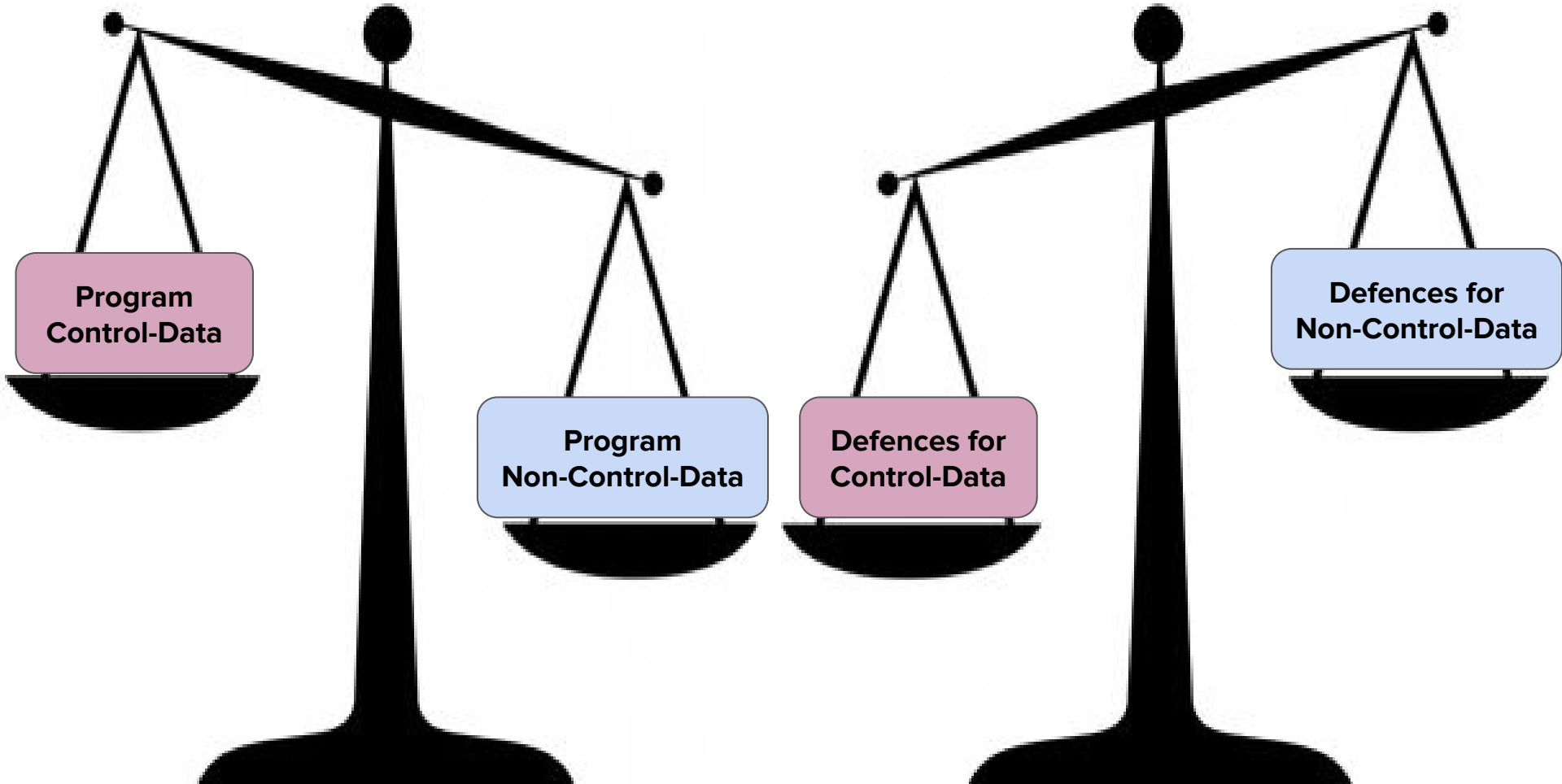
Data is Flowing in the Wind: A Review of Data-Flow Integrity Methods to Overcome Non-Control-Data Attacks

Irene Díez-Franco, Igor Santos
DeustoTech, University of Deusto
irene.diez@deusto.es isantos@deusto.es

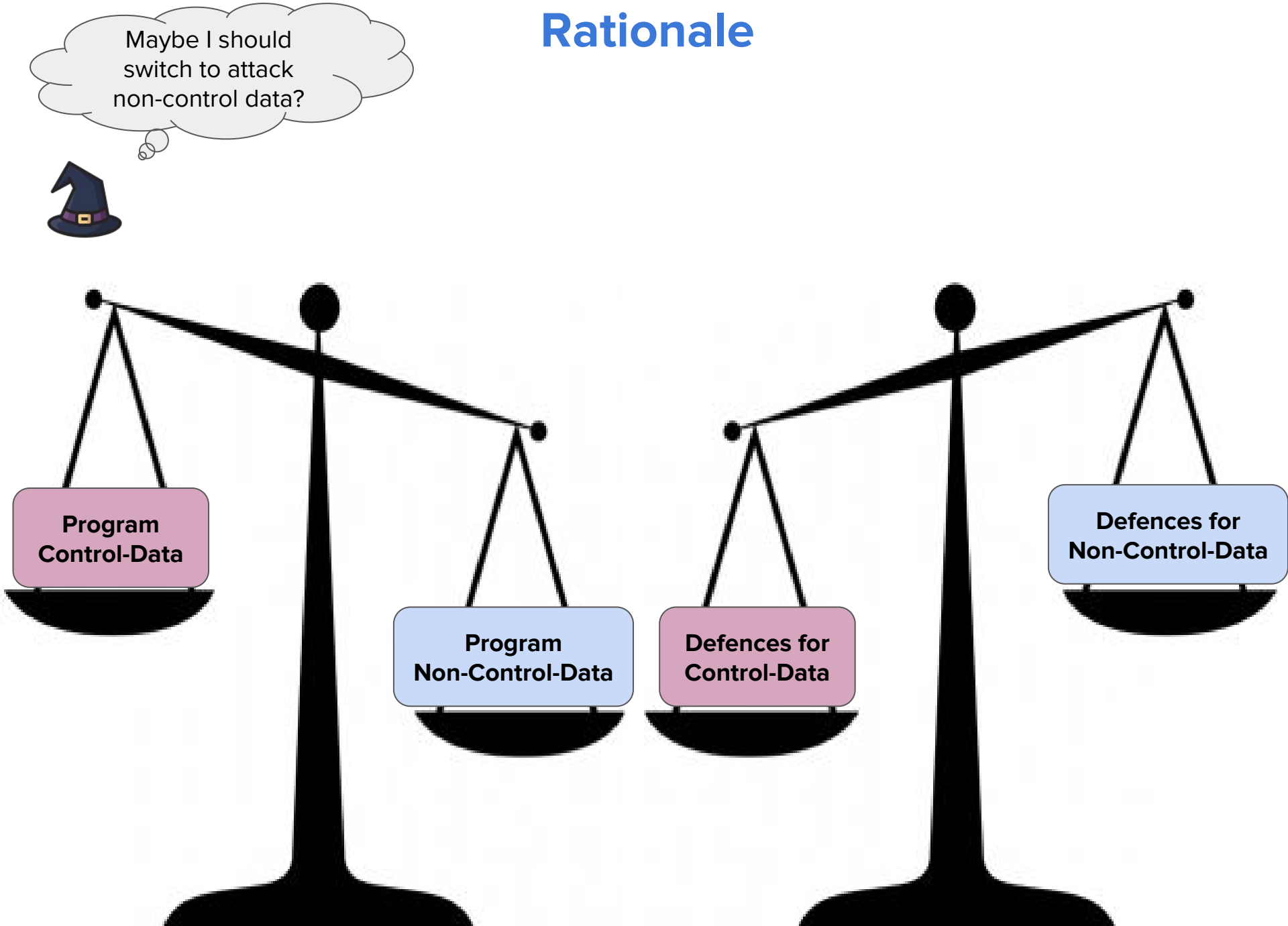
Rationale

Rationale

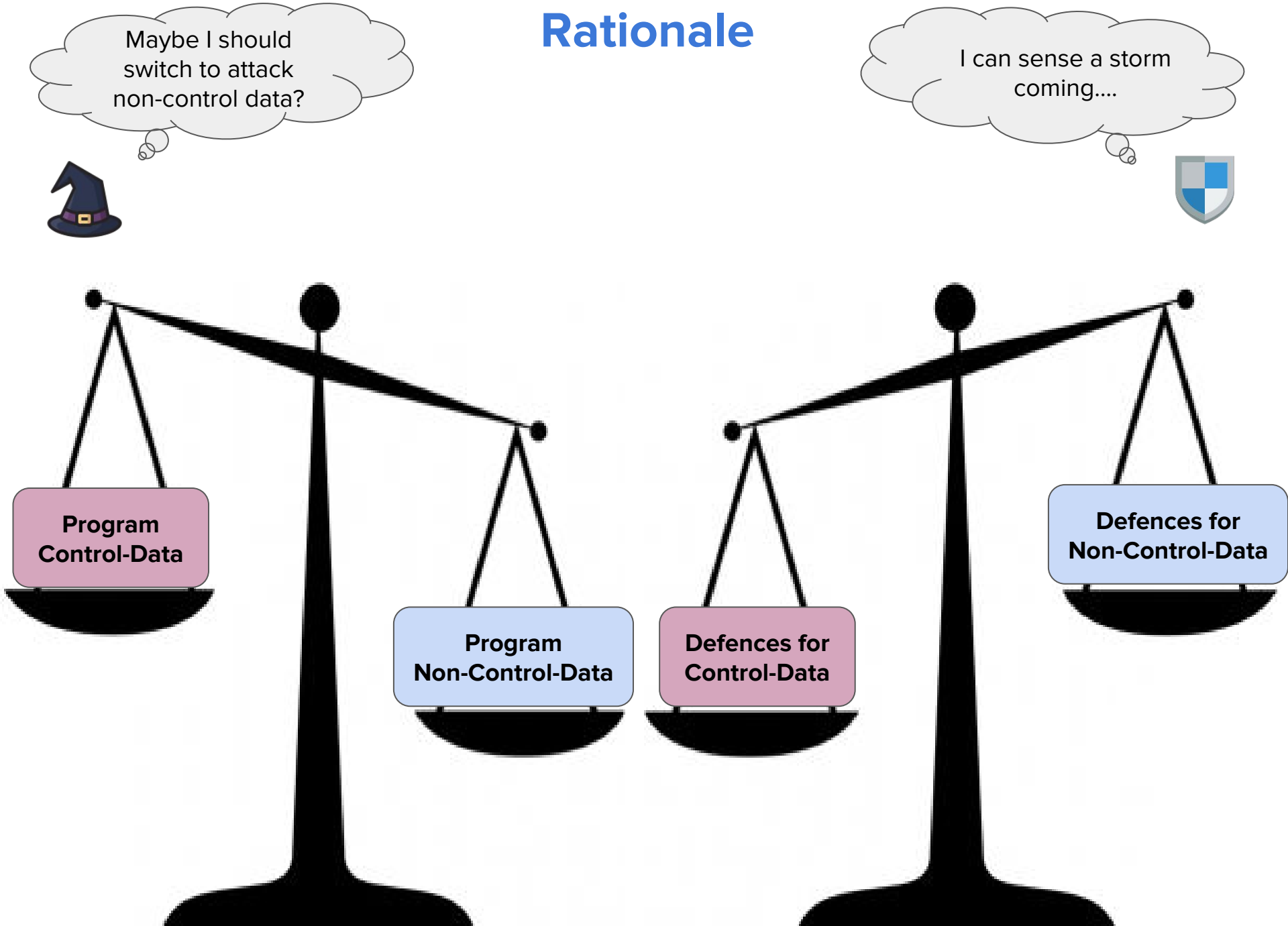
Rationale


Program
Control-Data
Program
Non-Control-Data
Defences for
Control-Data
Defences for
Non-Control-Data

Rationale


Maybe I should switch to attack non-control data?
Program Control-Data
Program Non-Control-Data
Defences for Control-Data
Defences for Non-Control-Data

Rationale

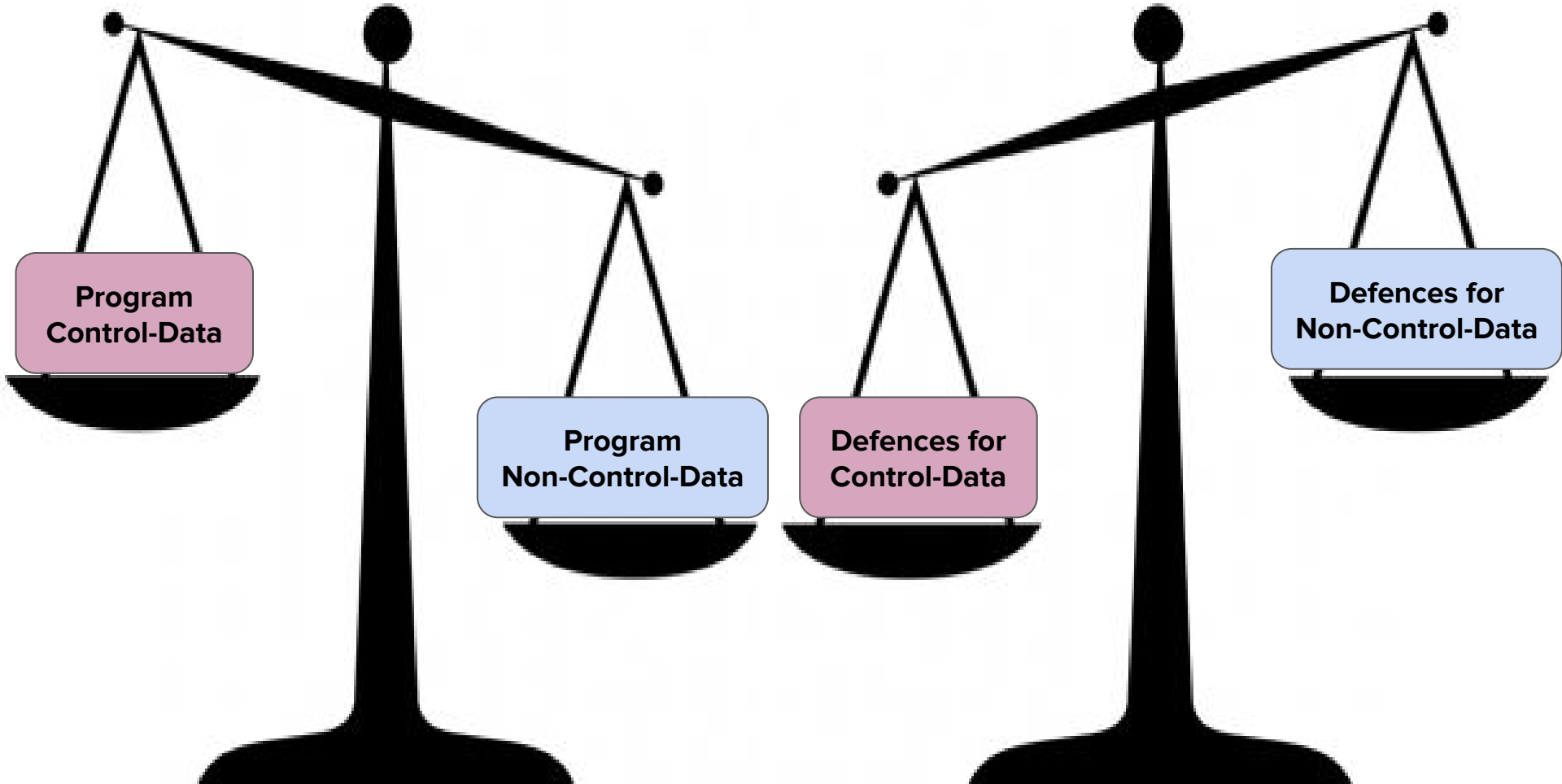

Maybe I should switch to attack non-control data?
I can sense a storm coming....
Program Control-Data
Program Non-Control-Data
Defences for Control-Data
Defences for Non-Control-Data

Rationale

Maybe I should switch to attack non-control data?

