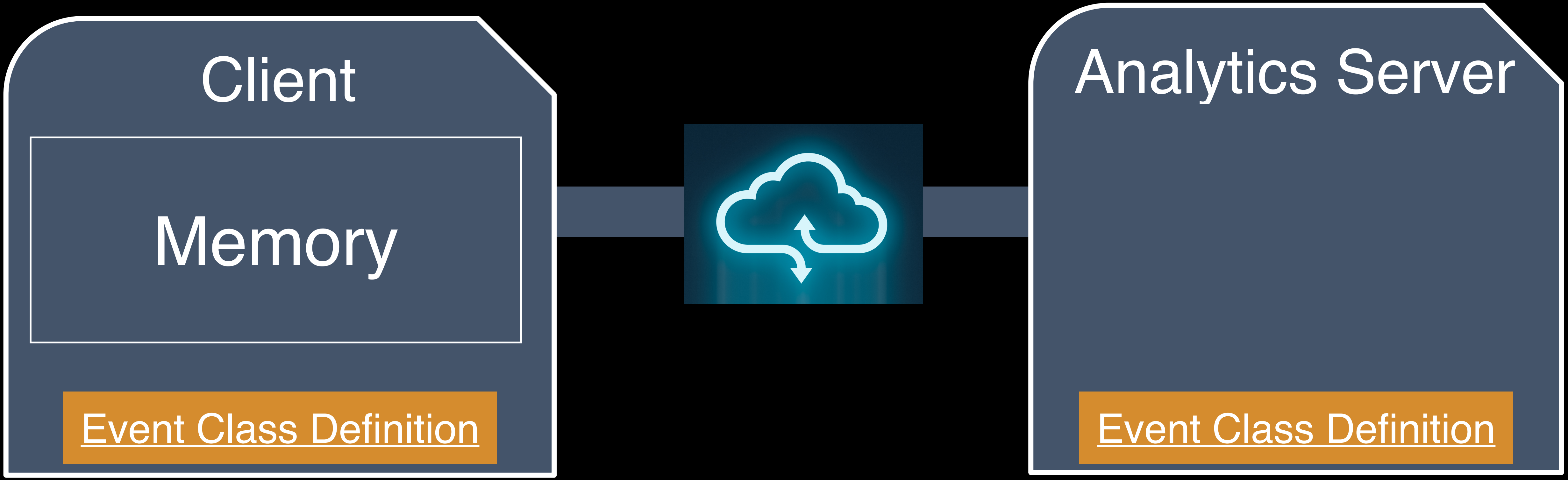
QUACK: Hindering Deserialization Attacks via Static Duck Typing

Yaniv David[1], Neophytos Christou[2], Andreas D. Kellas[1], Vasileios P. Kemerlis[2], Junfeng Yang[1]

[1]Columbia University [2]Brown University

Background: What’s the Deal With Serialization?



Client

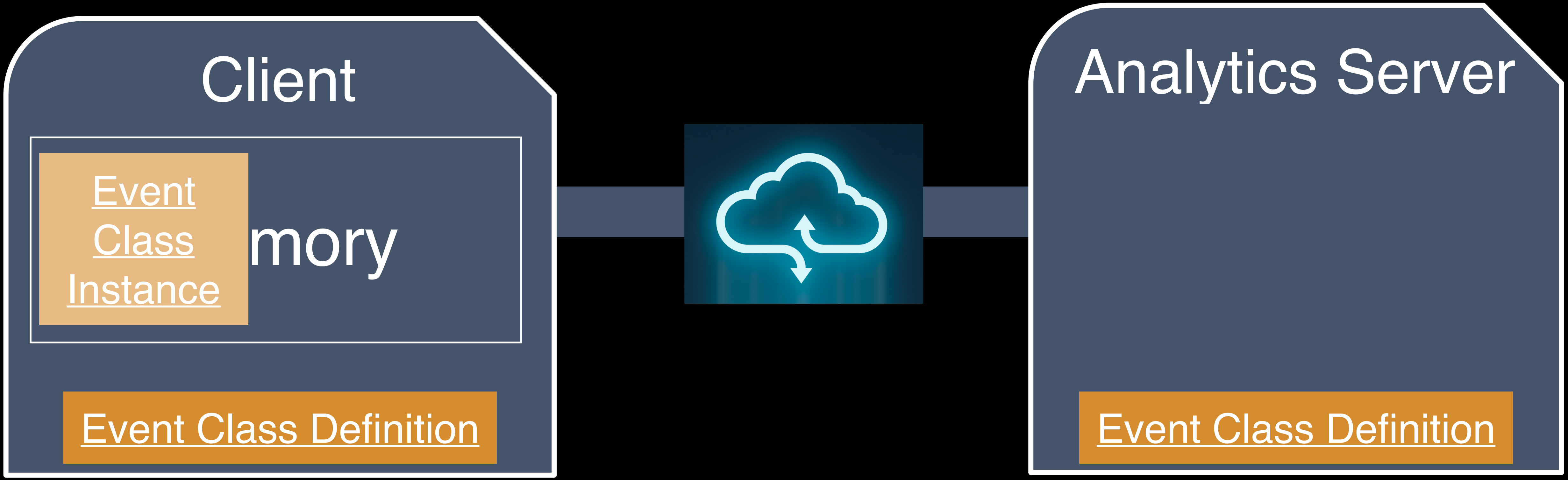
Memory

Event Class Definition

Analytics Server

Event Class Definition

Background: What’s the Deal With Serialization?



