Recognizing and Imitating Programmer Style: Adversaries in Program Authorship Attribution

Lucy Simko, Luke Zettlemoyer, Tadayoshi Kohno

simkol@cs.washington.edu
homes.cs.washington.edu/~simkol

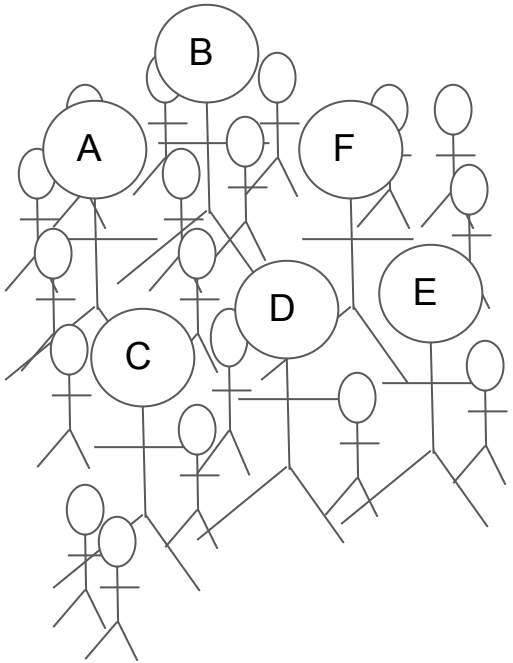
PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

Source Code Attribution

```
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
...
```

?



State of the Art: Source Code Attribution

Caliskan-Islam et al. “**De-anonymizing programmers via code stylometry**.” *24th USENIX Security Symposium (USENIX Security), Washington, DC*. 2015.

- 98% accuracy over 250 programmers
- Extract syntactic, lexical, and layout features from C/C++ code
- Random Forest classifier
- Data set: Google Code Jam
  - Programming competition
  - Lots of examples of people solving the same problem in different ways
- Open source

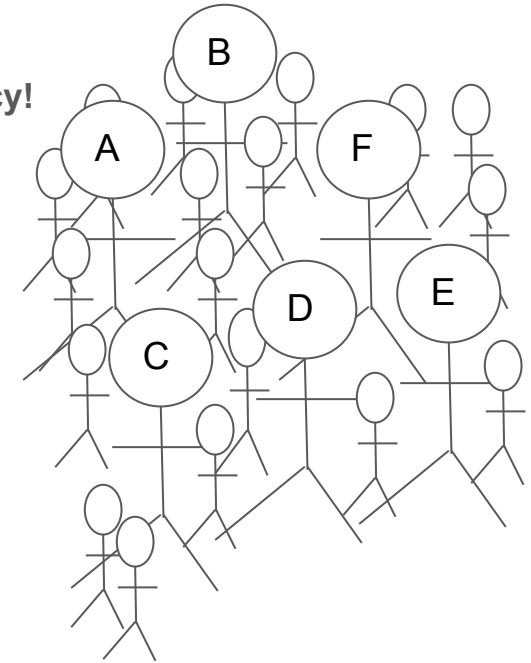
Source Code Attribution

```
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
...
```

98% accuracy!

?



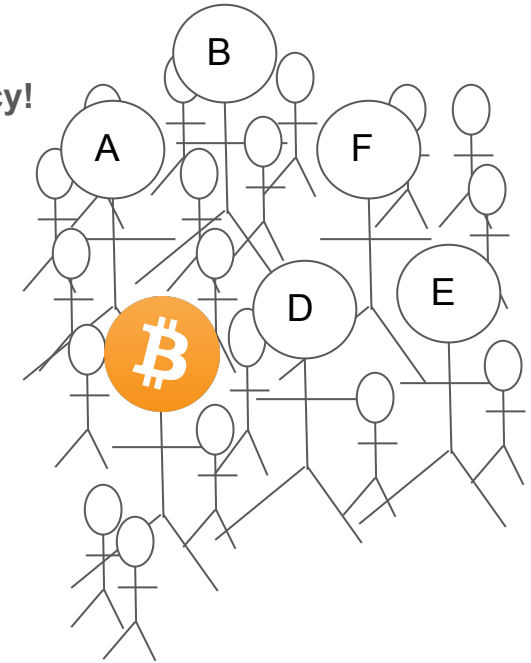
Source Code Attribution

```
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
...
```

98% accuracy!

?