I can sense a storm coming....

Non-Control-Data Attacks Are Realistic Threats. Chen et al. Usenix Sec'05



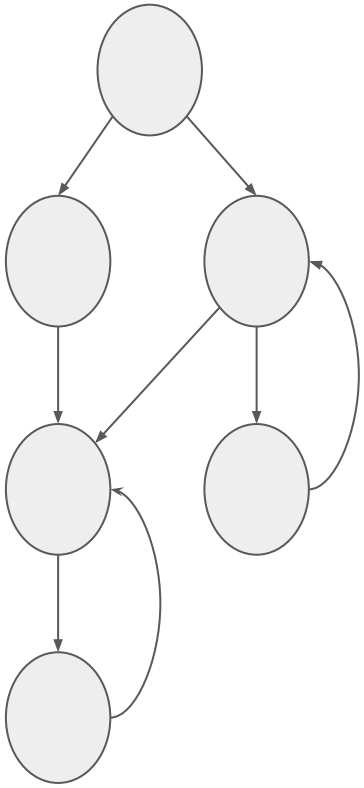
Control Data VS Non-Control-Data Attacks

Control Data VS Non-Control-Data Attacks

Control-Data Attacks

Modify the control-flow of a program

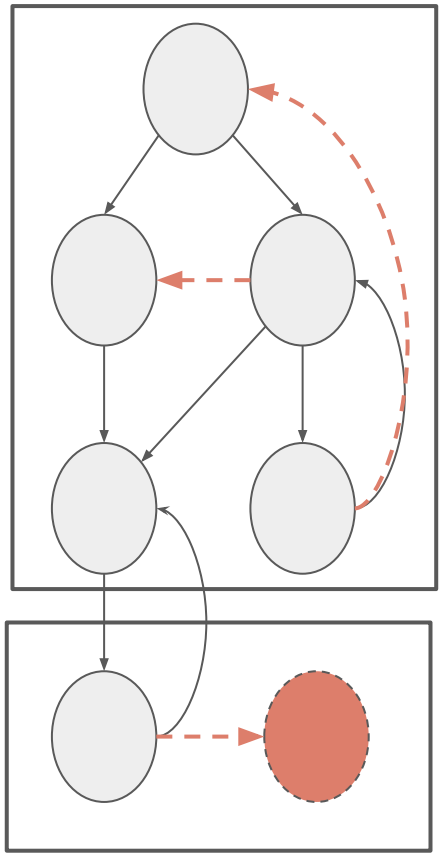
Control Data VS Non-Control-Data Attacks



Control-Data Attacks

Modify the control-flow of a program

Control Data VS Non-Control-Data Attacks



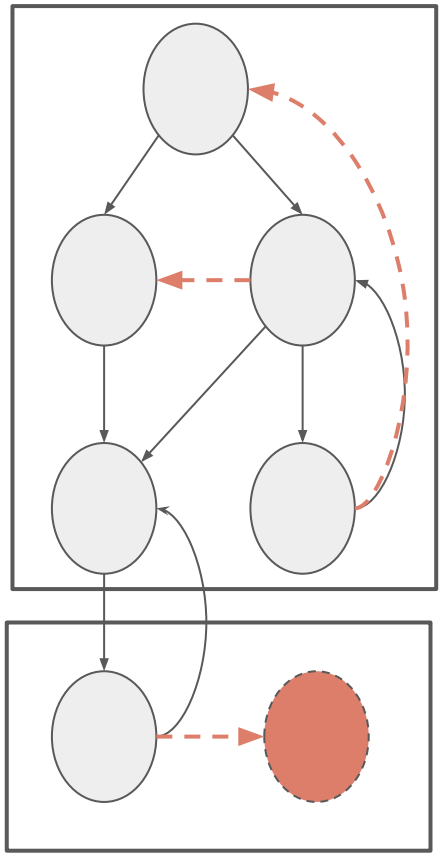
Code reuse

Code injection

Control-Data Attacks

Modify the control-flow of a program

Control Data VS Non-Control-Data Attacks



Code reuse

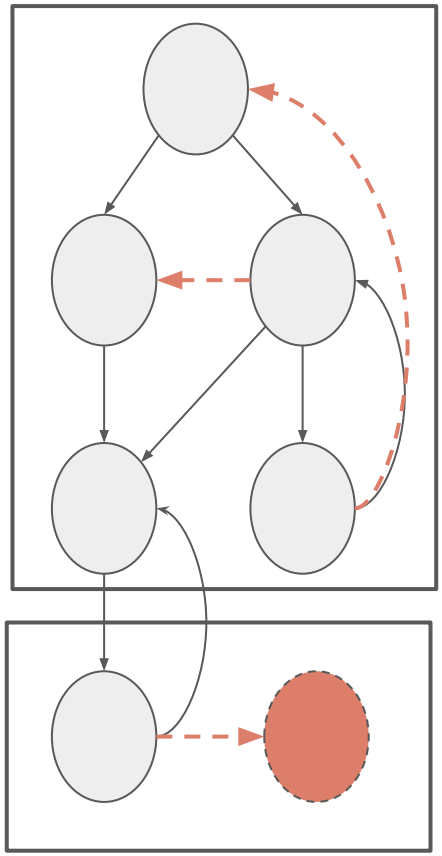
Modify the values of `ret, call, jmp` instructions

Code injection

Control-Data Attacks

Modify the control-flow of a program

Control Data VS Non-Control-Data Attacks



Code reuse

Modify the values of `ret, call, jmp` instructions

Code injection

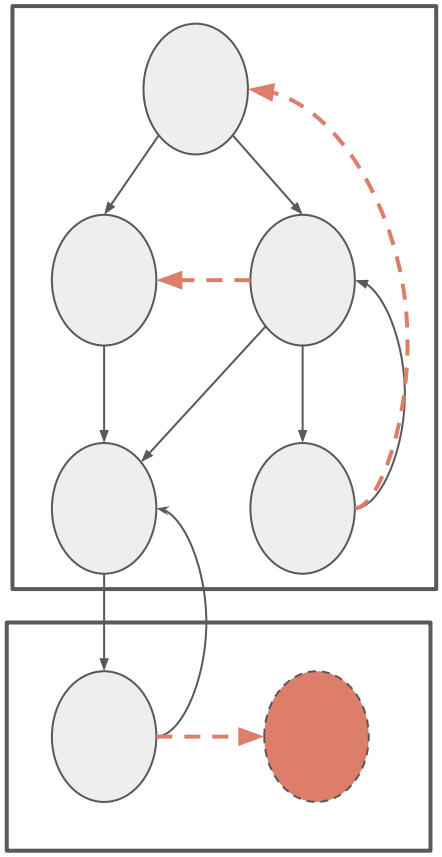
Control-Data Attacks

Modify the control-flow of a program

Non-Control-Data Attacks

Do not affect the control-flow of a program

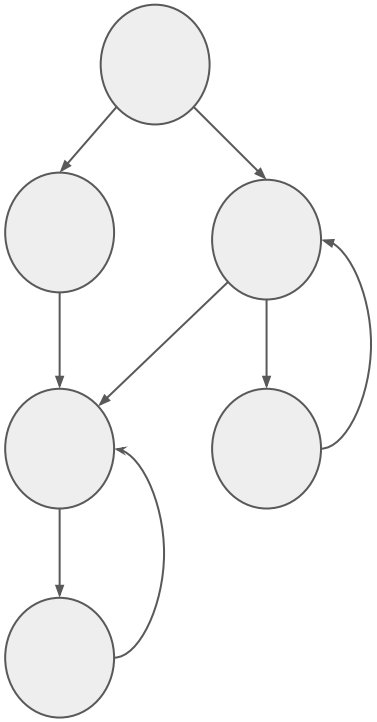
Control Data VS Non-Control-Data Attacks



Code reuse

Modify the values of `ret, call, jmp` instructions

Code injection



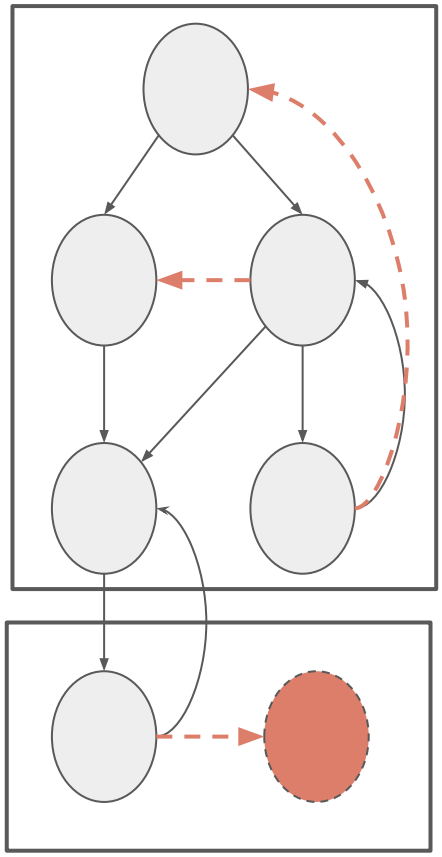
Control-Data Attacks

Modify the control-flow of a program

Non-Control-Data Attacks

Do not affect the control-flow of a program

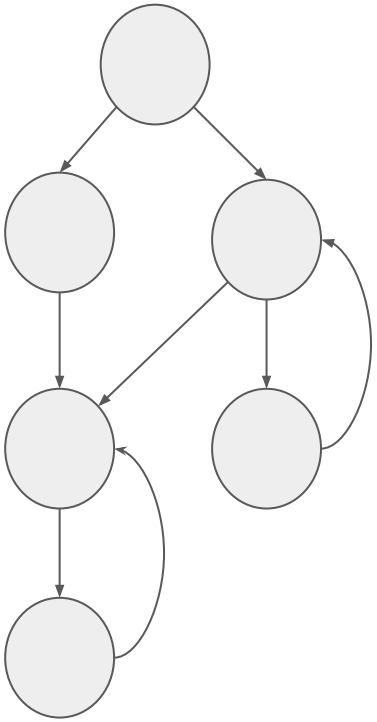
Control Data VS Non-Control-Data Attacks



Code reuse

Modify the values of `ret, call, jmp` instructions

Code injection



Remain invisible to techniques which only focus on control-data

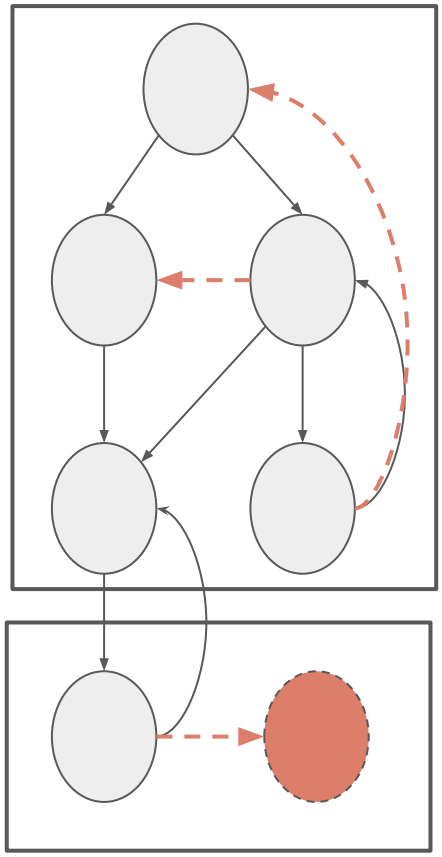
Control-Data Attacks

Modify the control-flow of a program

Non-Control-Data Attacks

Do not affect the control-flow of a program

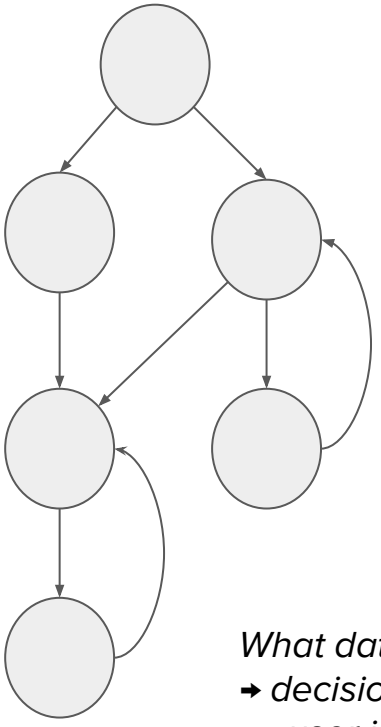
Control Data VS Non-Control-Data Attacks