Client

Event Class Instance

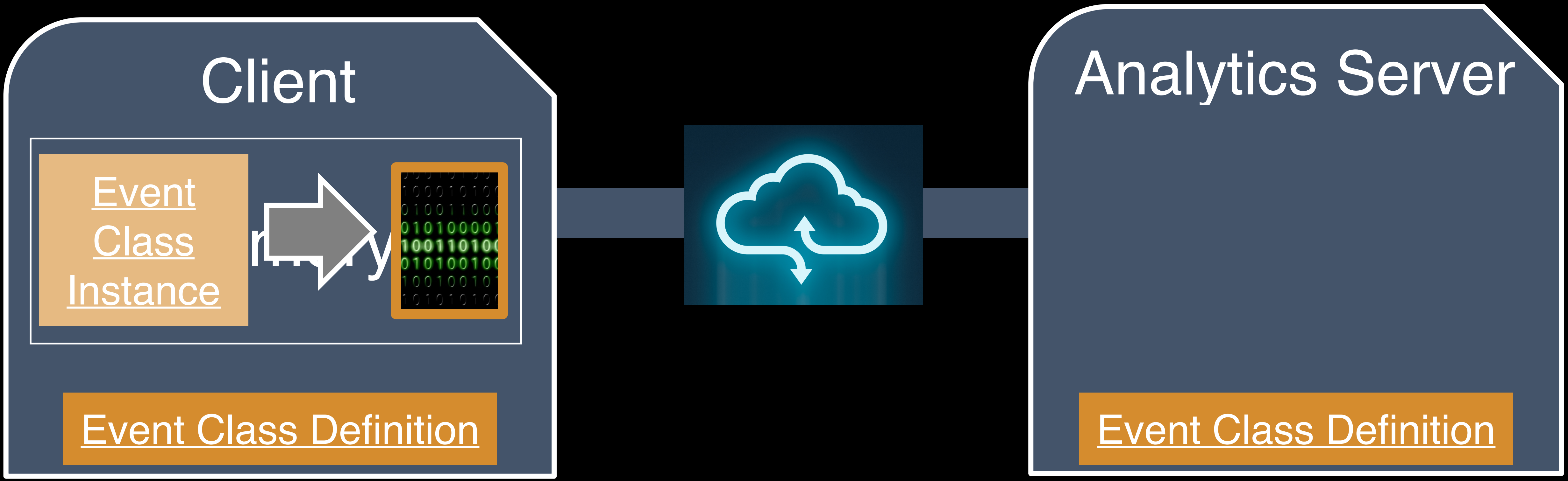
mory

Event Class Definition

Analytics Server

Event Class Definition

Background: What’s the Deal With Serialization?



