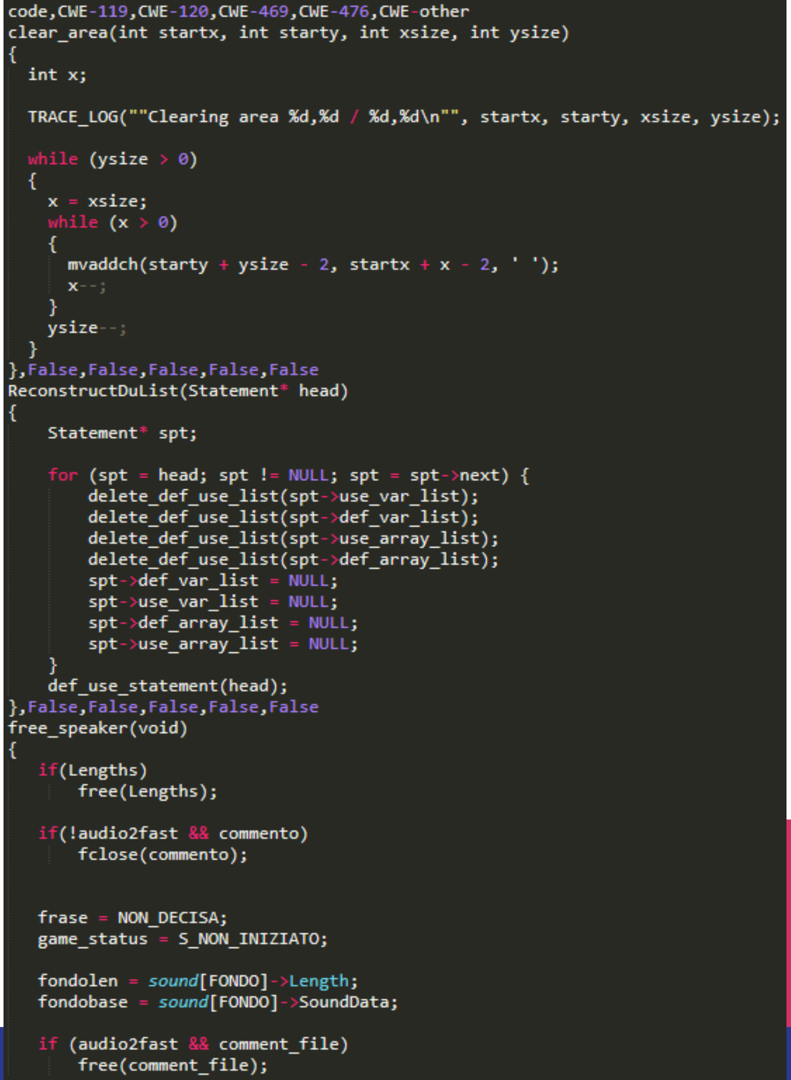


```
static av_cold int libx265_param_parse_float(AVCodecContext *avctx,
                                           const char *key, float value)
{
    libx265Context *ctx = avctx->priv_data;
    char buf[256];

    snprintf(buf, sizeof(buf), "%2.2f", value);
    if (ctx->api->param_parse(ctx->params, key, buf) == X265_PARAM_BAD_VALUE) {
        av_log(avctx, AV_LOG_ERROR, "Invalid value %2.2f for param \"%s\".\n", value, key);
        return AVERROR(EINVAL);
    }

    return 0;
}
```