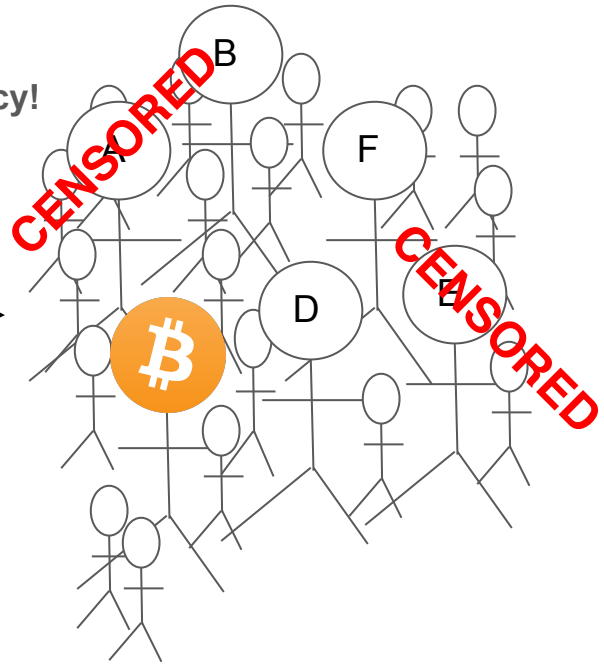
Source Code Attribution

```
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
...
```

98% accuracy!

?



Research Question

Can we fool source code attribution classifiers?

Yes!

Methodology: Lab study* with C programmers

*Approved by University of Washington's Human Subjects Division (IRB)

Outline

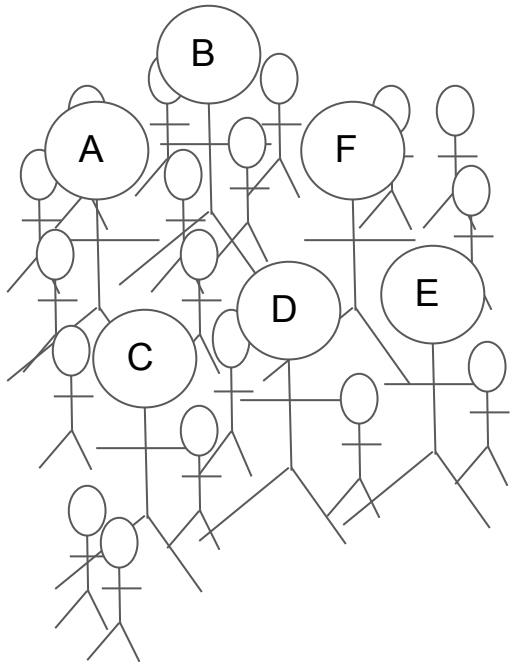
- Motivation and Research Question
- **Source Code Attribution: Overview and Background**
- Evading Source Code Attribution: Definitions and Goals
- Methodology
- Results: Conservative Estimate of Adversarial Success
- Results: How to Create Forgeries

Source Code Attribution

```
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
...
```

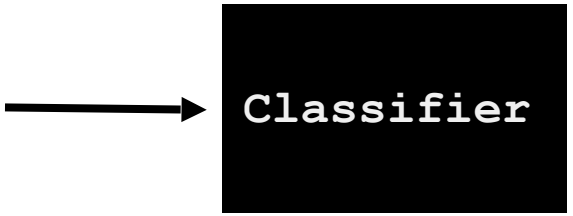
?