Client

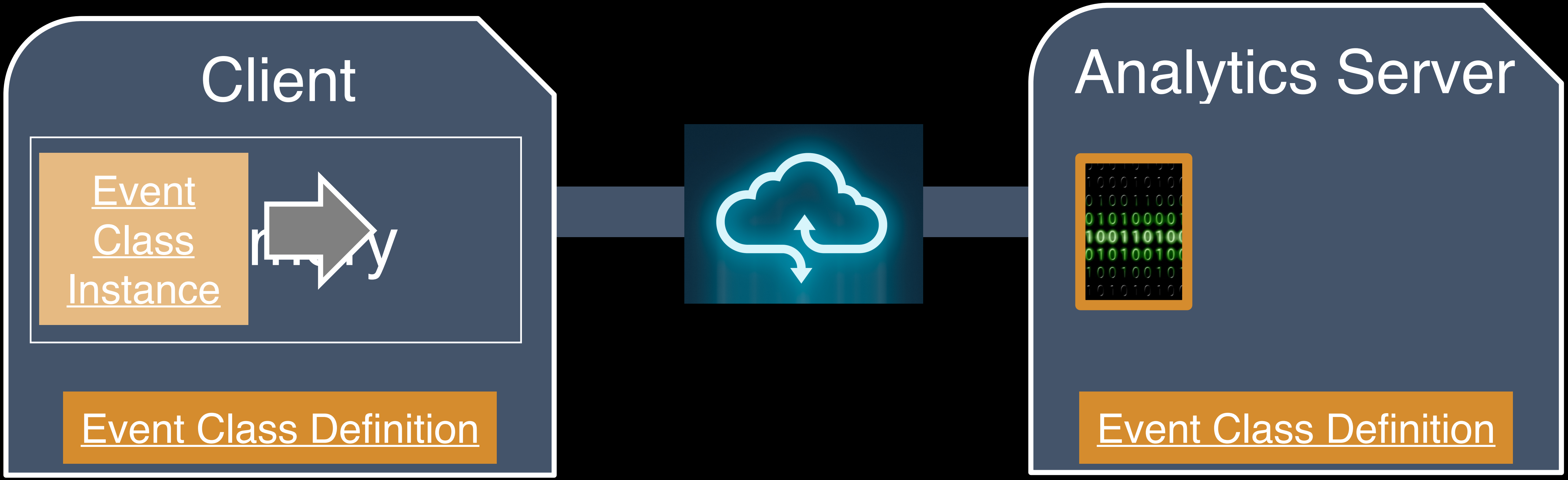
Event Class Instance

Event Class Definition

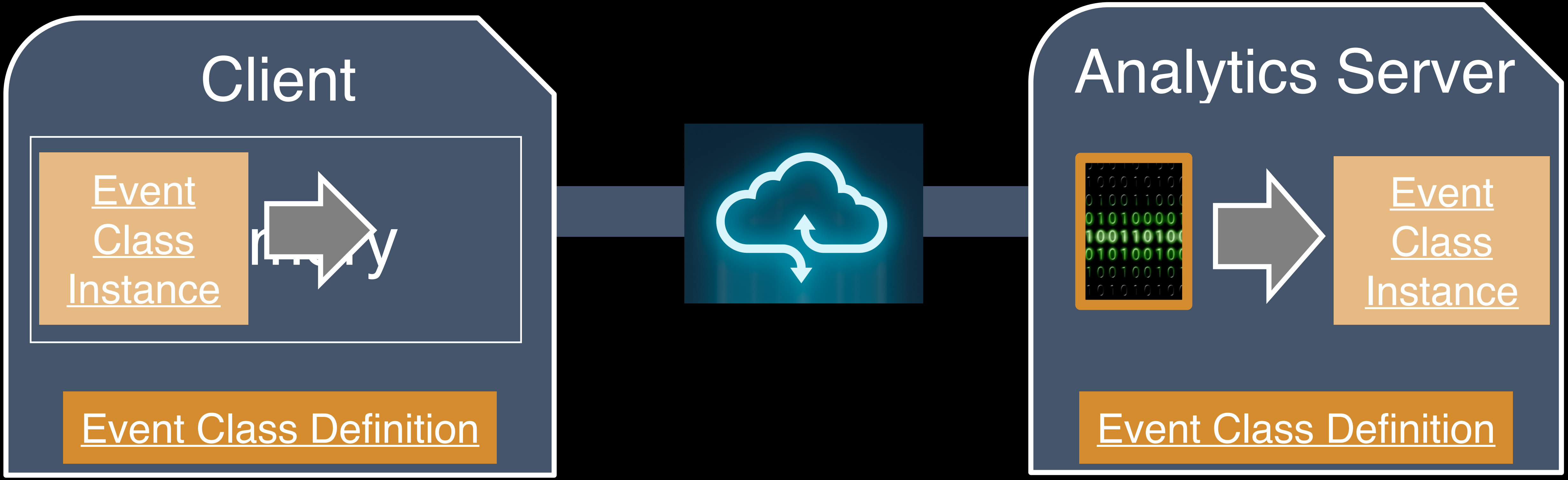
Analytics Server

Event Class Definition

Background: What's the Deal With Serialization?



Background: What’s the Deal With Serialization?