Code reuse

Modify the values of `ret, call, jmp` instructions

Code injection



Remain invisible to techniques which only focus on control-data

What data do they target?
➔ *decision making data, user input etc.*

Control-Data Attacks

Modify the control-flow of a program

Non-Control-Data Attacks

Do not affect the control-flow of a program

Security-Critical Non-Control Data

Chen et al. Usenix Sec'05
Hu et al. Usenix Sec'15

Security-Critical Non-Control Data

Chen et al. Usenix Sec'05
Hu et al. Usenix Sec'15

Configuration data

User identity data

Decision-making data

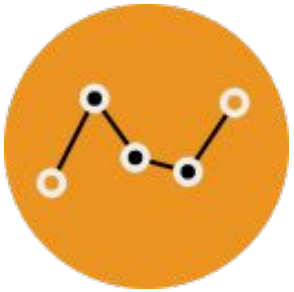
User input data

Security-Critical Non-Control Data

Chen et al. Usenix Sec'05
Hu et al. Usenix Sec'15

Configuration data

User identity data



Decision-making data

User input data

Passwords & private keys

System call parameters

Randomised values

A sample non-control-data attack

A sample non-control-data attack

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
...
int uid = getuid();
pw->pq_uid = uid; // save current uid
// [ format string vulnerability ]
...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid); // drop root priv
}
```

From wu-ftpd web server.
Hu et al. Usenix Sec'15

A sample non-control-data attack

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
...
int uid = getuid();
pw->pq_uid = uid; // save current uid
// [ format string vulnerability ]
...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid); // drop root priv
}
```

```
int uid = getuid();
pw->pq_uid = uid; // save current uid
// [ format string vulnerability    ]
// [ exploit it to overwrite 'uid'  ]
// [ pw->pq_uid = 0;                ]
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid); // avoid priv. drop
}
```

From wu-ftpd web server.
Hu et al. Usenix Sec'15

A sample non-control-data attack

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
...
int uid = getuid();
pw->pq_uid = uid; // save current uid
// [ format string vulnerability ]
...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid); // drop root priv
}
```

From wu-ftpd web server.
Hu et al. Usenix Sec'15