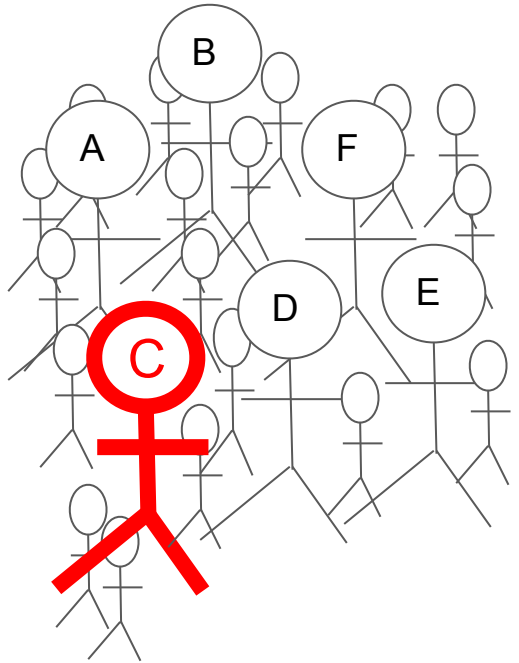


Source Code Attribution

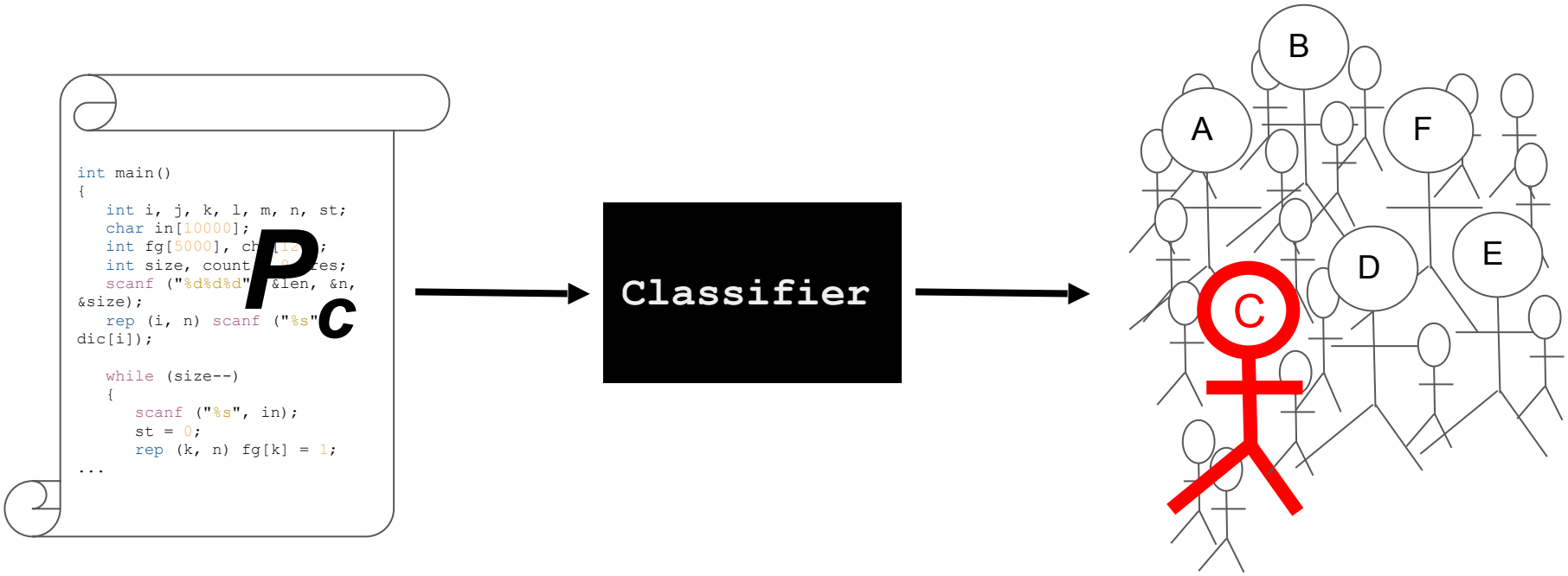
```
int main()
{
   int i, j, k, l, m, n, st;
   char in[10000];
   int fg[5000], chk[128];
   int size, count = 0, res;
   scanf ("%d%d%d", &len, &n,
&size);
   rep (i, n) scanf ("%s",
dic[i]);

   while (size--)
   {
      scanf ("%s", in);
      st = 0;
      rep (k, n) fg[k] = 1;
...
```