Client

Event Class Instance

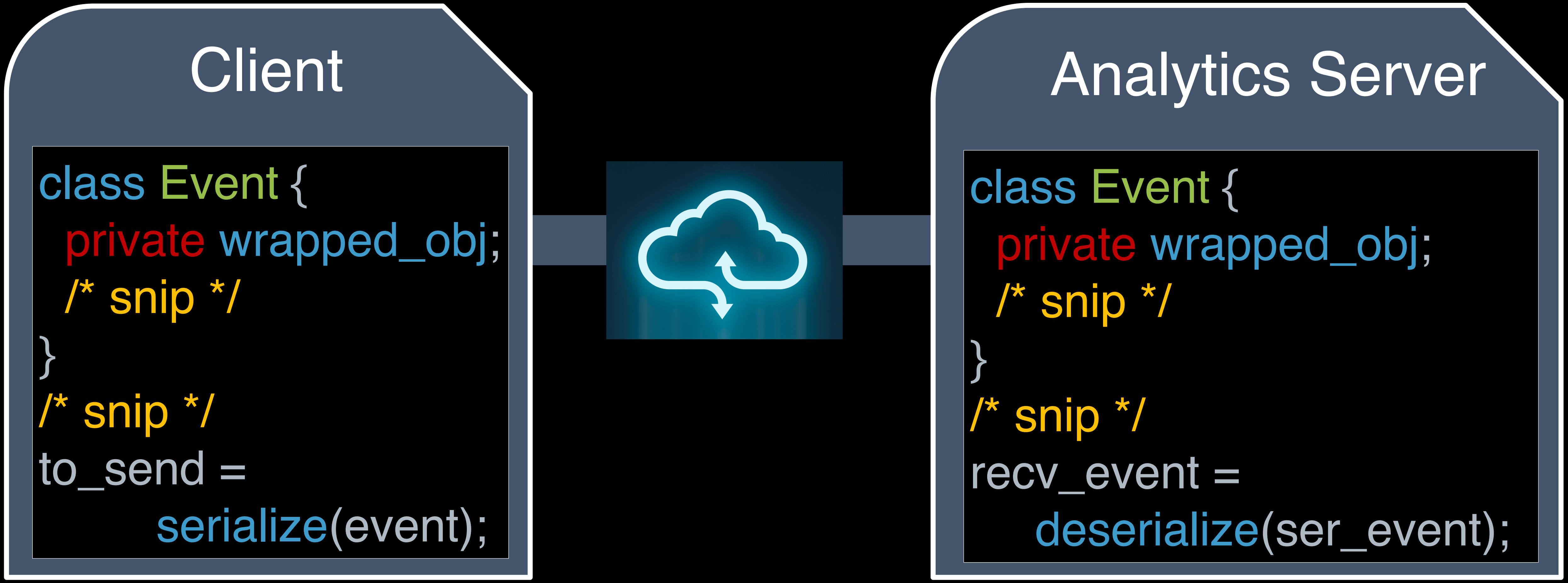
Event Class Definition

Analytics Server

Event Class Instance

Event Class Definition

Ease-Of-Use Trumps Safety



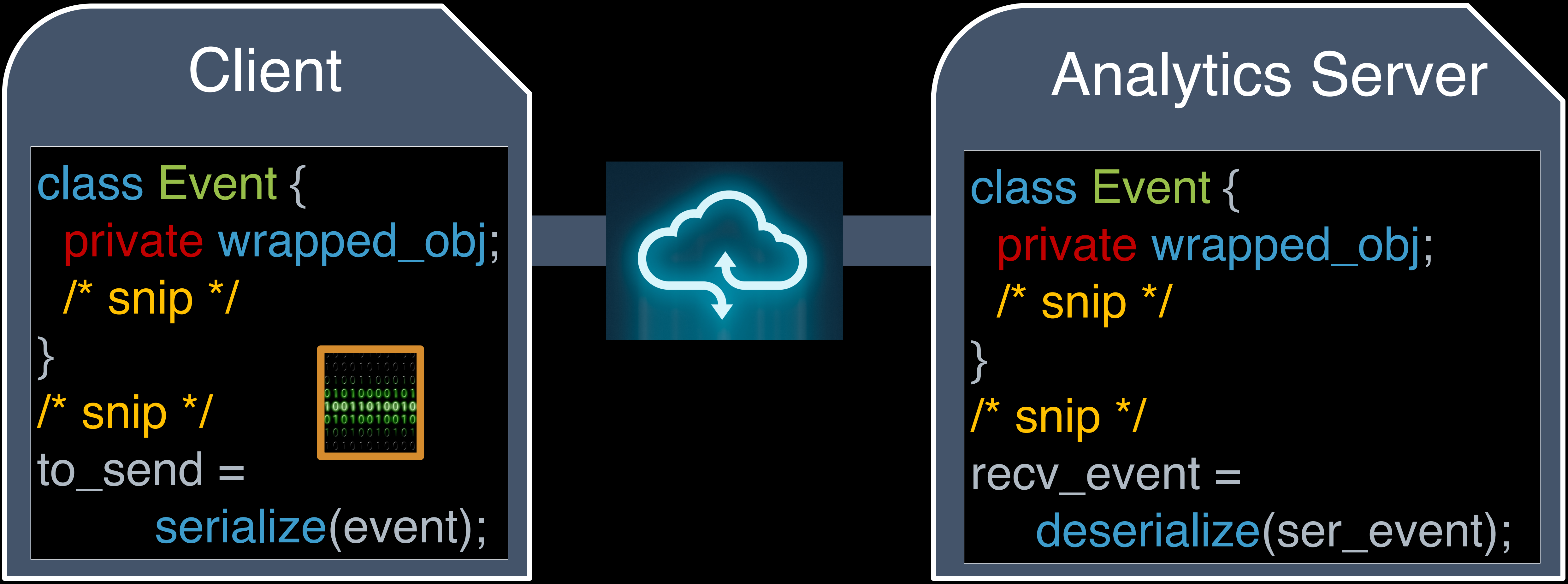
Client

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send =
        serialize(event);
```

Analytics Server

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
recv_event =
    deserialize(ser_event);
```