```
int uid = getuid();
pw->pq_uid = uid; // save current uid
// [ format string vulnerability    ]
// [ exploit it to overwrite 'uid'  ]
// [ pw->pq_uid = 0;                ]
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid); // avoid priv. drop
}
```

Circumvents Control Flow Integrity?

Data-Flow Stitching

Hu et al. Usenix Sec'15

Data-Flow Stitching

Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```

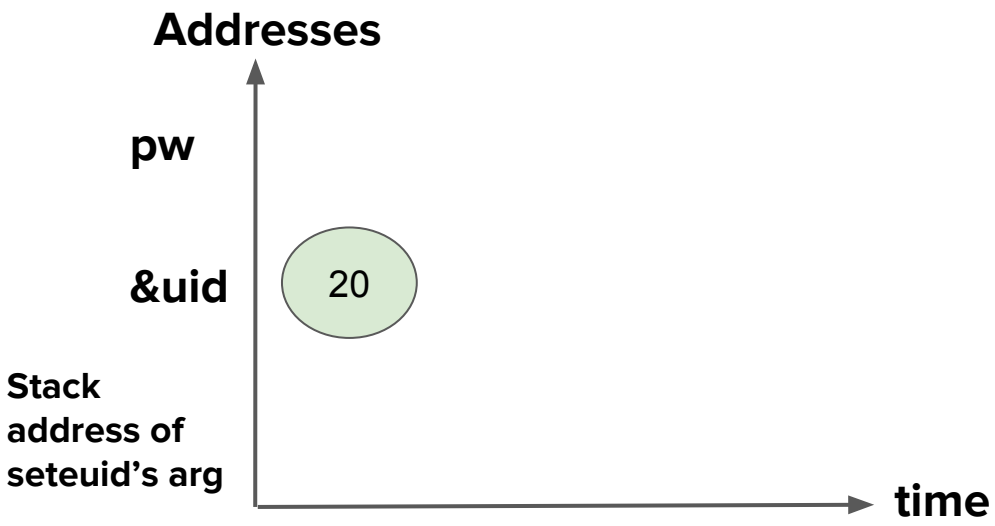
Data-Flow Stitching

Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```



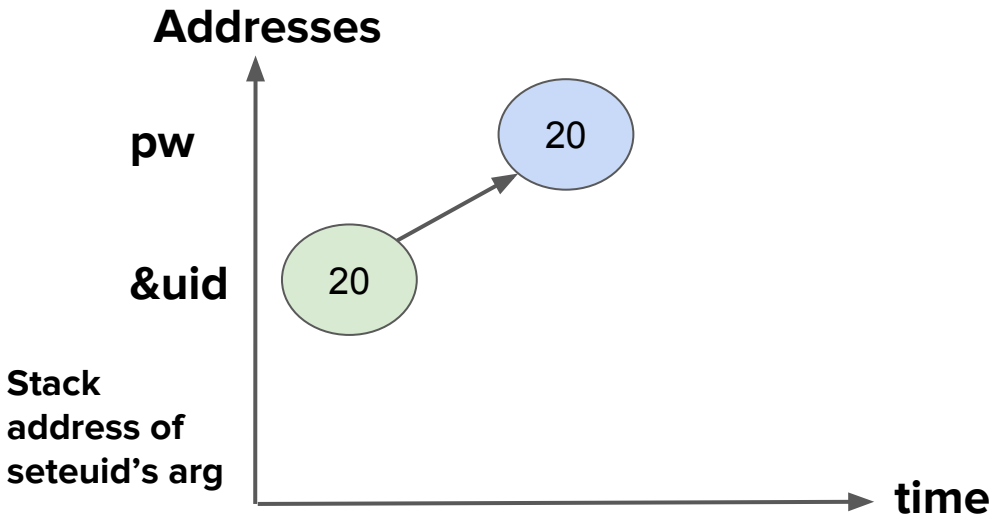
Data-Flow Stitching

Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```



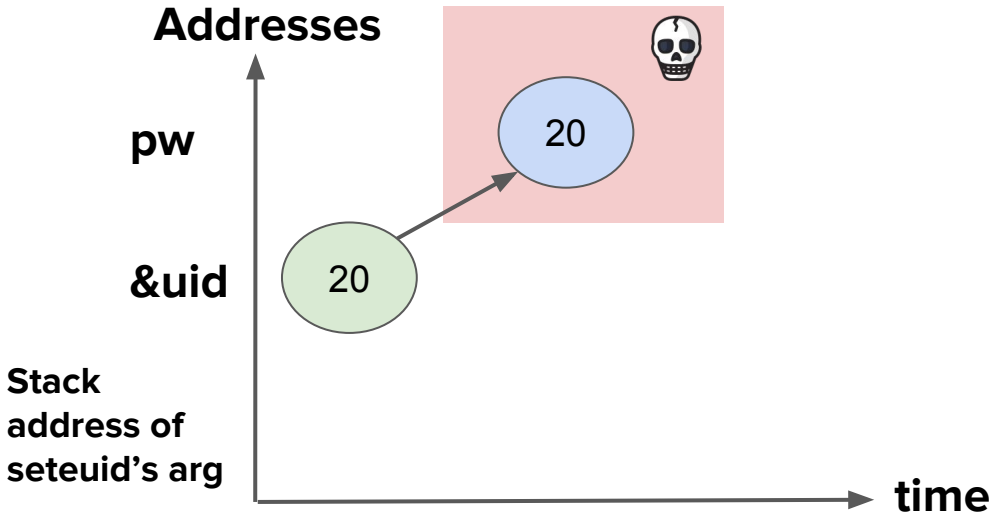
Data-Flow Stitching

Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```



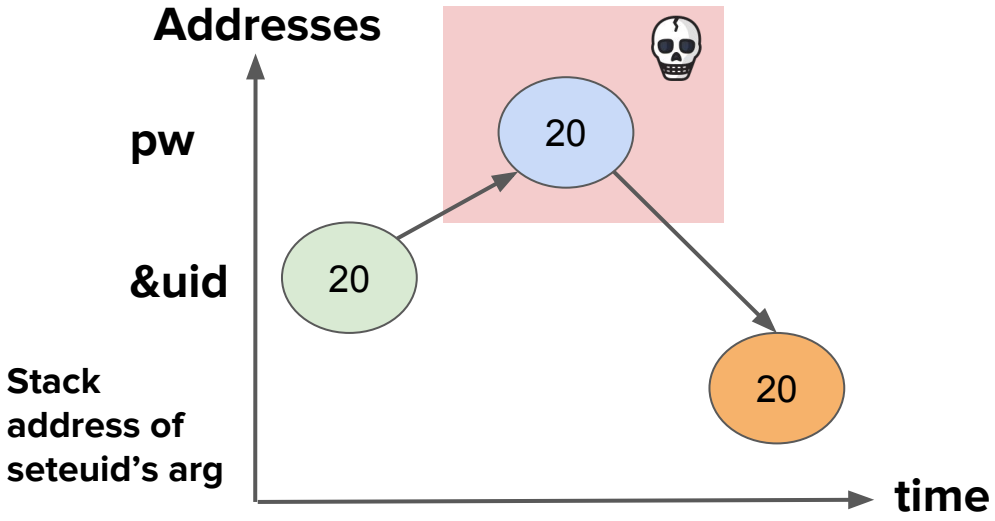
Data-Flow Stitching

Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```



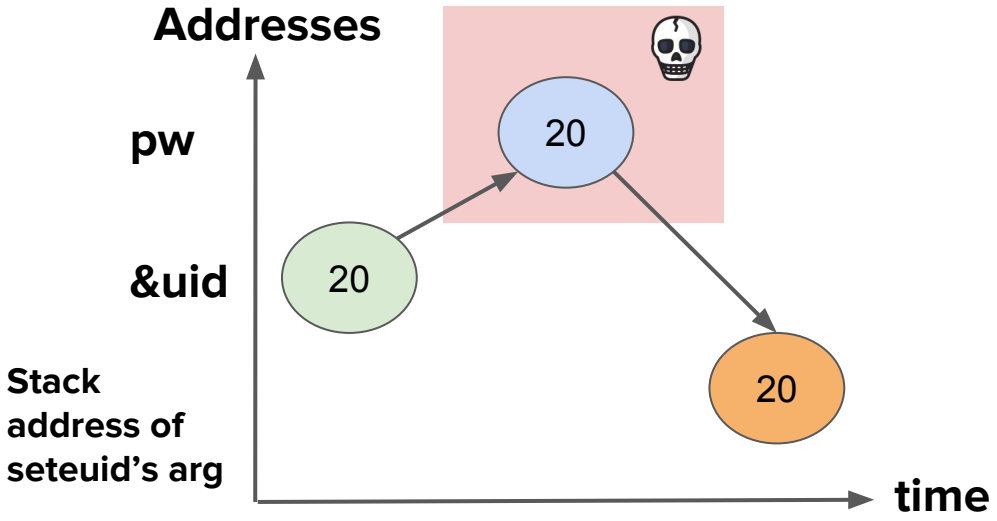
Data-Flow Stitching

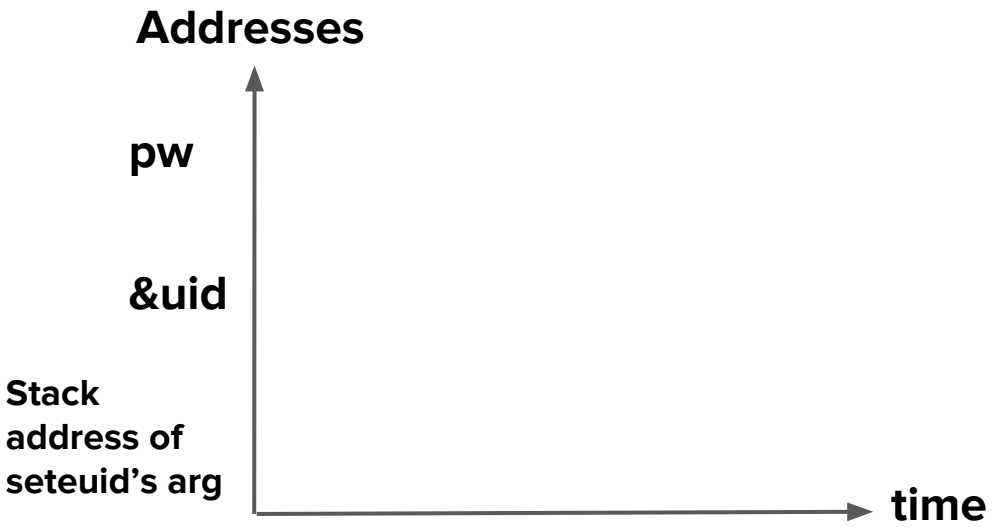
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





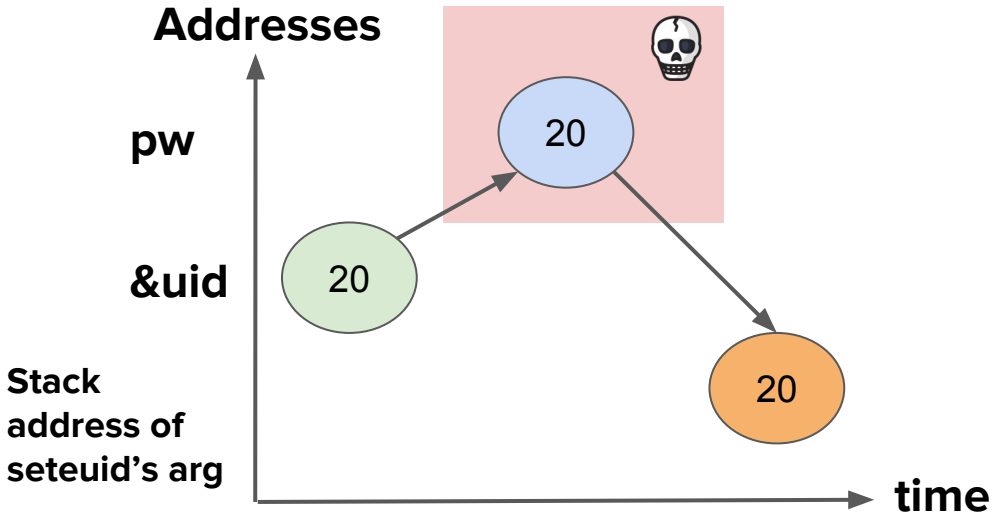
Data-Flow Stitching

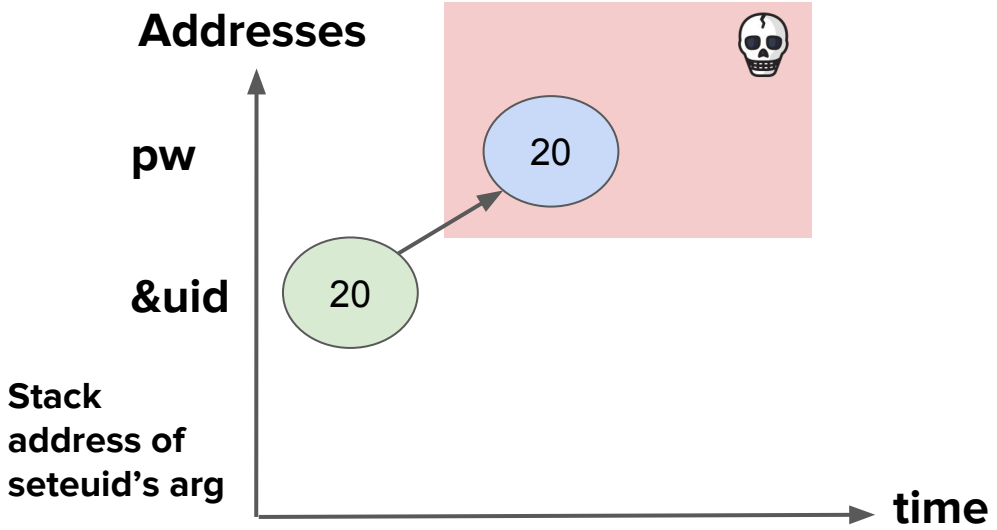
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





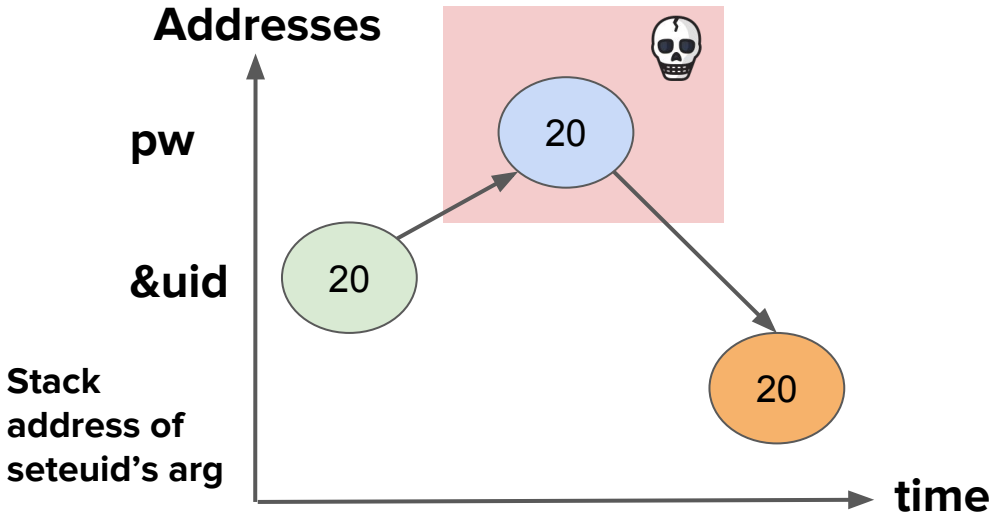
Data-Flow Stitching

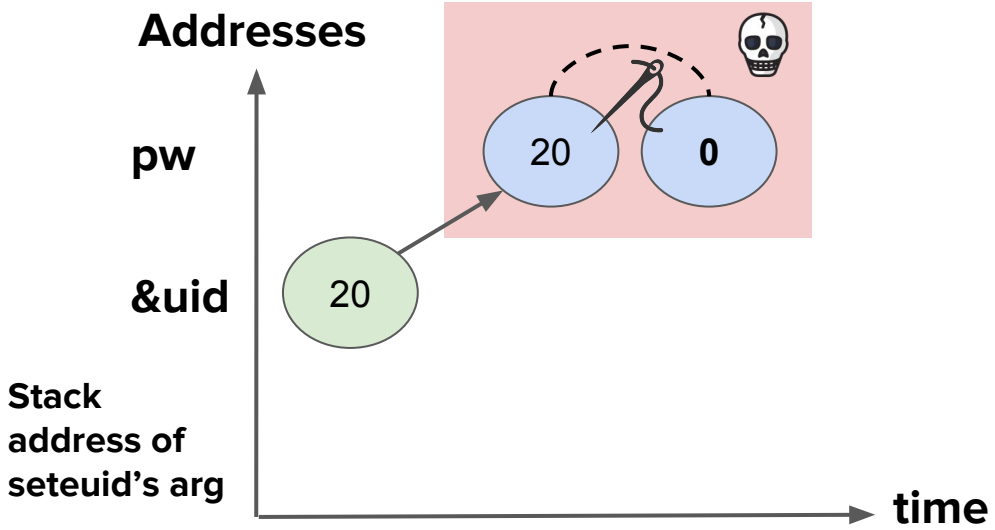
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





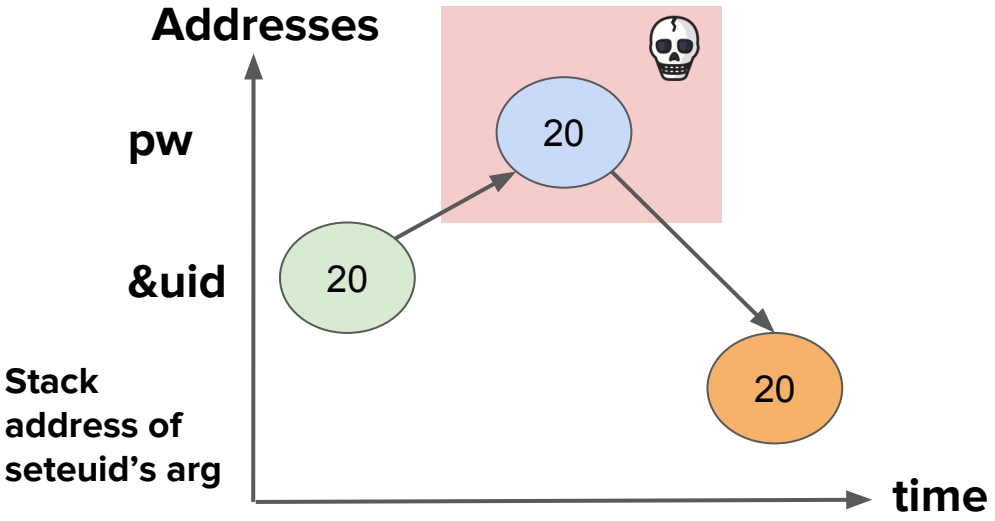
Data-Flow Stitching

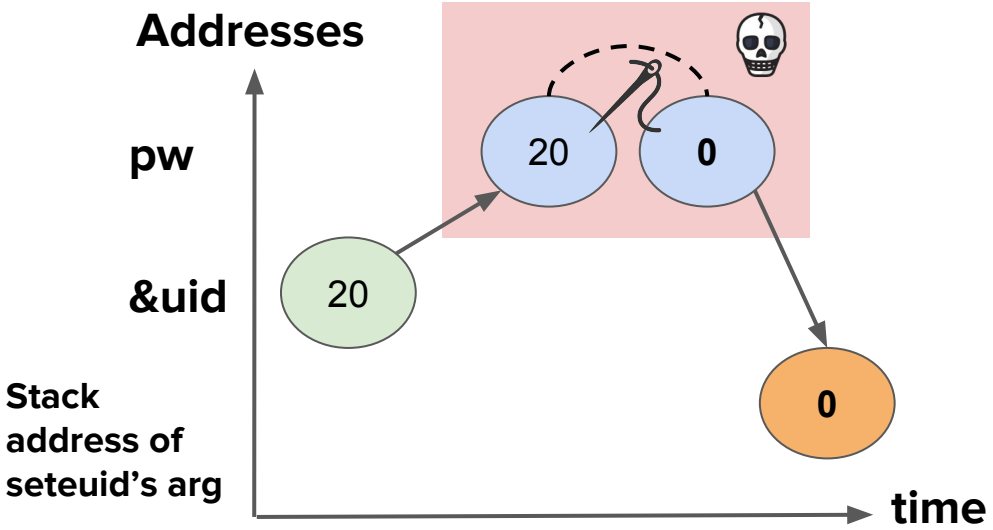
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





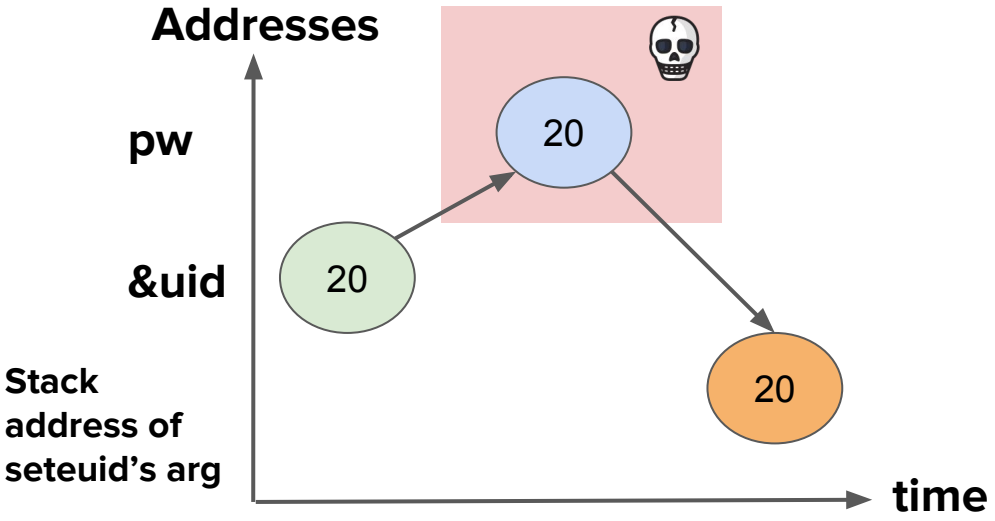
Data-Flow Stitching

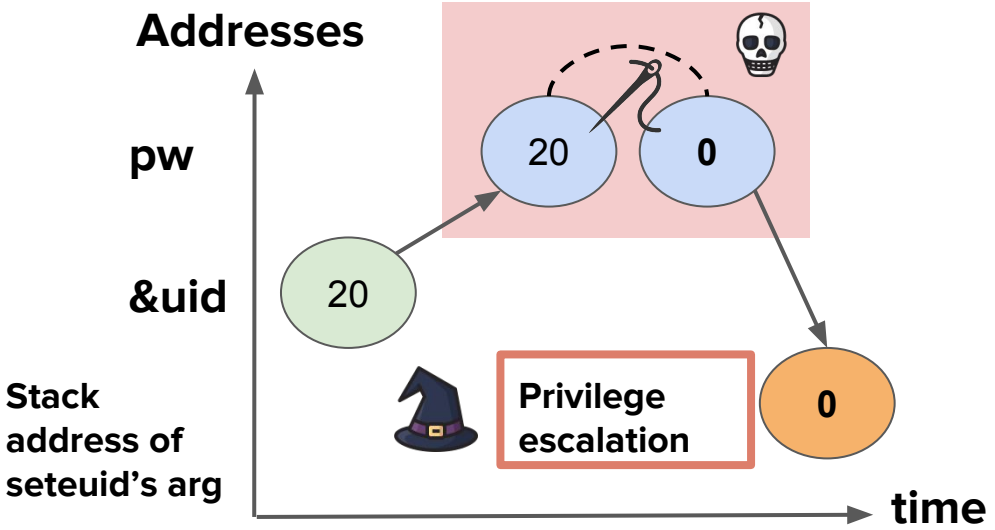
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





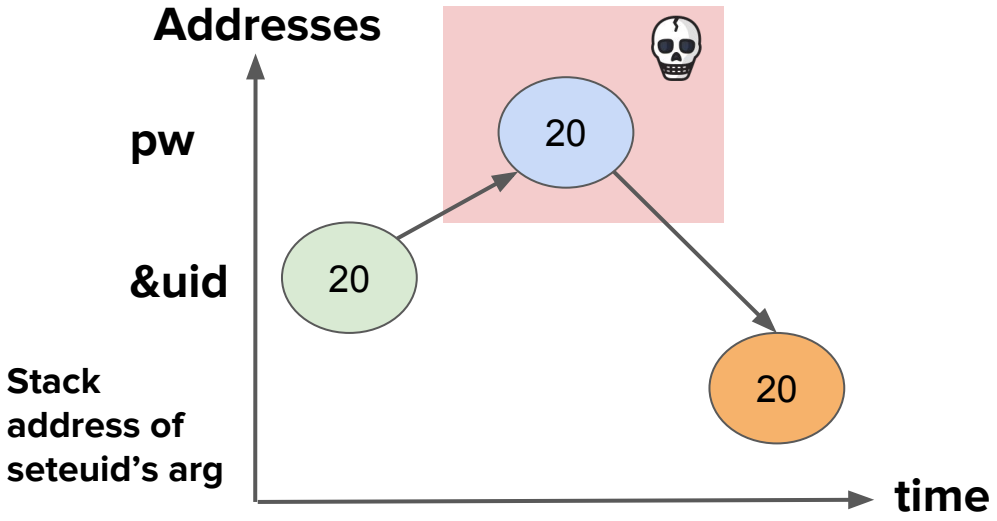
Data-Flow Stitching

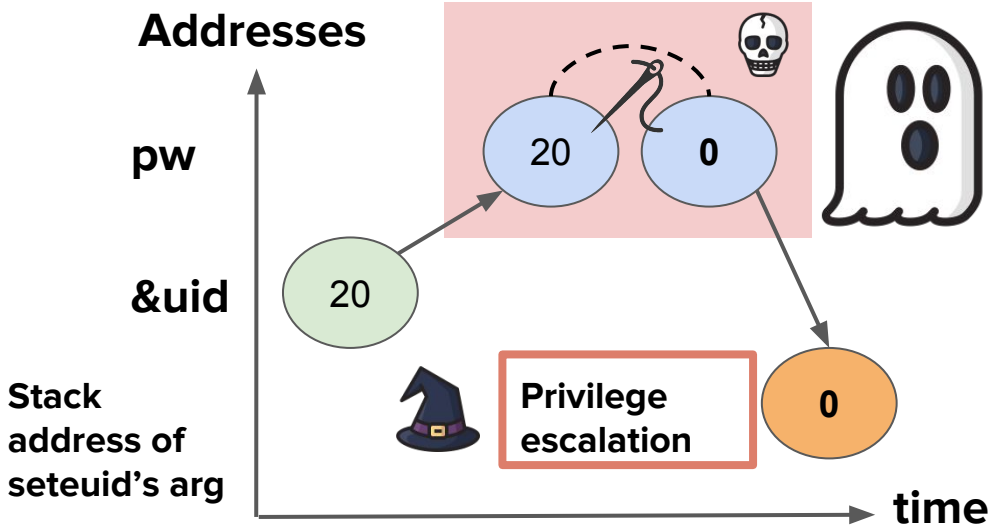
Hu et al. Usenix Sec'15

```
struct passwd
{
  uid_t pw_uid;
  ...
} *pw;
…

int uid = getuid();
pw->pq_uid = uid;
// [ format string vulnerability ]