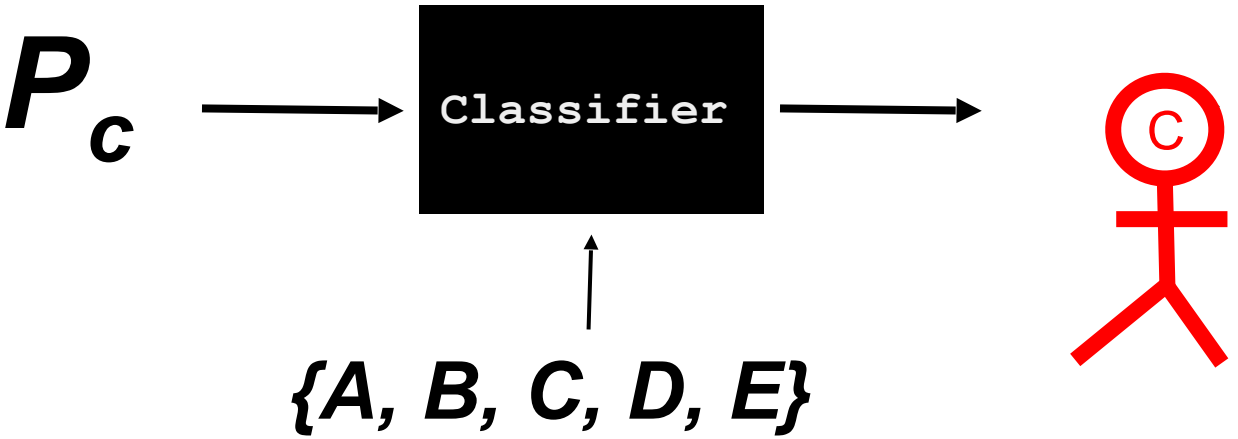




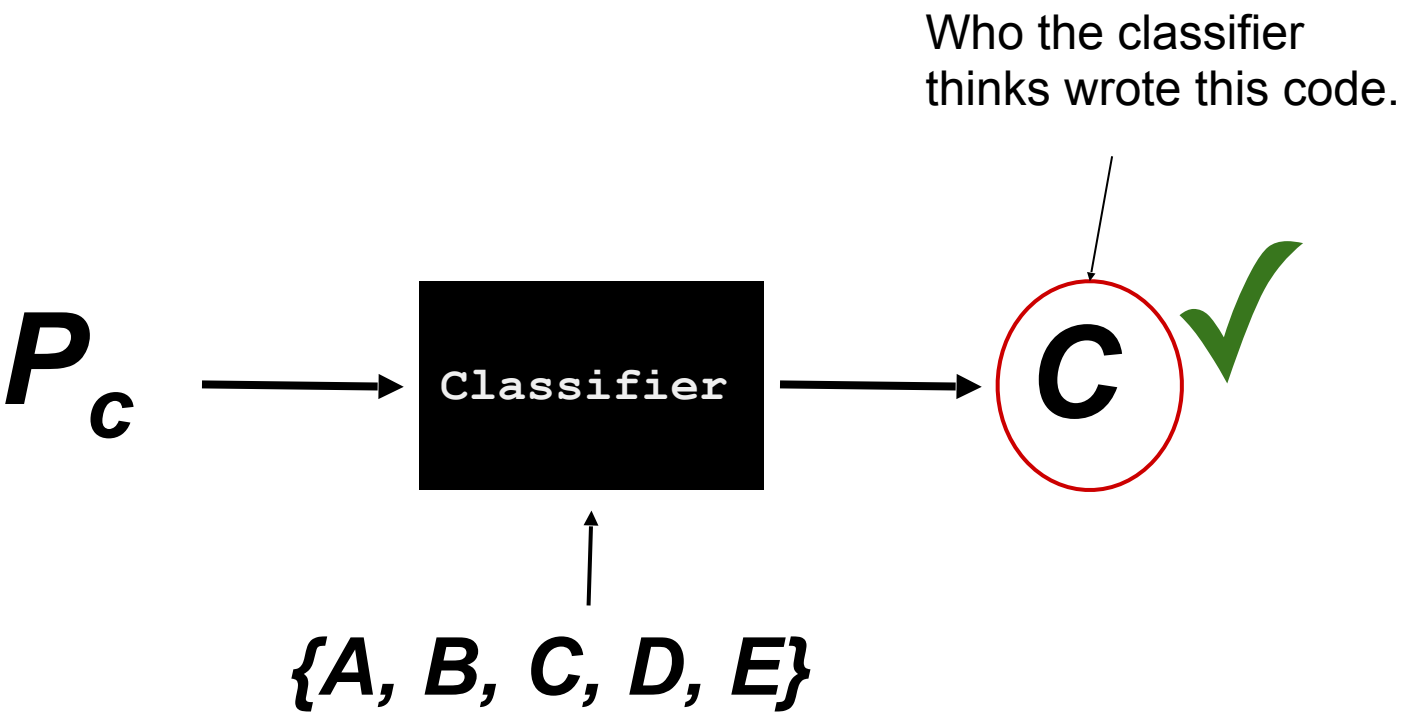
Source Code Attribution



Source Code Attribution



Source Code Attribution

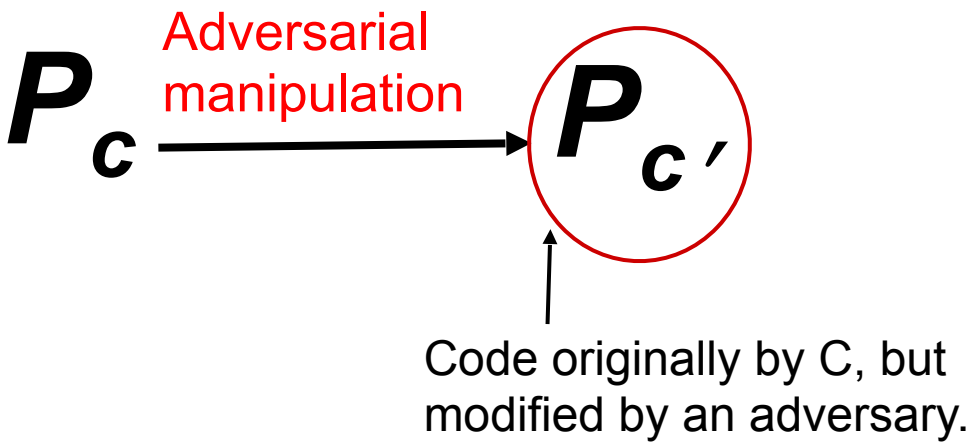


Outline

- Motivation and Research Question
- Source Code Attribution: Overview and Background
- **Evading Source Code Attribution: Definitions and Goals**
- Methodology
- Results: Conservative Estimate of Adversarial Success
- Results: How to Create Forgeries

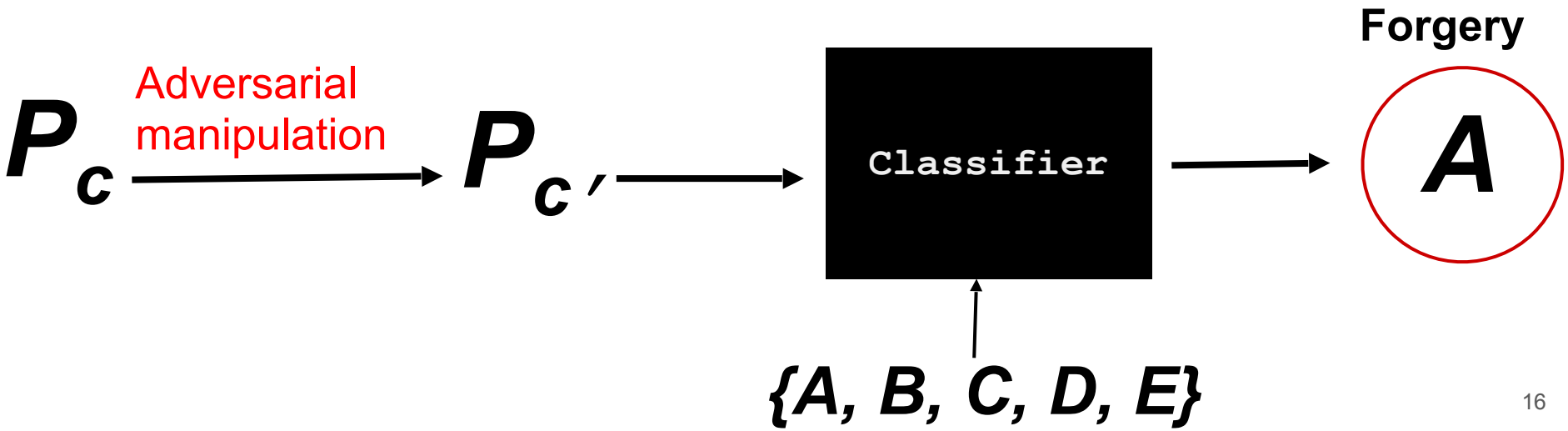
Evading Source Code Attribution

1. **Train:** Given code from **original and target authors,** learn styles
2. **Modify original code** to imitate target author (**forgery**)
   - Or just hide the original author's style (**masking**)



Evading Source Code Attribution

1. **Train:** Given code from **original and target authors,** learn styles
2. **Modify original code** to imitate target author (**forgery**)
   - Or just hide the original author's style (**masking**)



Outline

- Motivation and Research Question
- Source Code Attribution: Overview and Background
- Evading Source Code Attribution: Definitions and Goals
- **Methodology**
- Results: Conservative Estimate of Adversarial Success
- Results: How to Create Forgeries

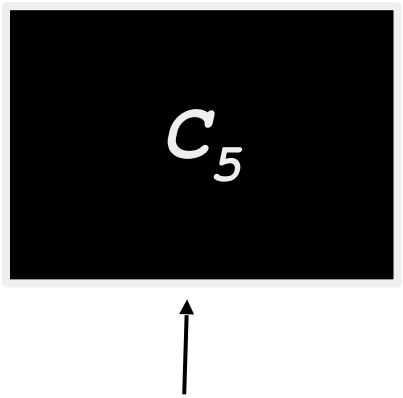
Lab Study: Dataset

code jam

- C code
- We used a linter[1] to eliminate many typographic style differences
- ~4000 authors: avg 2.2 files each
- 5 authors with the most files: avg ~42.8 files
  - Authors: A, B, C, D, E