Ease-Of-Use Trumps Safety



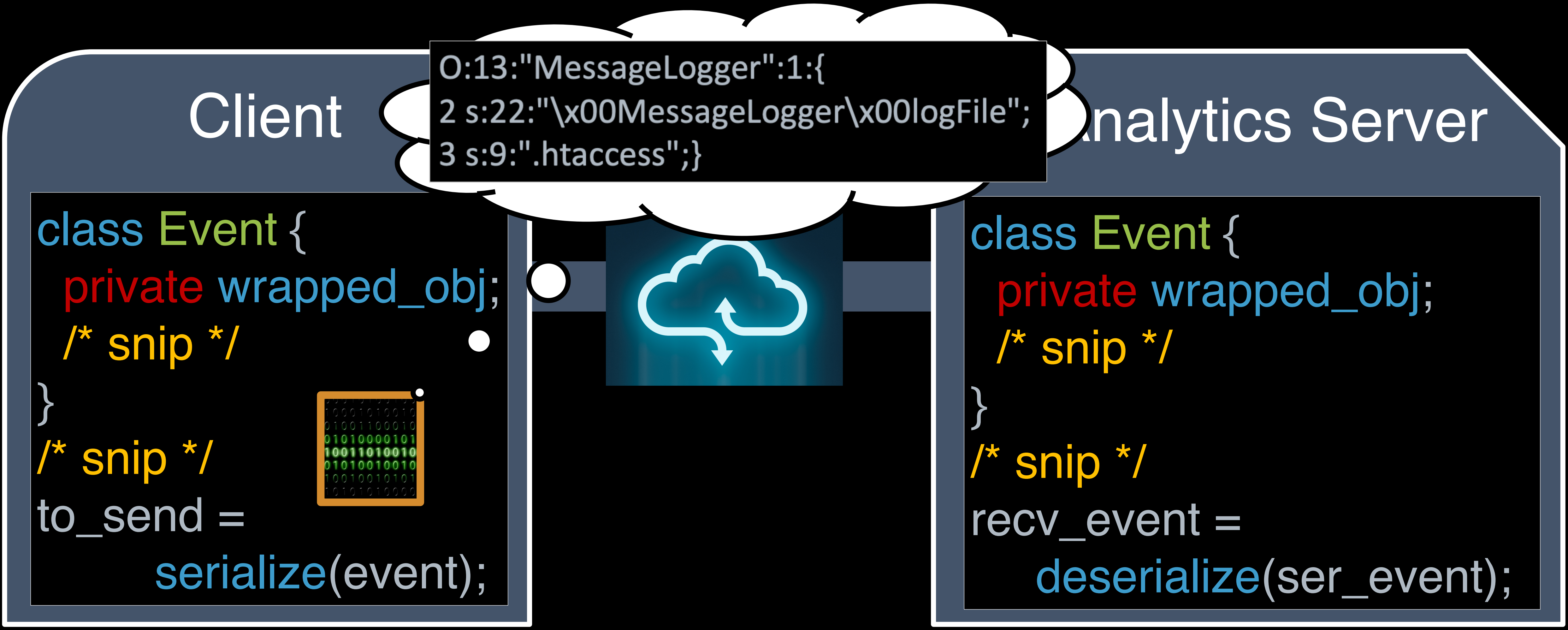
Client

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send =
         serialize(event);
```

Analytics Server

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
recv_event =
    deserialize(ser_event);
```

Ease-Of-Use Trumps Safety



Client

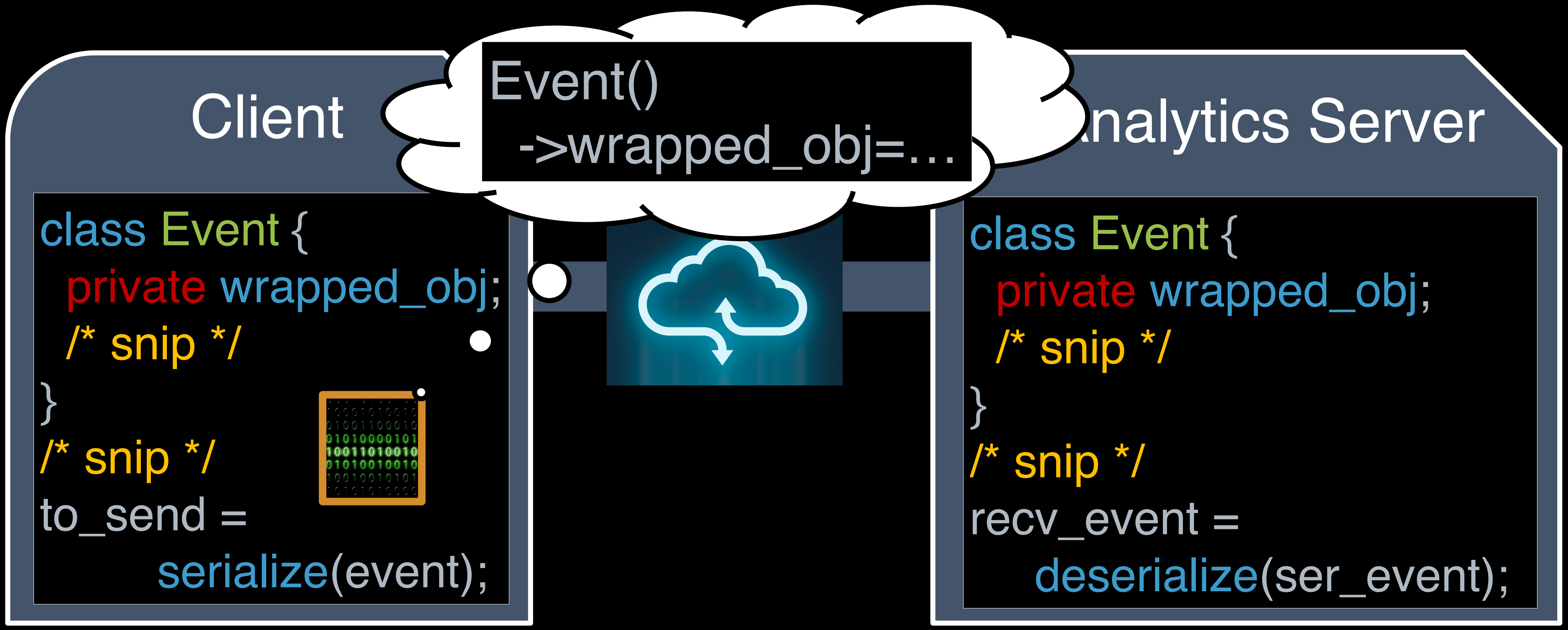
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send =
        serialize(event);
```