...
void passive(void)
{
  …
  setuid(0); // become root
  …
  seteuid(pw->pw_uid);
}
```





Data-Oriented Programming (DOP)

Hu et al. Oakland'16

Data-Oriented Programming (DOP)

Hu et al. Oakland'16

Code-reuse attacks with Control-Data

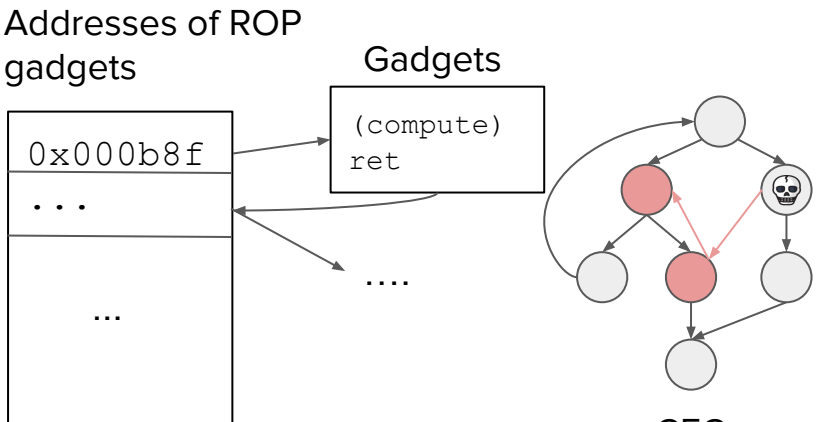
ROP, JOP, SROP

Data-Oriented Programming (DOP)

Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP



Data-Oriented Programming (DOP)

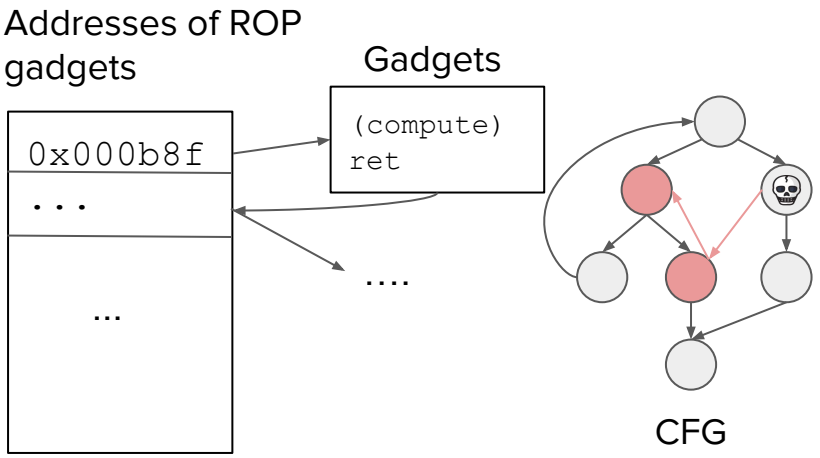
Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP

Requirements:

1. Classic gadgets
2. The gadgets must be chained



Data-Oriented Programming (DOP)

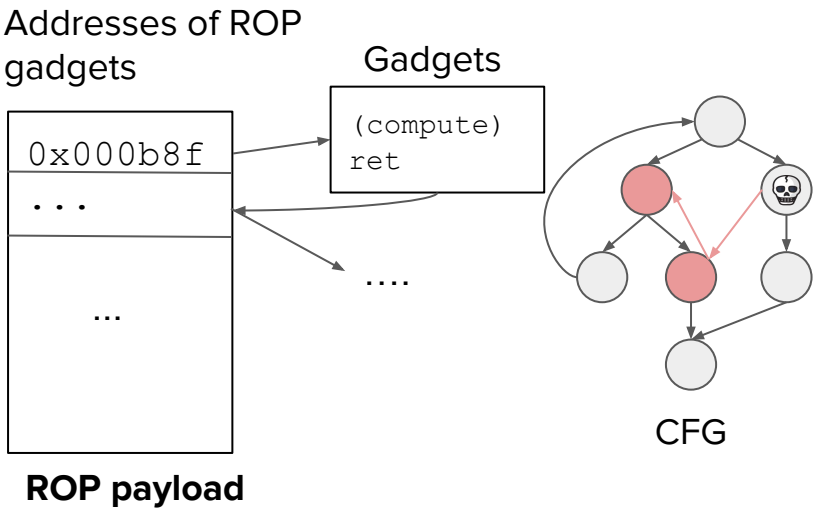
Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP

Requirements:

1. Classic gadgets
2. The gadgets must be chained



Code-reuse attacks with Non-Control-Data

DOP

Data-Oriented Programming (DOP)

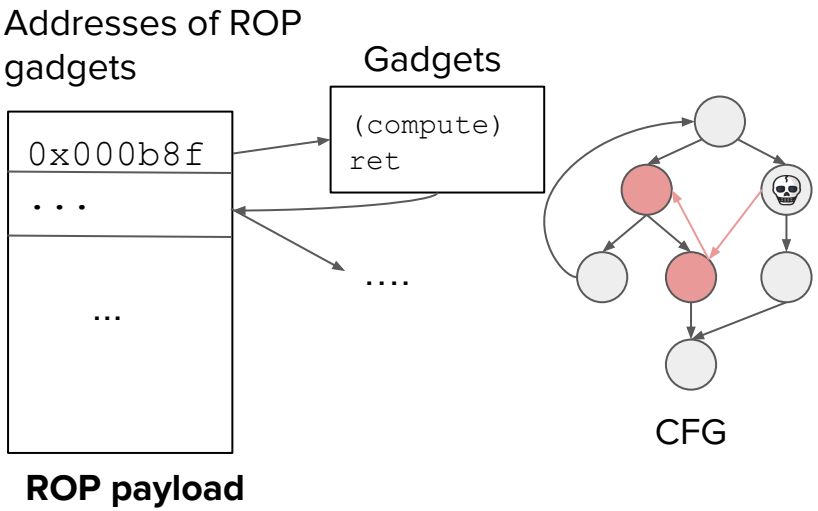
Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP

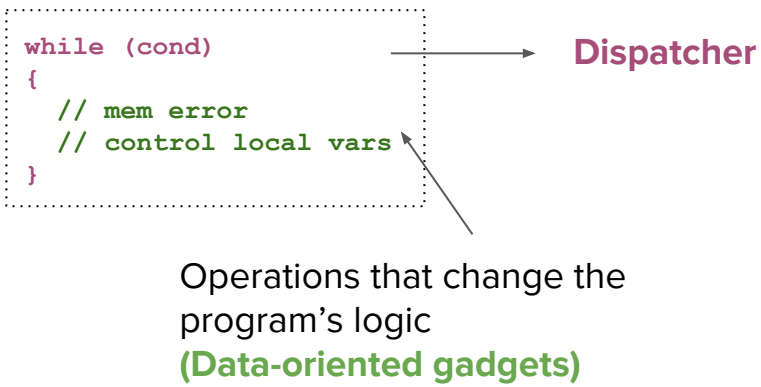
Requirements:

1. Classic gadgets
2. The gadgets must be chained



Code-reuse attacks with Non-Control-Data

DOP



Data-Oriented Programming (DOP)

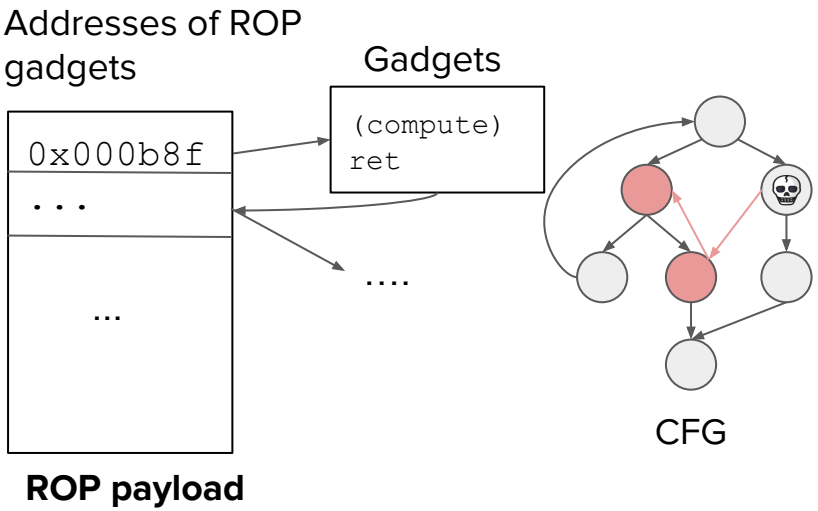
Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP

Requirements:

1. Classic gadgets
2. The gadgets must be chained

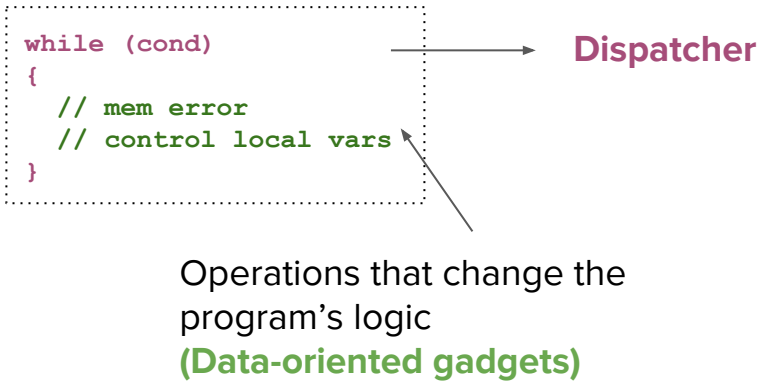


Code-reuse attacks with Non-Control-Data

DOP

Requirements:

1. Data-oriented gadgets
2. Gadget dispatcher



Data-Oriented Programming (DOP)

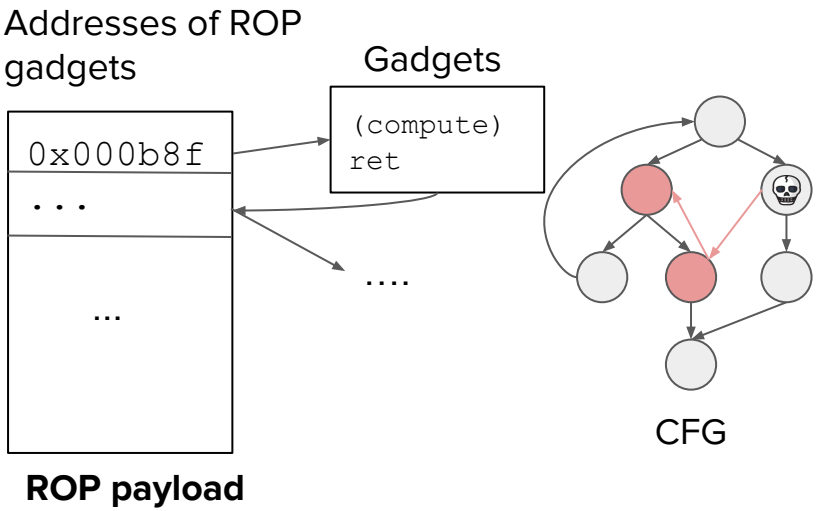
Hu et al. Oakland'16

Code-reuse attacks with Control-Data

ROP, JOP, SROP

Requirements:

1. Classic gadgets
2. The gadgets must be chained

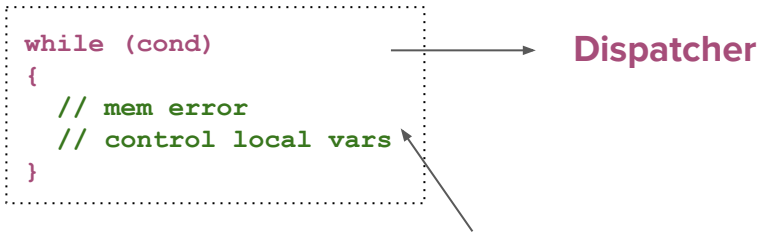


Code-reuse attacks with Non-Control-Data

DOP

Requirements:

1. Data-oriented gadgets
2. Gadget dispatcher
3. **Must follow the legitimate execution path**



Operations that change the program's logic
(Data-oriented gadgets)

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

1 - Offline: DFG computation

Data-Flow Integrity (DFI)

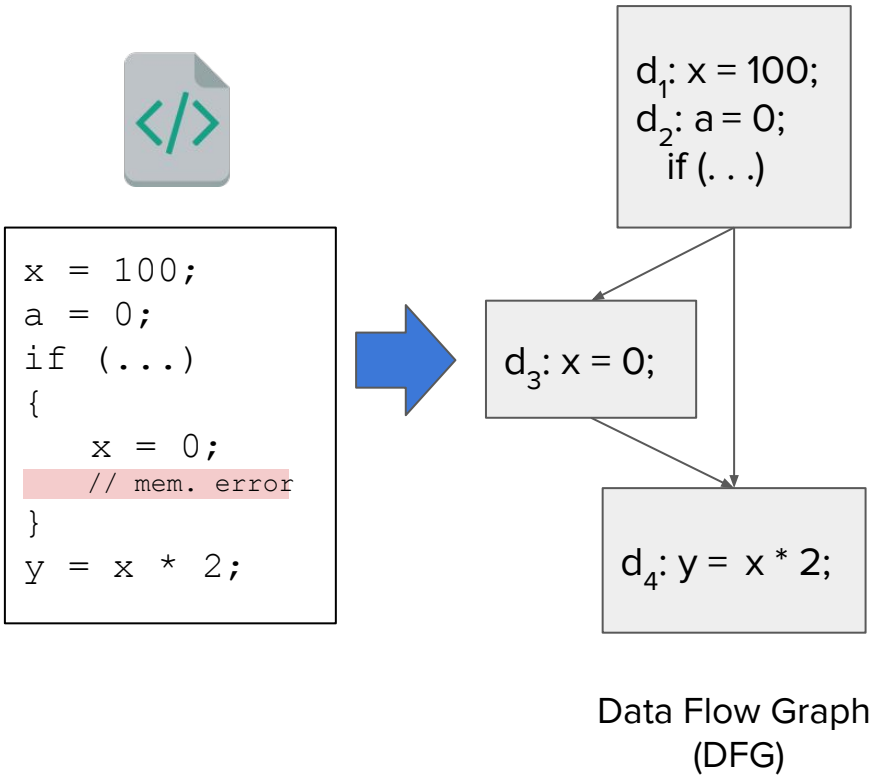
Castro et al. OSDI'06