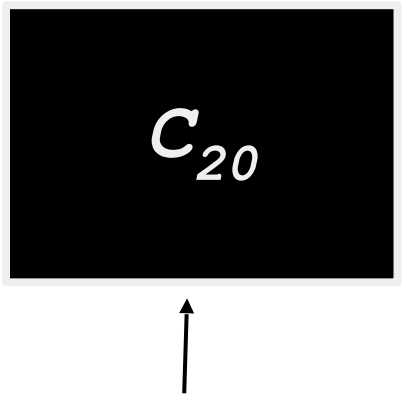
[1] http://astyle.sourceforge.net/

Lab Study: Create Forgeries



Precision: 100%
Recall: 100%
(10-fold XV)

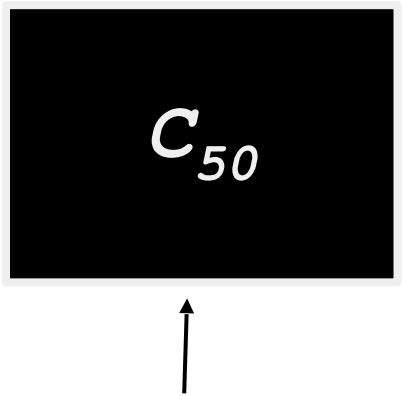
Lab Study: Create Forgeries



Precision: 87.6%
Recall: 88.2%
(10-fold XV)

{A, B, C, D, E, ... + 15}

Lab Study: Create Forgeries



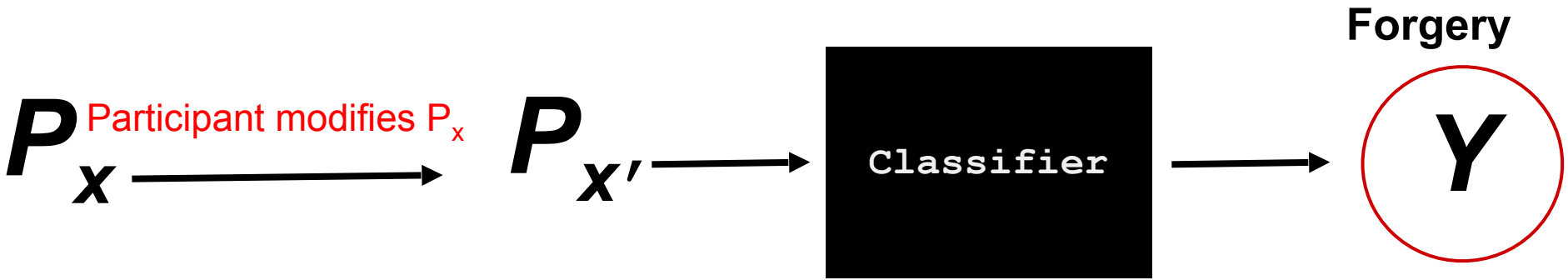
Precision: 82.3%
Recall: 84.5%
(10-fold XV)

{A, B, C, D, E, ... + 45}

Lab Study: Create Forgeries

28 C programmers (participants):

1. **Train:** Given code from **original and target author,** learn styles
2. **Modify original code** to imitate target author's style **(forgery)**