```
O:13:"MessageLogger":1:{
s:22:"\x00MessageLogger\x00logFile";
s:9:".htaccess";}
```

nalytics Server

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
recv_event =
      deserialize(ser_event);
```

Ease-Of-Use Trumps Safety



Client

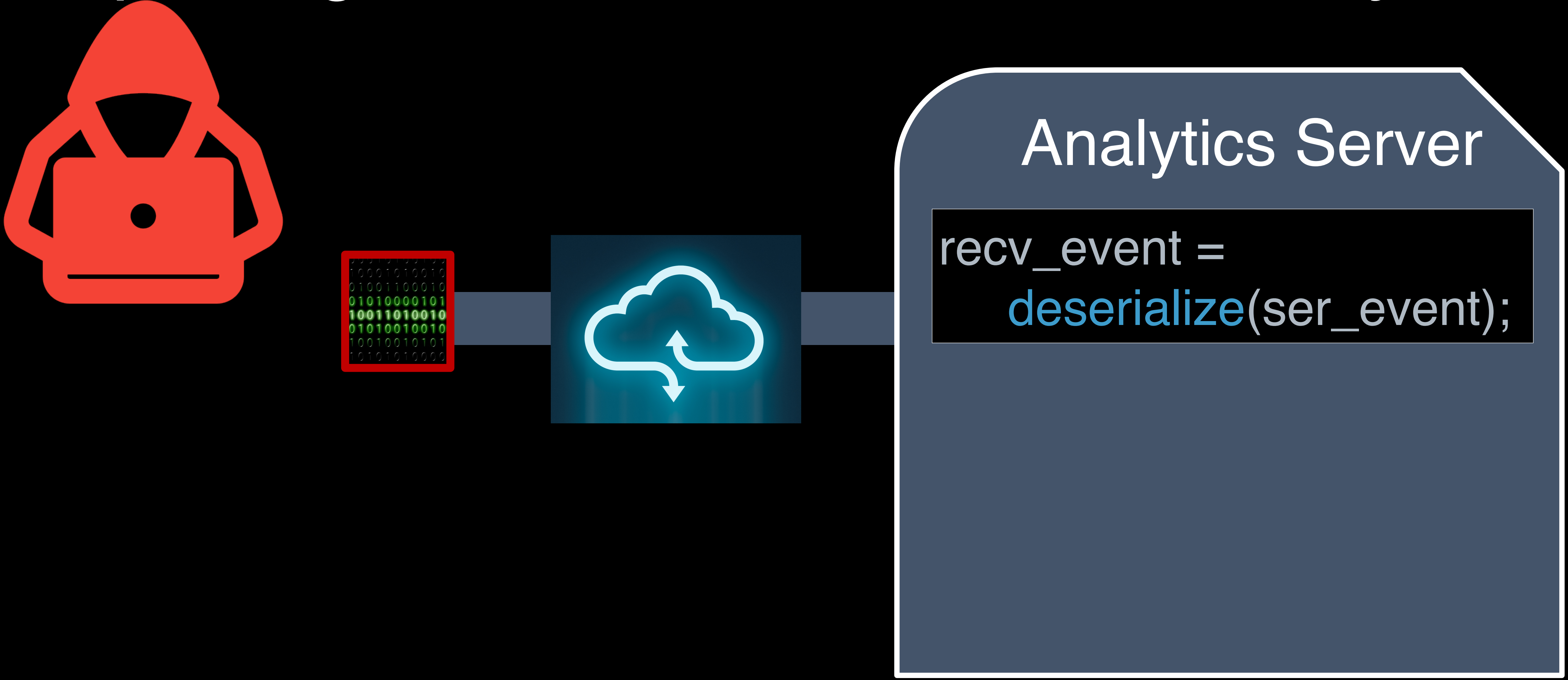
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send =
        serialize(event);
```