Future works

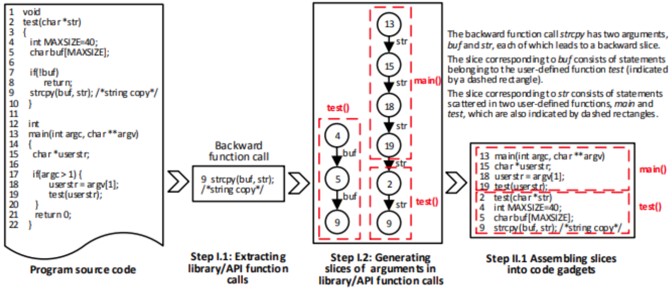


Since the open-source functions from Debian and GitHub are not labeled, we used a suite of static analysis tools to generate the labels. Details of the label generation are explained in Subsection III-C.



```
code,CWE-119,CWE-120,CWE-469,CWE-476,CWE-other
clear_area(int startx, int starty, int xsize, int ysize)
{
 int x;

 TRACE_LOG(""Clearing area %d,%d / %d,%d\n"", startx, starty, xsize, ysize);

 while (ysize > 0)
 {
  x = xsize;
  while (x > 0)
  {
   mvaddch(starty + ysize - 2, startx + x - 2, ' ');
   x--;
  }
  ysize--;
 }
},False,False,False,False,False
ReconstructDuList(Statement* head)
{
  Statement* spt;

  for (spt = head; spt != NULL; spt = spt->next) {
    delete_def_use_list(spt->use_var_list);
    delete_def_use_list(spt->def_var_list);
    delete_def_use_list(spt->use_array_list);
    delete_def_use_list(spt->def_array_list);
    spt->def_var_list = NULL;
    spt->use_var_list = NULL;
    spt->def_array_list = NULL;
    spt->use_array_list = NULL;
  }
  def_use_statement(head);
},False,False,False,False,False
free_speaker(void)
{
  if(Lengths)
    free(Lengths);

  if(!audio2fast && commento)
    fclose(commento);


  frase = NON_DECISA;
  game_status = S_NON_INIZIATO;

  fondolen = sound[FONDO]->Length;
  fondobase = sound[FONDO]->SoundData;

  if (audio2fast && comment_file)
    free(comment_file);
```

Future works



```
1  void
2  test(char *str)
3  {
4    int MAXSIZE=40;
5    char buf[MAXSIZE];
6
7    if(!buf)
8        return;
9    strcpy(buf, str); /*string copy*/
10  }
11
12  int
13  main(int argc, char **argv)
14  {
15   char *userstr;
16
17   if(argc > 1) {
18        userstr = argv[1];
19        test(userstr);
20    }
21    return 0;
22  }
```

Program source code

Backward function call

```
9 strcpy(buf, str);
/*string copy*/
```

Step I.1: Extracting library/API function calls

Step I.2: Generating slices of arguments in library/API function calls

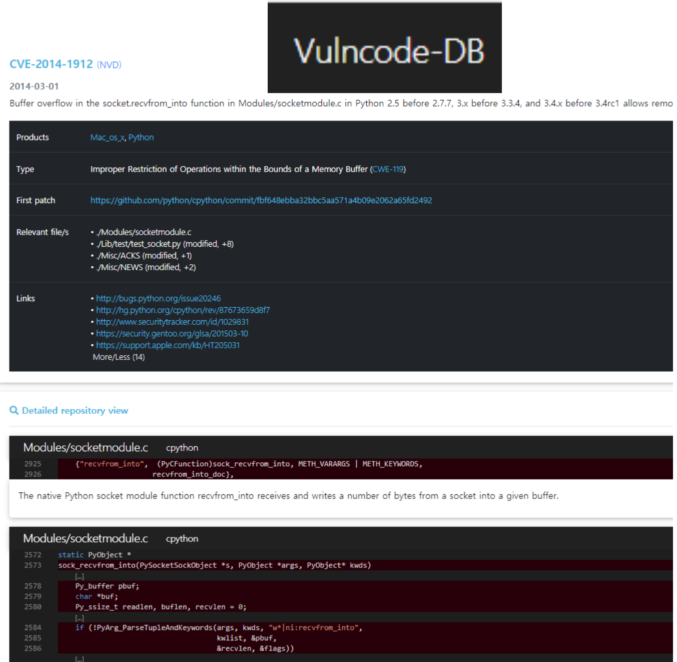
The backward function call *strcpy* has two arguments, *buf* and *str*, each of which leads to a backward slice.

The slice corresponding to *buf* consists of statements belonging to the user-defined function *test* (indicated by a dashed rectangle).