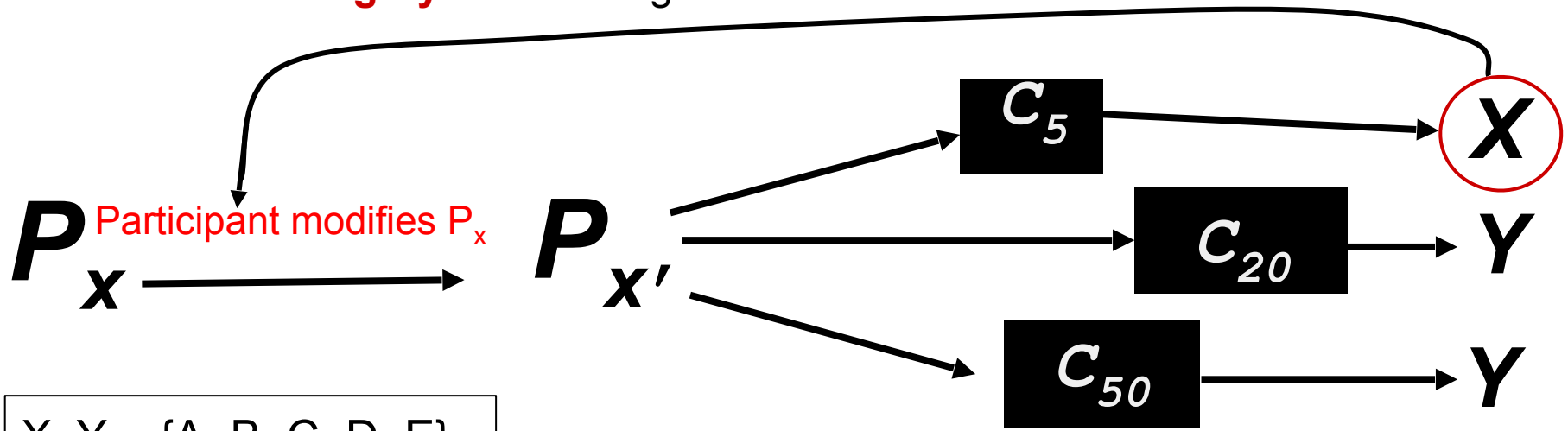


X, Y ∈ {A, B, C, D, E}

Lab Study: Create Forgeries

28 C programmers (participants):

1. **Train:** Given code from **original and target author,** learn styles
2. **Modify original code** to imitate target author's style **(forgery)**
3. **Check forgery success** against oracle classifiers



X, Y ∈ {A, B, C, D, E}

Outline

- Motivation and Research Question
- Source Code Attribution: Overview and Background
- Evading Source Code Attribution: Definitions and Goals
- Methodology
- **Results: Conservative Estimate of Adversarial Success**
- Results: How to Create Forgeries

Results: Estimate of Adversarial Success

Versions of the state-of-the-art machine classifier.
The subscript indicates the number of authors in the training set.

	C_5	C_{20}	C_{50}
Forgery	66.6%	70.0%	73.0%
Masking	76.6%	76.6%	86.6%

Percent of final forgery attempts that were successful attacks

Results: Estimate of Adversarial Success

Forgery: adversary is pretending to be a *specific target author*.
Masking: adversary is *obscuring the original author.*

	C_5	C_{20}	C_{50}
Forgery	66.6%	70.0%	73.0%
Masking	76.6%	76.6%	86.6%

Percent of final forgery attempts that were successful attacks

Results: Estimate of Adversarial Success

A *successful* forgery attack means the classifier output the target author instead of the original author of the code.
66.6% of forgery attacks against the C_5 classifier were successful.

	C_5	C_{20}	C_{50}
Forgery	66.6%	70.0%	73.0%
Masking	76.6%	76.6%	86.6%

Percent of final forgery attempts that were successful attacks

Results: Estimate of Adversarial Success

C50 attributed forgeries correctly only 13.4% of the time.

	C_5	C_{20}	C_{50}
Forgery	66.6%	70.0%	73.0%
Masking	76.6%	76.6%	86.6%

Percent of final forgery attempts that produced a misclassification

Results: Estimate of Adversarial Success

Lesson: Non-experts **can** successfully attack this state-of-the-art classifier, suggesting other authorship classifiers may be vulnerable to the same type of attacks.

	C_5	C_{20}	C_{50}
Forgery	66.6%	70.0%	73.0%
Masking	76.6%	76.6%	86.6%

Percent of final forgery attempts that produced a misclassification

Outline

- Motivation and Research Question
- Source Code Attribution: Overview and Background
- Evading Source Code Attribution: Definitions and Goals
- Methodology
- Results: Conservative Estimate of Adversarial Success
- **Results: How to Create Forgeries**

Results: Methods of Forgery Creation

Lesson: Forgers did not know the features the classifier was using for attribution. This suggests that **forgeries in the wild might contain the same types of modifications**.