```
x = 100;
a = 0;
if (...)
{
    x = 0;
    // mem. error
}
y = x * 2;
```

1 - Offline: DFG computation

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

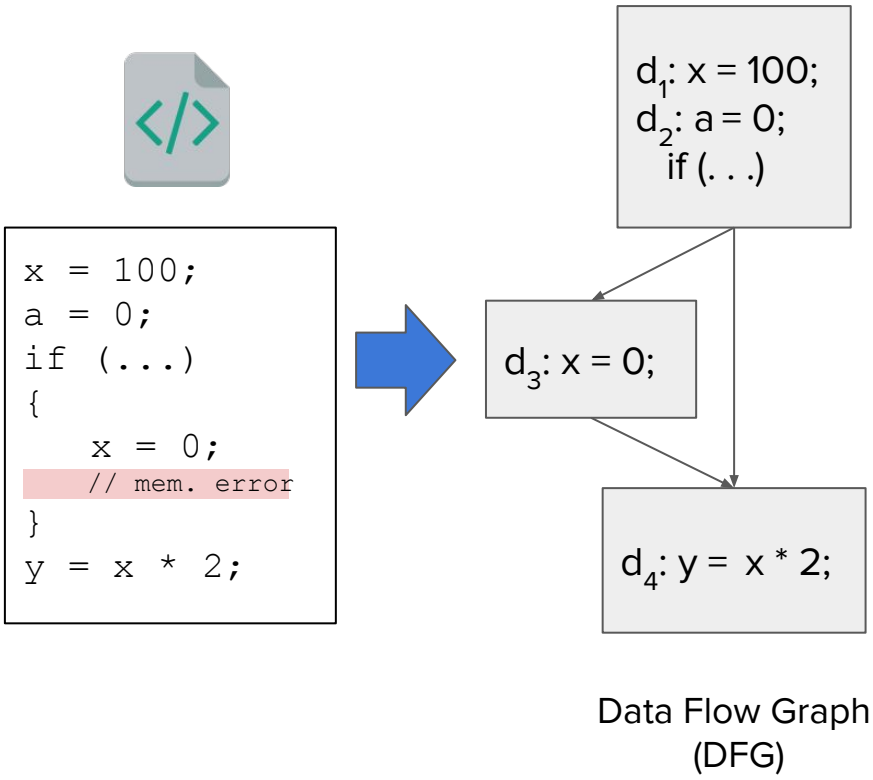


Data Flow Graph
(DFG)

1 - Offline: DFG computation

Data-Flow Integrity (DFI)

Castro et al. OSDI'06



Data Flow Graph (DFG)

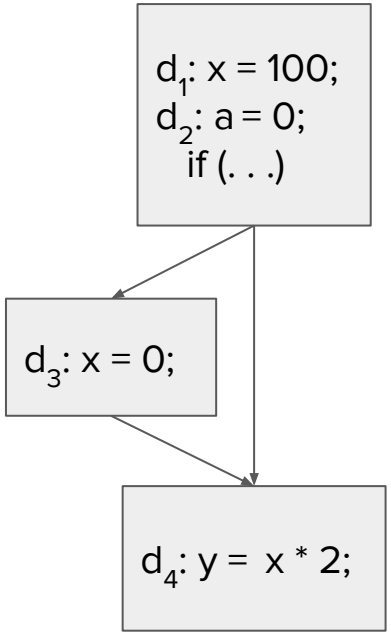
1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