```
Event()
  ->wrapped_obj=…
```

Analytics Server

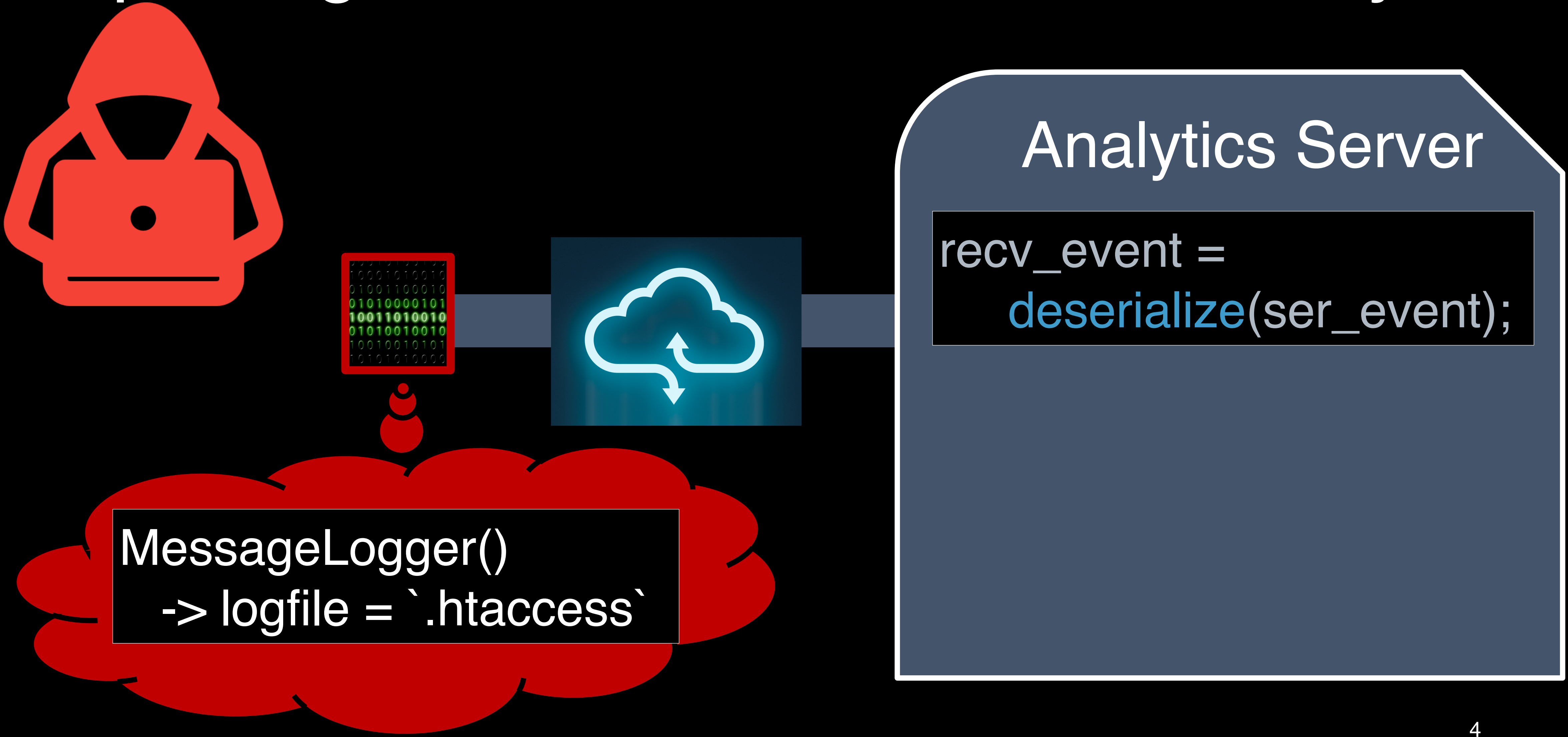
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
recv_event =
     deserialize(ser_event);
```

Exploiting A Deserialization Vulnerability



```
recv_event =
    deserialize(ser_event);
```

Exploiting A Deserialization Vulnerability

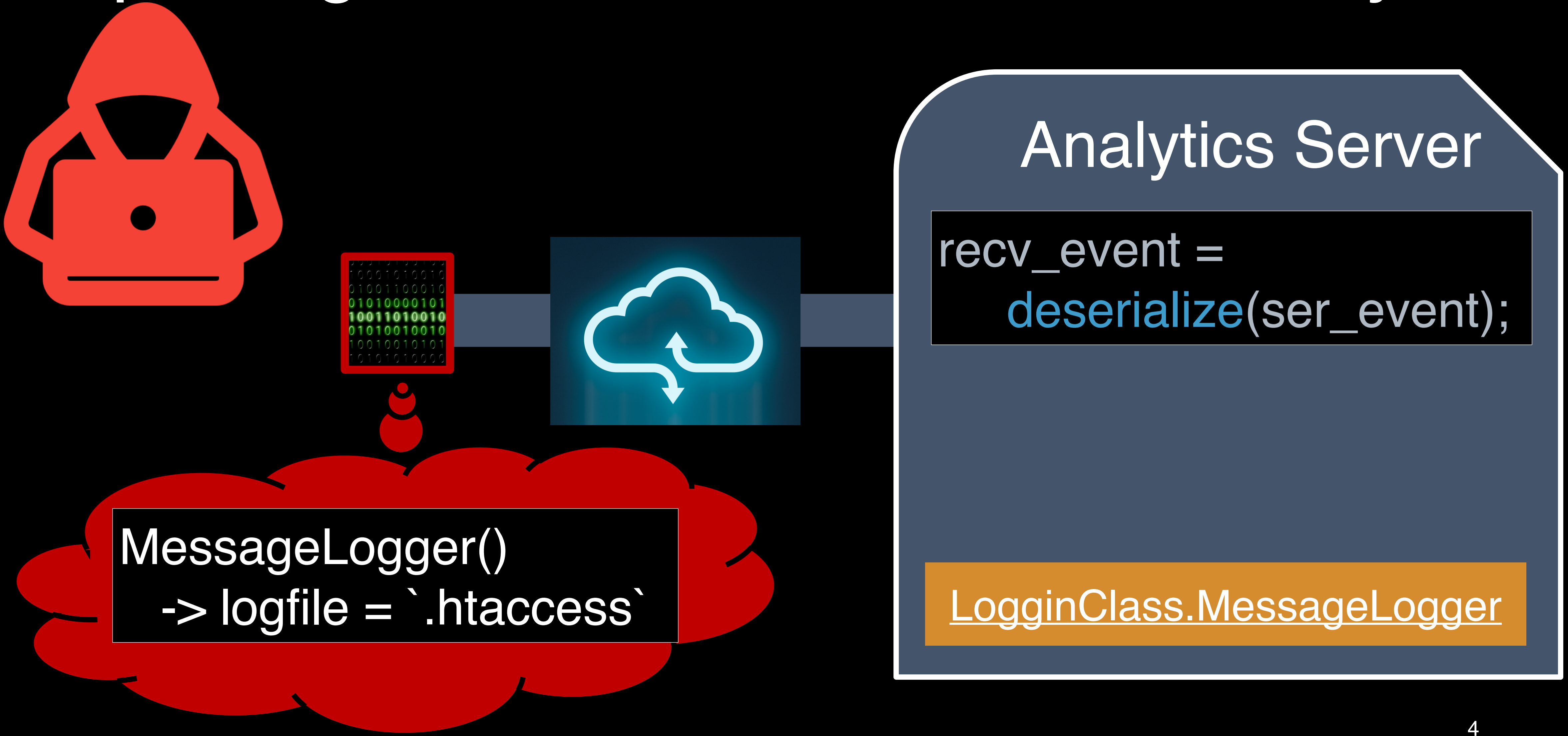


```
MessageLogger()
  -> logfile = `.htaccess`
```