The slice corresponding to *str* consists of statements scattered in two user-defined functions, *main* and *test*, which are also indicated by dashed rectangles.

```
13  main(int argc, char **argv)
15  char *userstr;
18  userstr = argv[1];
19  test(userstr);
2   test(char *str)
4   int MAXSIZE=40;
5   charbuf[MAXSIZE];
9   strcpy(buf, str); /*string copy*/
```

Step II.1 Assembling slices into code gadgets



Vulncode-DB

CVE-2014-1912 (NVD)

2014-03-01

Buffer overflow in the socket.recvfrom_into function in Modules/socketmodule.c in Python 2.5 before 2.7.7, 3.x before 3.3.4, and 3.4.x before 3.4rc1 allows remo

Products	Mac_os_x, Python
Type	Improper Restriction of Operations within the Bounds of a Memory Buffer (CWE-119)
First patch	https://github.com/python/cpython/commit/fbf648ebba32bbc5aa571a4b09e2062a65fd2492
Relevant file/s	• ./Modules/socketmodule.c • ./Lib/test/test_socket.py (modified, +8) • ./Misc/ACKS (modified, +1) • ./Misc/NEWS (modified, +2)
Links	• http://bugs.python.org/issue20246 • http://hg.python.org/cpython/rev/87673659d8f7 • http://www.securitytracker.com/id/1029831 • https://security.gentoo.org/glsa/201503-10 • https://support.apple.com/kb/HT205031 More/Less (14)

Detailed repository view

Modules/socketmodule.c cpython

```
    {"recvfrom_into",  (PyCFunction)sock_recvfrom_into, METH_VARARGS | METH_KEYWORDS,
                       recvfrom_into_doc},
```

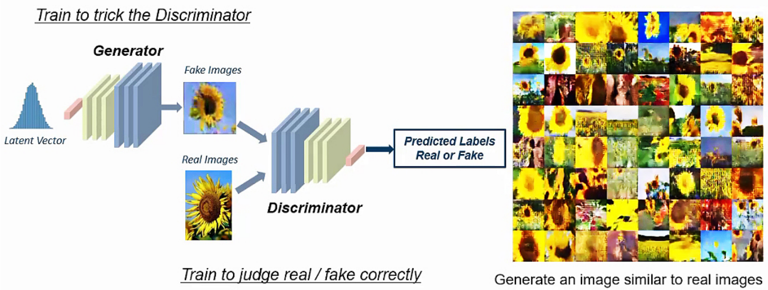
The native Python socket module function recvfrom_into receives and writes a number of bytes from a socket into a given buffer.

Modules/socketmodule.c cpython

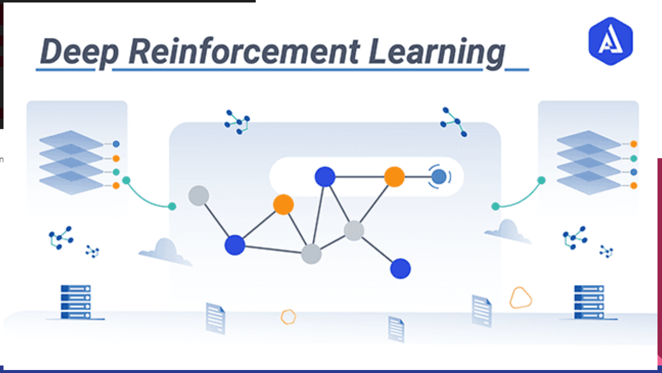
```
static PyObject *
sock_recvfrom_into(PySocketSockObject *s, PyObject *args, PyObject* kwds)
    [...]
    Py_buffer pbuf;
    char *buf;
    Py_ssize_t readlen, buflen, recvlen = 0;
    [...]
    if (!PyArg_ParseTupleAndKeywords(args, kwds, "w*|ni:recvfrom_into",
                                     kwlist, &pbuf,
                                     &recvlen, &flags))
    [...]
    buf = pbuf.buf;
    buflen = pbuf.len;
    [...]
    if (recvlen < 0) {
    [...]
    if (recvlen == 0) {
        /* If nbytes was not specified, use the buffer's length */
    [...]
    }

    readlen = sock_recvfrom_guts(s, buf, recvlen, flags, &addr);
```

This is called from Python as socket.recvfrom_into(buffer[, nbytes[, flags]]).
The C function sock_recvfrom_into then creates a buffer structure buf for the purpose of receivin

What is GAN(Generative Adversarial Network)?



Generate an image similar to real images



감사합니다

긴 질문은 메일로

rkwk0112@gmail.com