Example: Two Programs by Author C

```
// libraries imported
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)
// variables defined
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
```

```
// libraries imported
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)
// variables defined
int main()
{
    int i, j, k, l, m, n, t, ok;
    int a, b, c;
    int size, count = 0;
    scanf ("%d", &size);

    while (size--)
    {
        scanf ("%d%d", &n, &m);
        rep (i, m)
        {
            scanf ("%d", s + i);
```

Example: Two Programs by Author C

```
// libraries imported
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)
// variables defined
int main()
{
    int i, j, k, l, m, n, st;
    char in[10000];
    int fg[5000], chk[128];
    int size, count = 0, res;
    scanf ("%d%d%d", &len, &n, &size);
    rep (i, n) scanf ("%s", dic[i]);

    while (size--)
    {
        scanf ("%s", in);
        st = 0;
        rep (k, n) fg[k] = 1;
```

```
// libraries imported
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)
// variables defined
int main()
{
    int i, j, k, l, m, n, t, ok;
    int a, b, c;
    int size, count = 0;
    scanf ("%d", &size);

    while (size--)
    {
        scanf ("%d%d", &n, &m);
        rep (i, m)
        {
            scanf ("%d", s + i);
```

Example: Forgery of Author C

Information Structure	Control Flow
• Variable name • Syntax • Macros • API calls	• Loop type

Example: Creating a Forgery of Author C

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    cin >> size;
    for(count=1;count<=size;count++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    cin >> size;
    for(count=1;count<=size;count++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    scanf("%d", &size);
    for(count=1;count<=size;count++)
    {
        scanf("%d%d%d%d", &D, &I, &M, &N);
        for(i=0; i<N; i++)
            scanf("%d", original+i);
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    scanf("%d", &size);
    while (size--)
    {
        scanf("%d%d%d%d", &D, &I, &M, &N);
        rep (i,N)
            scanf("%d", original+i);
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    scanf("%d", &size);
    while (size--)
    {
        scanf("%d%d%d%d", &D, &I, &M, &N);
        rep (i,N)scanf("%d", original+i);
...
```

Classifier output: ??

ORIGINAL

```
int main()
{
    int i,j,k;
    int cc,ca;
    cin >> ca;
    for(cc=1;cc<=ca;cc++)
    {
        cin >> D >> I >> M >> N;
        for(i=0; i<N; i++)
            cin >> original[i];
...
```

Classifier output: A

FORGERY

```
#define REP(i,a,b) for(i=a;i<b;i++)
#define rep(i,n) REP(i,0,n)

int main()
{
    int i,j,k;
    int size, count = 0;
    scanf("%d", &size);
    while (size--)
    {
        scanf("%d%d%d%d", &D, &I, &M, &N);
        rep (i,N)scanf("%d", original+i);
...
```

Classifier output: C

Results: Methods of Forgery Creation

Information Structure	Control Flow
• Variable name • Syntax • Macros • API calls	• Loop type

Results: Methods of Forgery Creation

Information Structure	Control Flow
• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic

Results: Methods of Forgery Creation

Local modifications: only need to understand a line or two of code

Local

Information Structure	Control Flow
• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic

Results: Methods of Forgery Creation

Local modifications: only need to understand a line or two of code

Local

Information Structure	Control Flow
• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic

Algorithmic modifications: need a more comprehensive understanding of the code

Results: Methods of Forgery Creation

Local modifications: only need to understand a line or two of code

Algorithmic modifications: need a more comprehensive understanding of the code

	Information Structure	Control Flow
Local	• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic
Algorithmic	• Variable type • Data structures • Static and dynamic memory usage	

Results: Methods of Forgery Creation

Local modifications: only need to understand a line or two of code

Algorithmic modifications: need a more comprehensive understanding of the code

	Information Structure	Control Flow
Local	• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic
Algorithmic	• Variable type • Data structures • Static and dynamic memory usage	

Results: Methods of Forgery Creation

Local modifications: only need to understand a line or two of code

Algorithmic modifications: need a more comprehensive understanding of the code

	Information Structure	Control Flow
Local	• Variable name • Syntax • Macros • API calls • Libraries imported • Variable decl location	• Loop type • If-statements • Assignments per line • Control flow keywords • Loop logic
Algorithmic	• Variable type • Data structures • Static and dynamic method usage	• Functions refactored • Inlined API calls • Major addition or removal of control structures

Results: Methods of Forgery Creation

Lessons from methods of forgery creation:

- Local modifications are common.
- Some forgers copied code directly the target author's training set.

Summary

- Programmers desiring privacy *or* with malicious intent may seek to evade source code attribution classifiers
- Lab study with C programmers producing forgeries, showing unsophisticated adversaries *can* fool a state of the art classifier
- Forgeries were successful with local changes that do not require a high-level understanding of the programming style.
- More recommendations in paper!

My coauthors: Luke Zettlemoyer, Tadayoshi Kohno
Contact me: Lucy Simko, simkol@cs.washington.edu, https://homes.cs.washington.edu/~simkol/