```
x = 100;
a = 0;
if (...)
{
    x = 0;
    // mem. error
}
y = x * 2;
```



Data Flow Graph (DFG)

$x \rightarrow \{ d_1, d_3 \}$
$a \rightarrow \{ d_2 \}$
$y \rightarrow \{ d_4 \}$

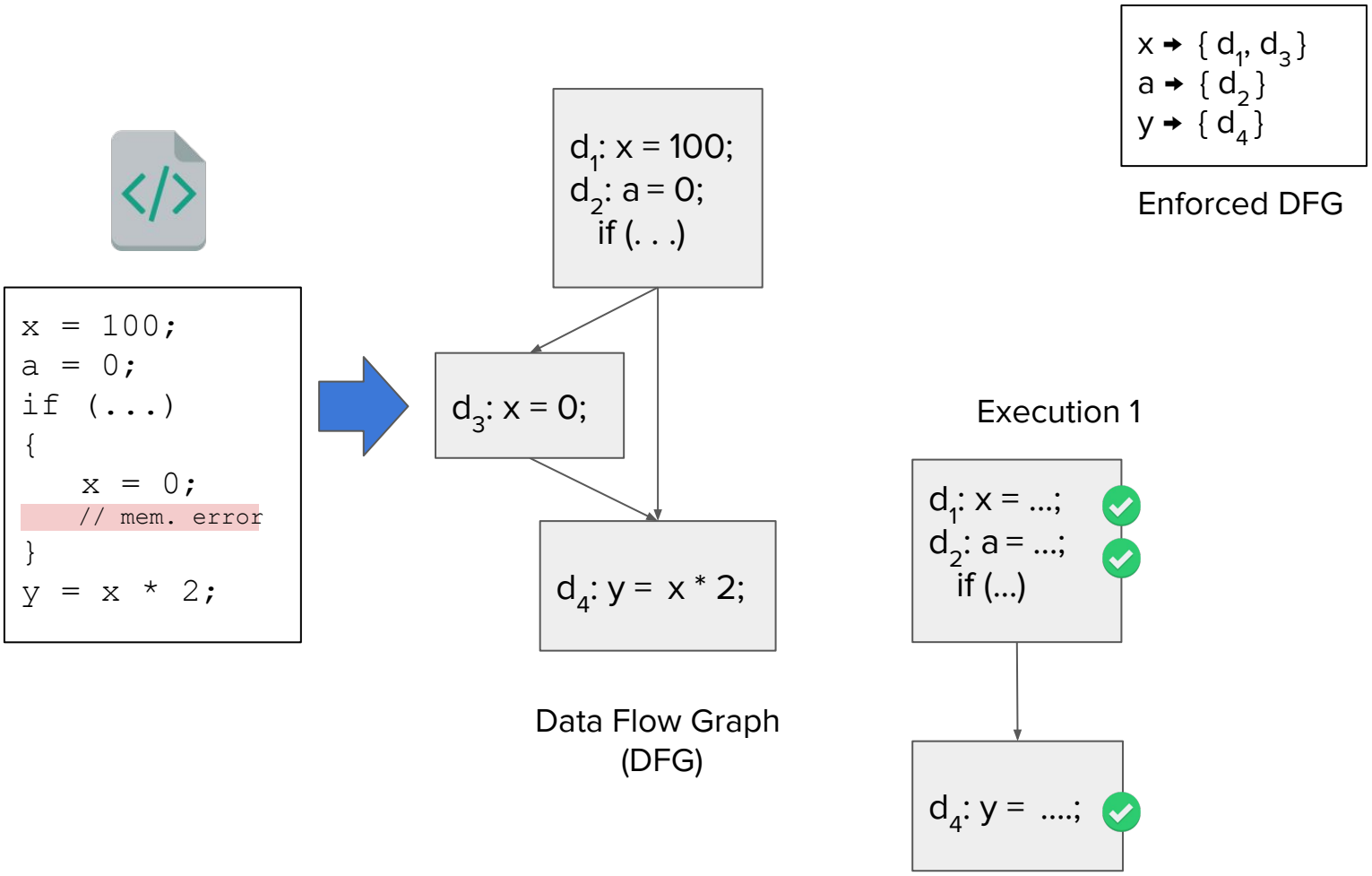
Enforced DFG

1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

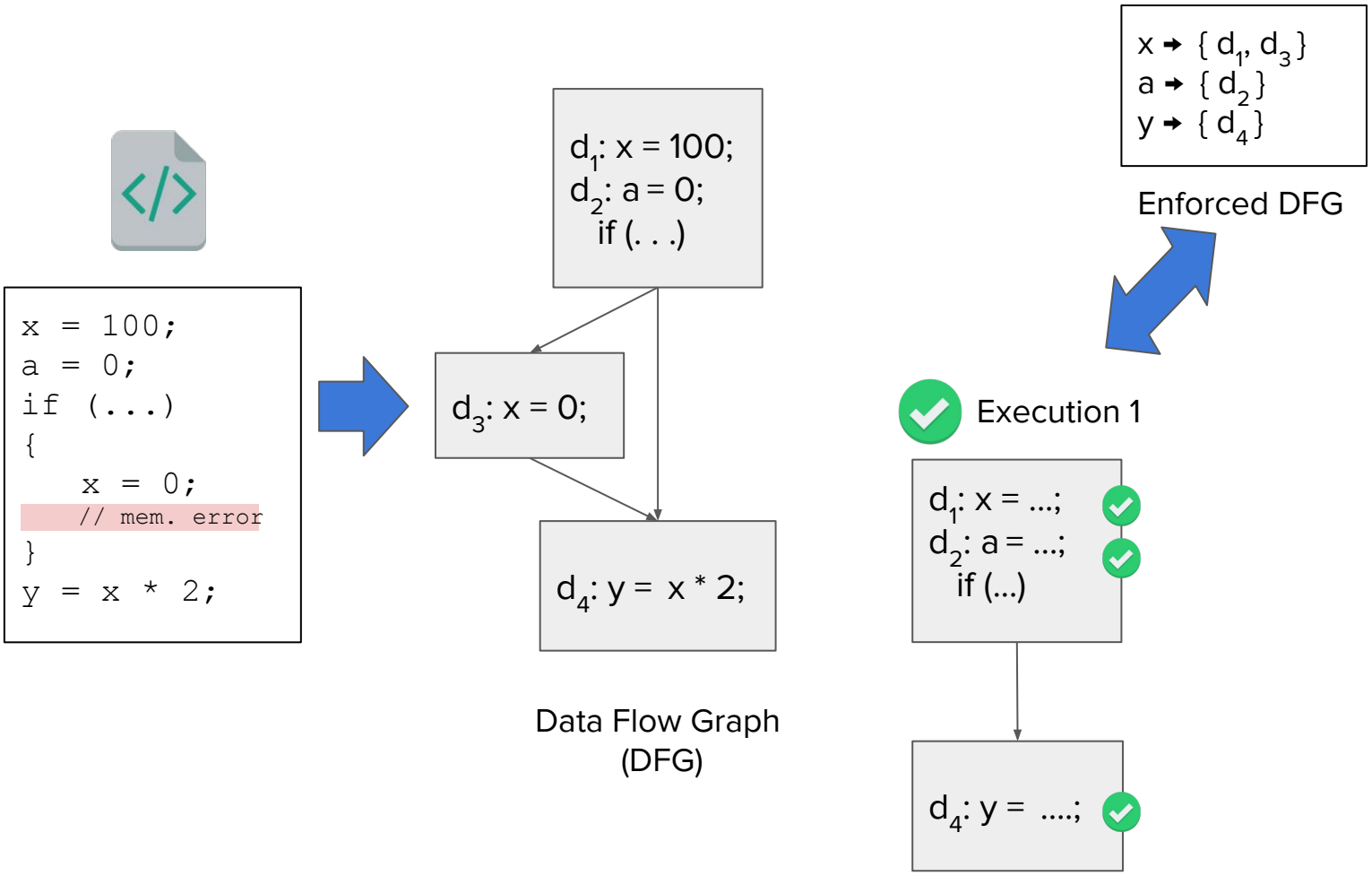


1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

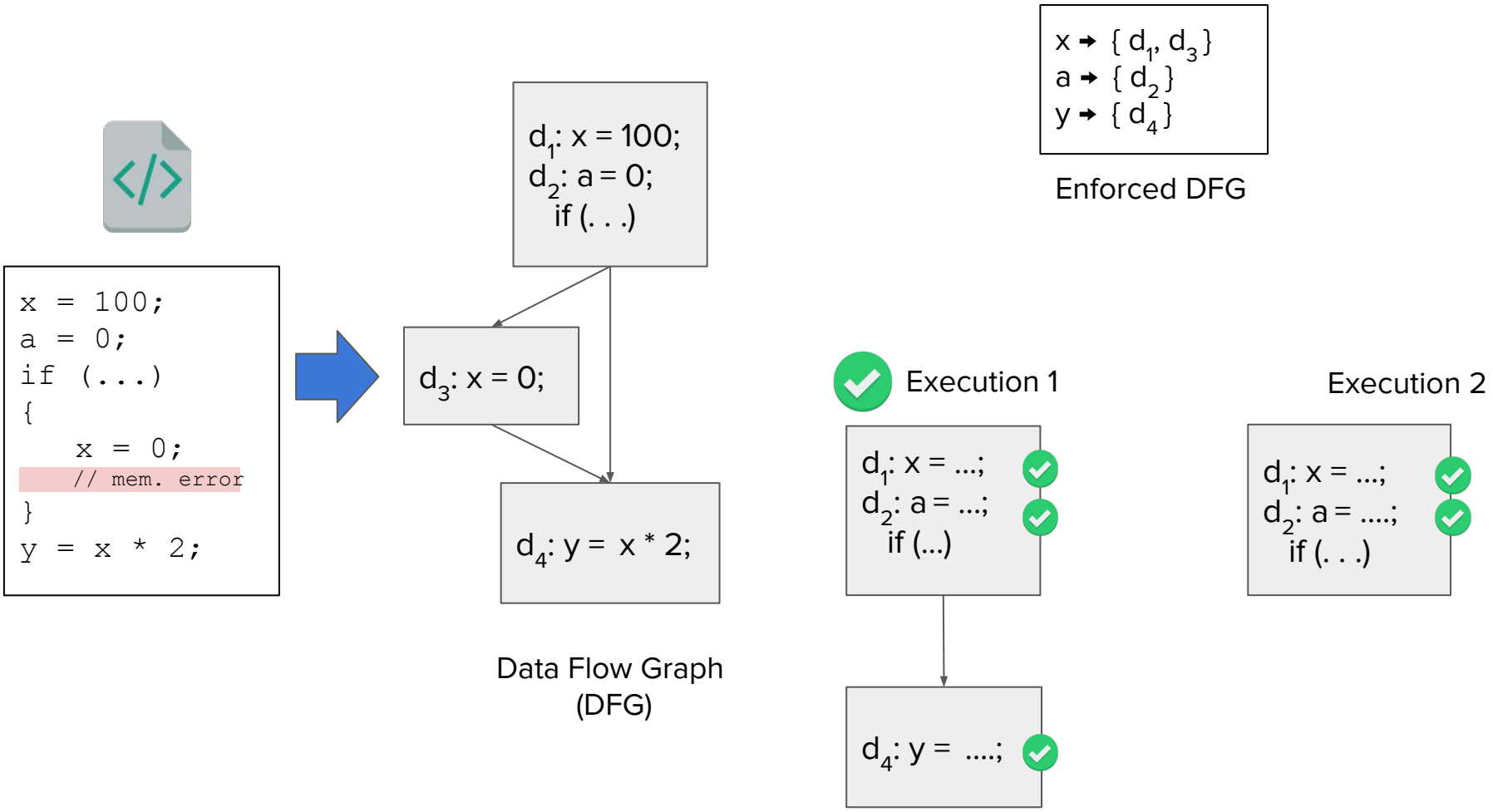


1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

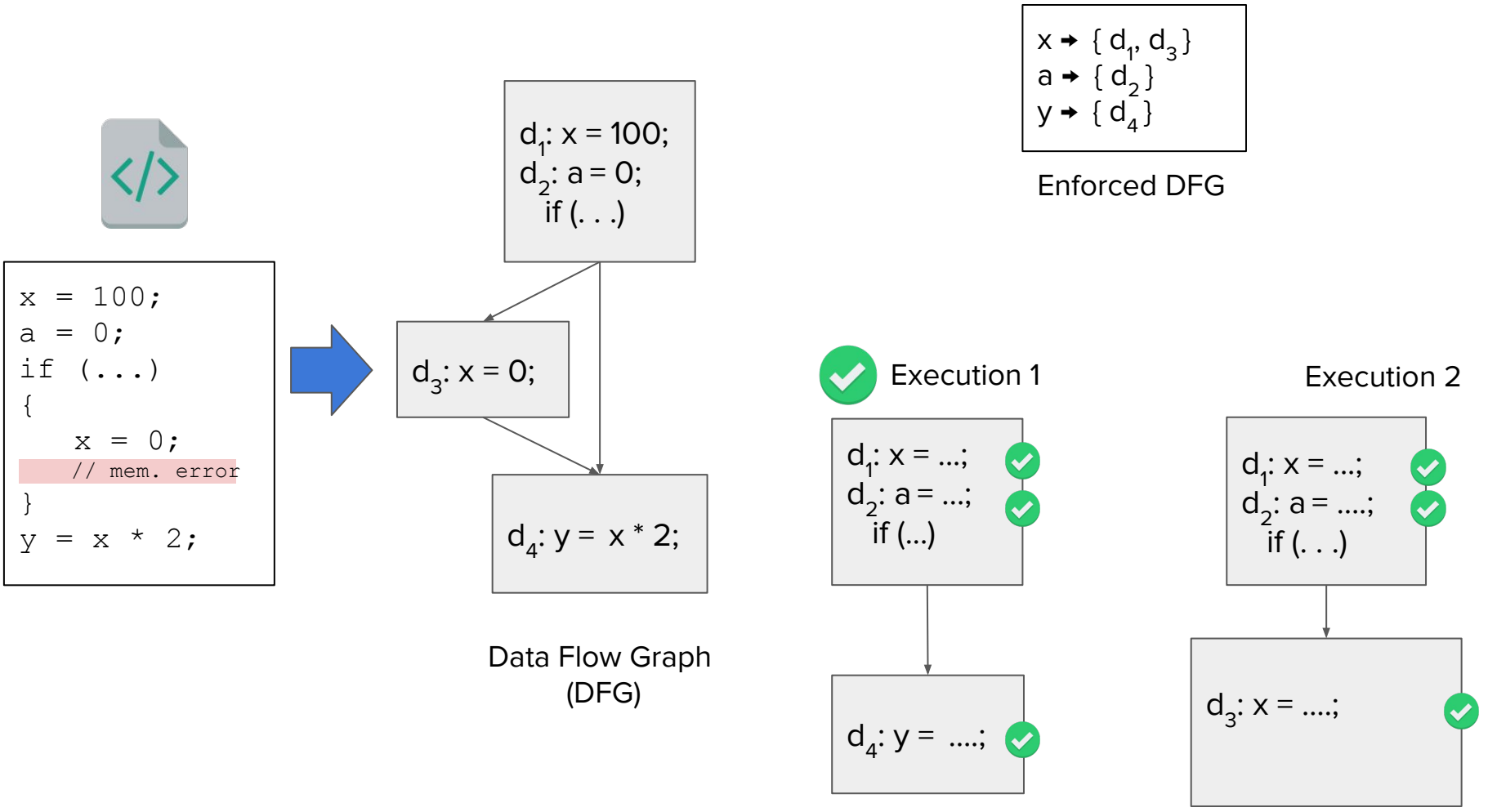


1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06

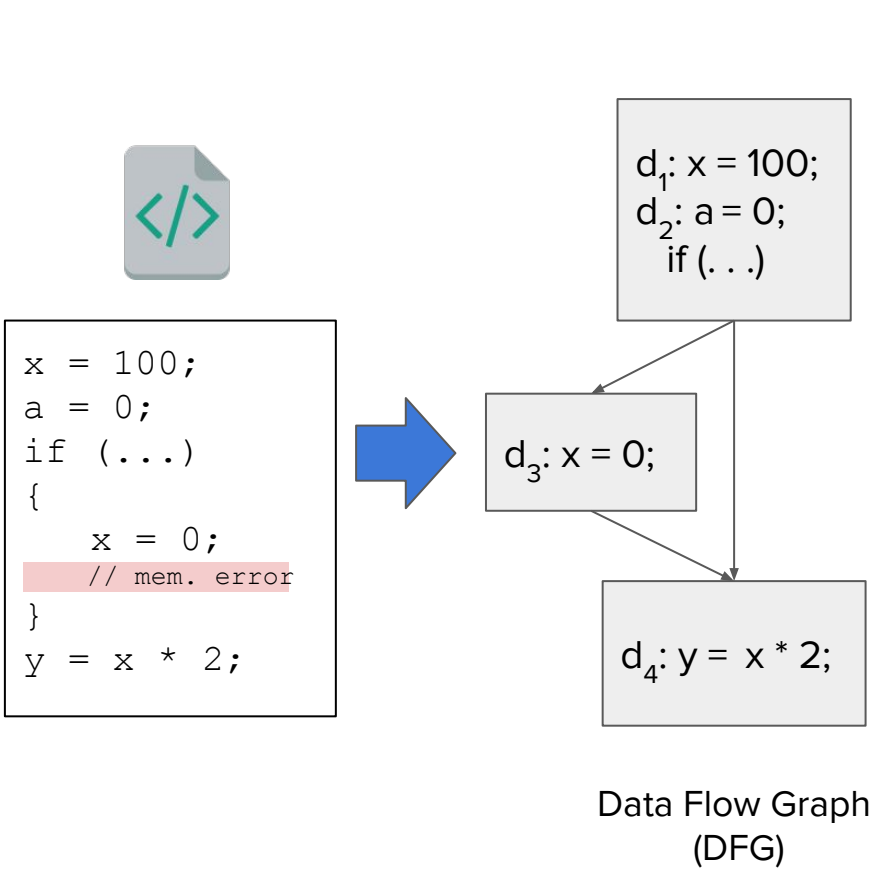


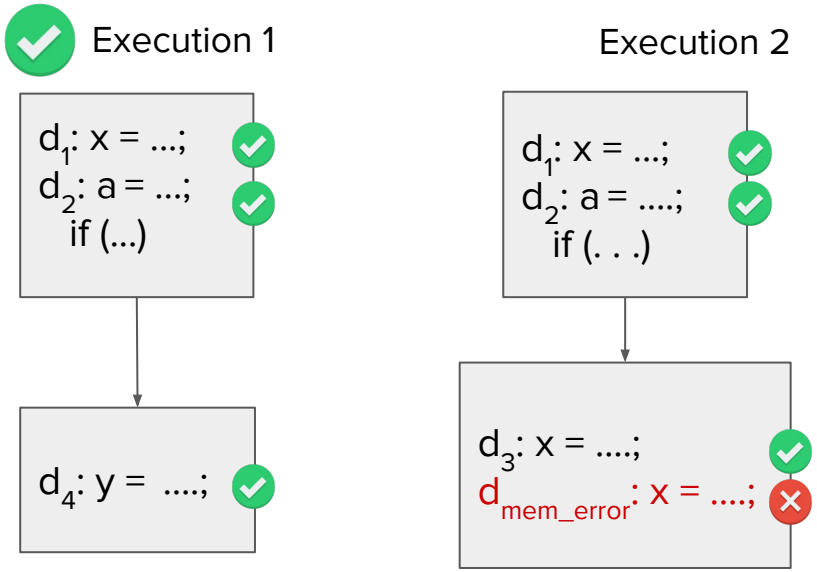
1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06



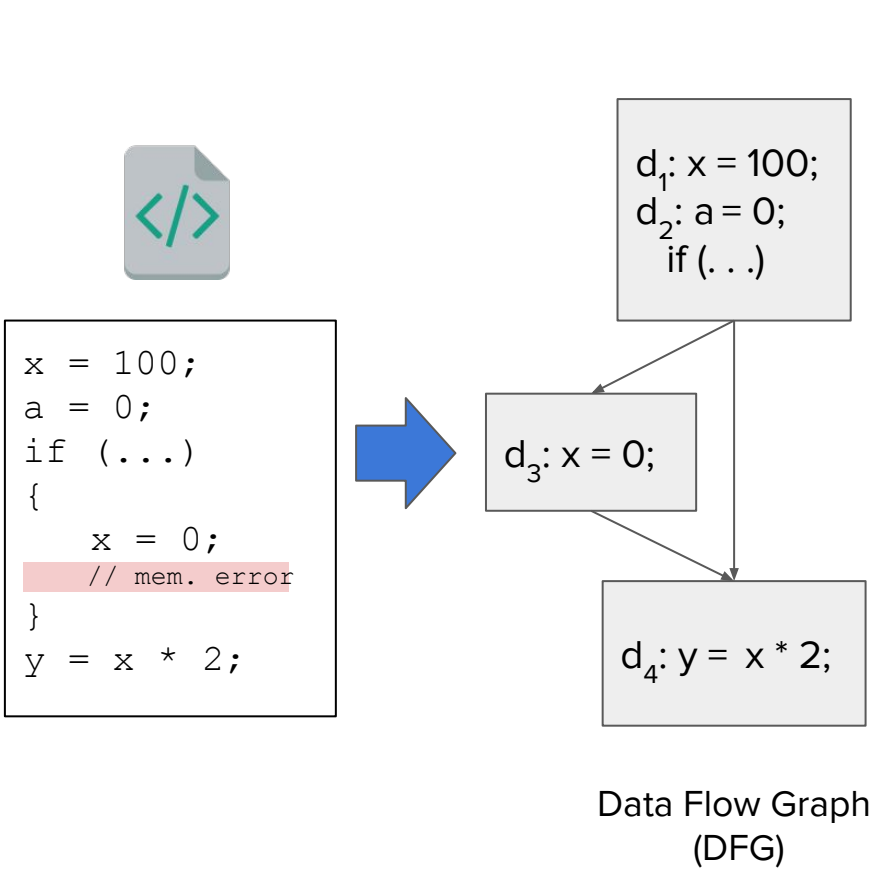


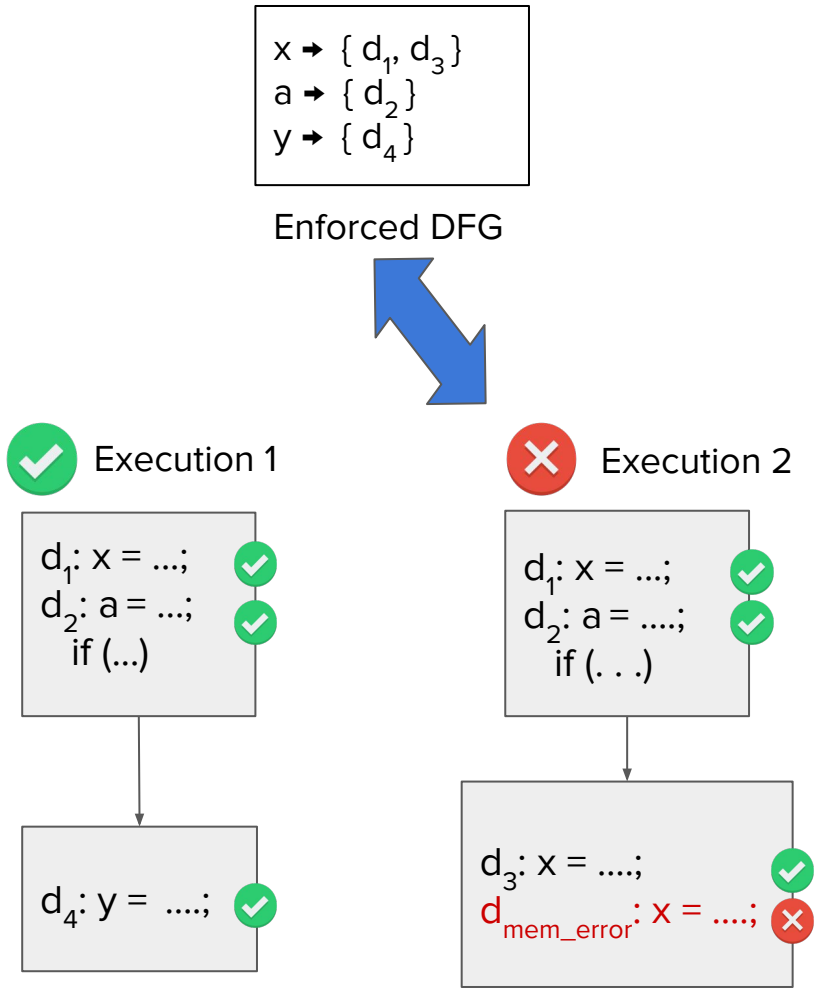
1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Data-Flow Integrity (DFI)

Castro et al. OSDI'06





1 - Offline: DFG computation

2 - Runtime: DFG enforcement

Kernel Data-Flow Integrity

Song et al. NDSS'16

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

Integrity of the code & data of the access controls

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

Integrity of the code & data of the access controls

InferDists

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

ProtectDists

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

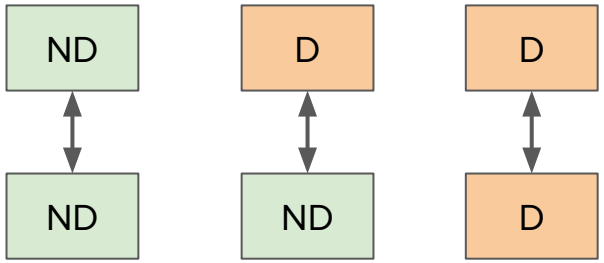
Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

ProtectDists



D: distinguishing, **ND**:non-distinguishing

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

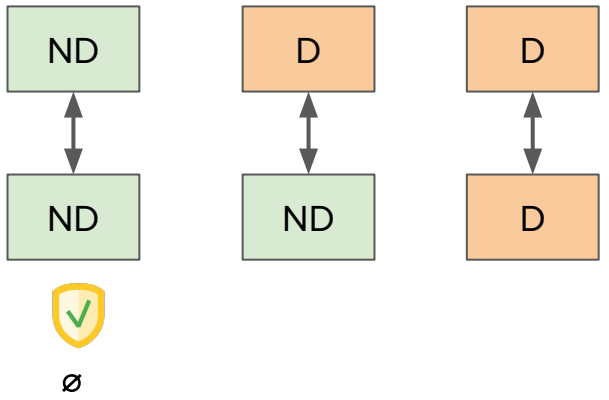
Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

ProtectDists



D: distinguishing, **ND**:non-distinguishing

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

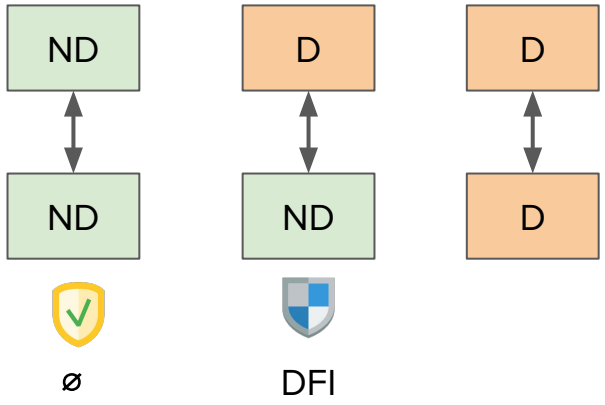
Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

ProtectDists



D: distinguishing, **ND**:non-distinguishing

Kernel Data-Flow Integrity

Song et al. NDSS'16

Protect the kernel against memory-corruption-based privilege escalation attacks

Protect the access control mechanisms

Access control checks

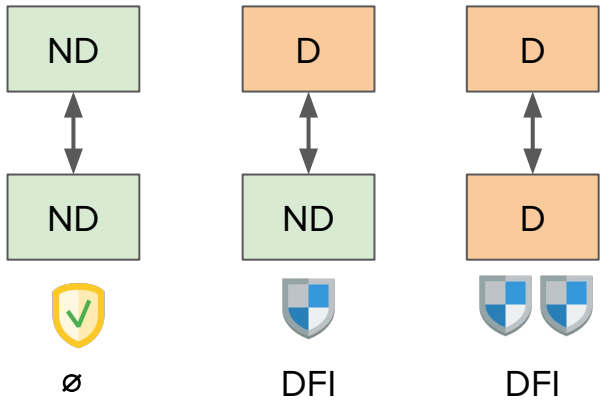
Integrity of the code & data of the access controls

InferDists

Control-Data + Non-Control-Data

Which are essential to enforce the security invariants?

ProtectDists

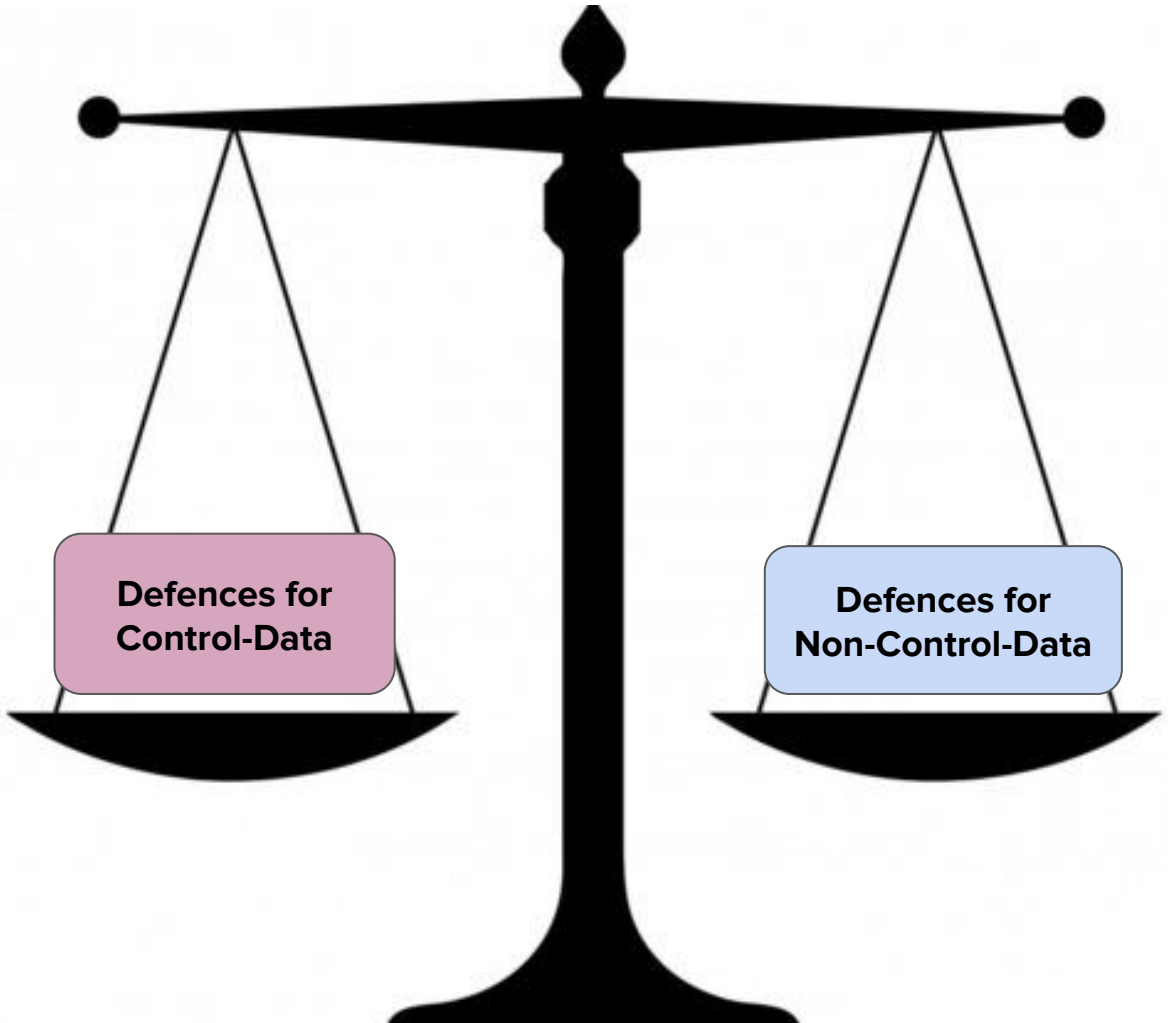


D: distinguishing, **ND**:non-distinguishing

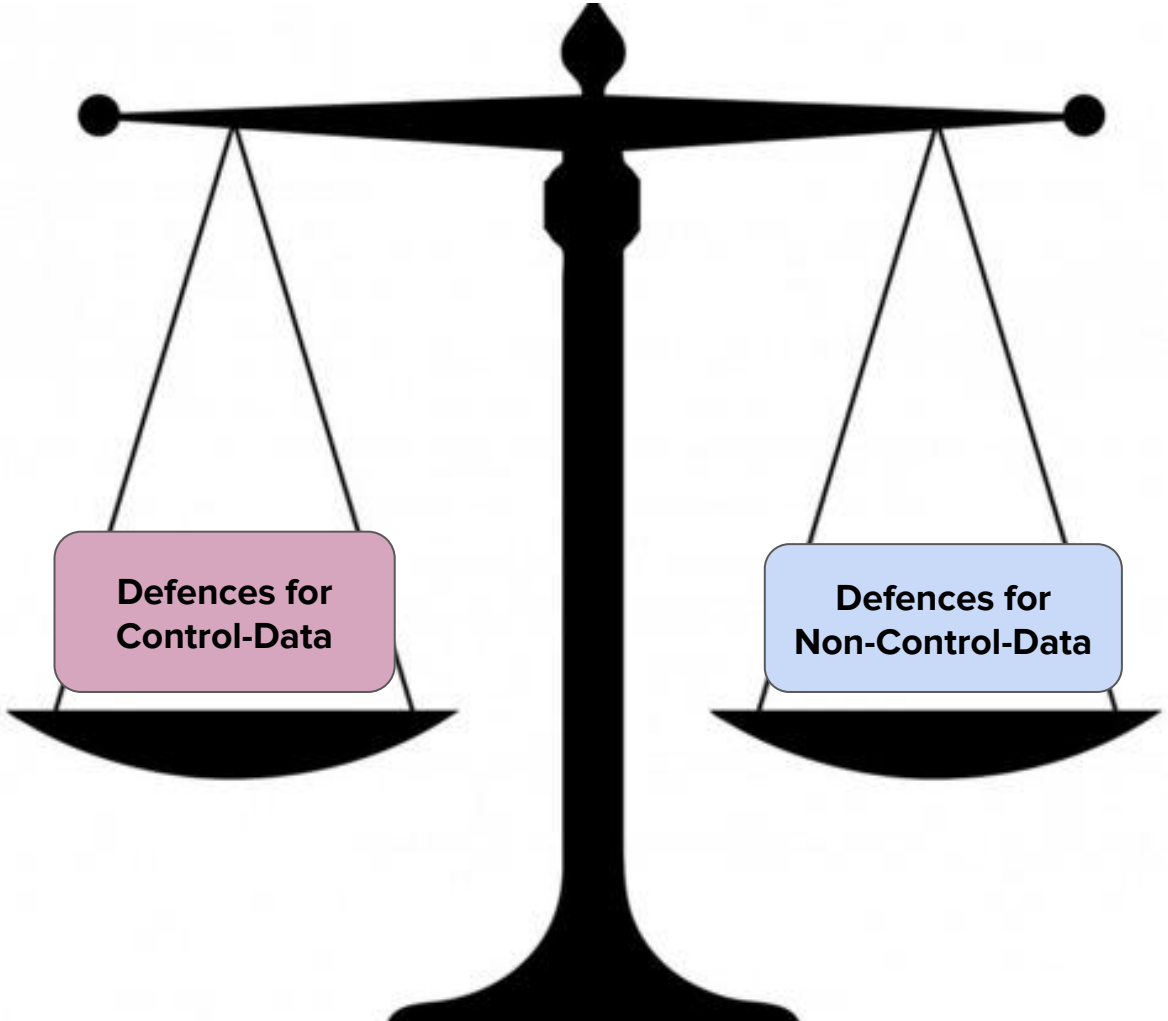
Closing remarks

Closing remarks

Closing remarks



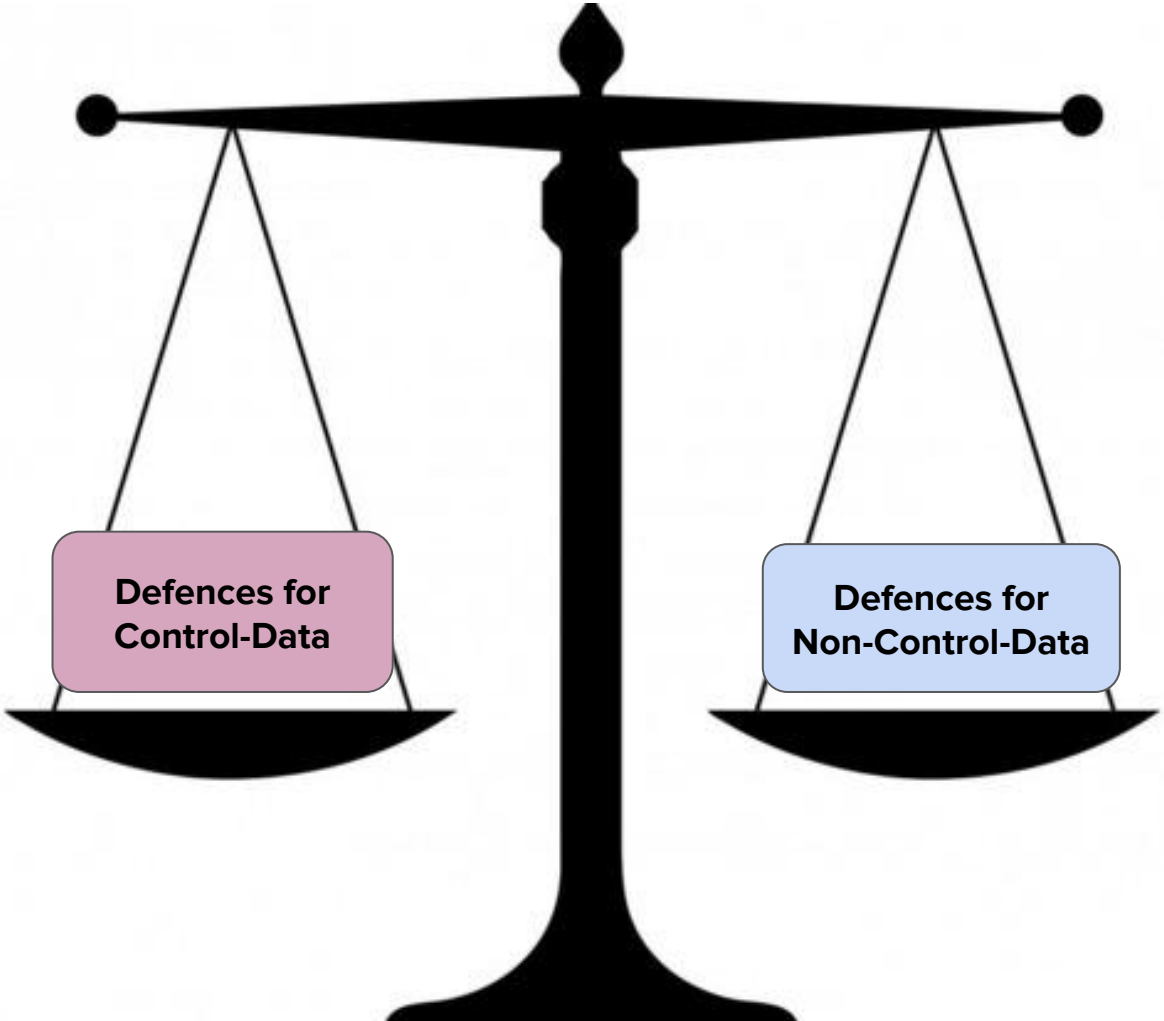
Closing remarks



Closing remarks

Why are they constantly making my life harder?

Haters gonna hate



Data is Flowing in the Wind: A Review of Data-Flow Integrity Methods to Overcome Non-Control-Data Attacks

Irene Díez-Franco, Igor Santos
DeustoTech, University of Deusto
irene.diez@deusto.es isantos@deusto.es