Analytics Server

```
recv_event =
    deserialize(ser_event);
```

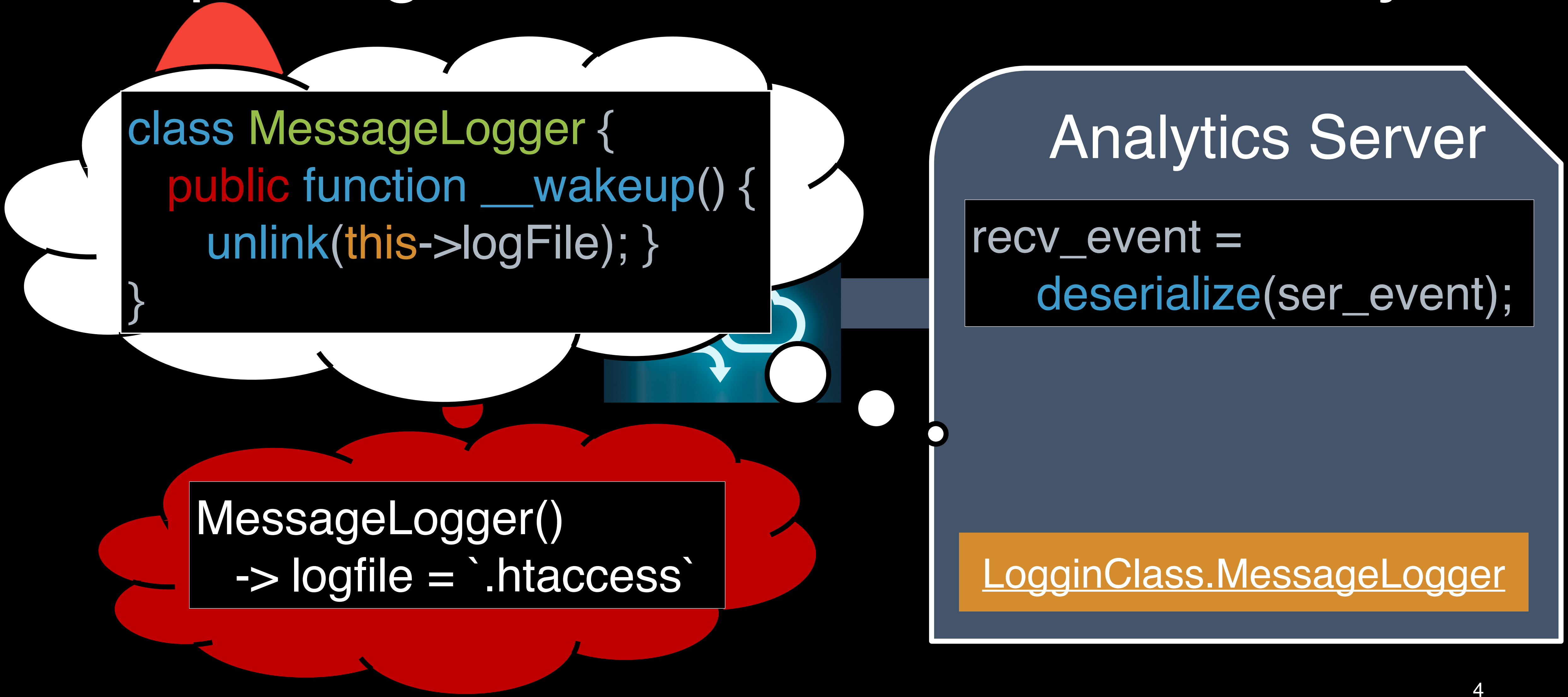
Exploiting A Deserialization Vulnerability



```
recv_event =
    deserialize(ser_event);
```

```
MessageLogger()
  -> logfile = `.htaccess`
```

Exploiting A Deserialization Vulnerability



```
class MessageLogger {
  public function __wakeup() {
    unlink(this->logFile); }
}
```

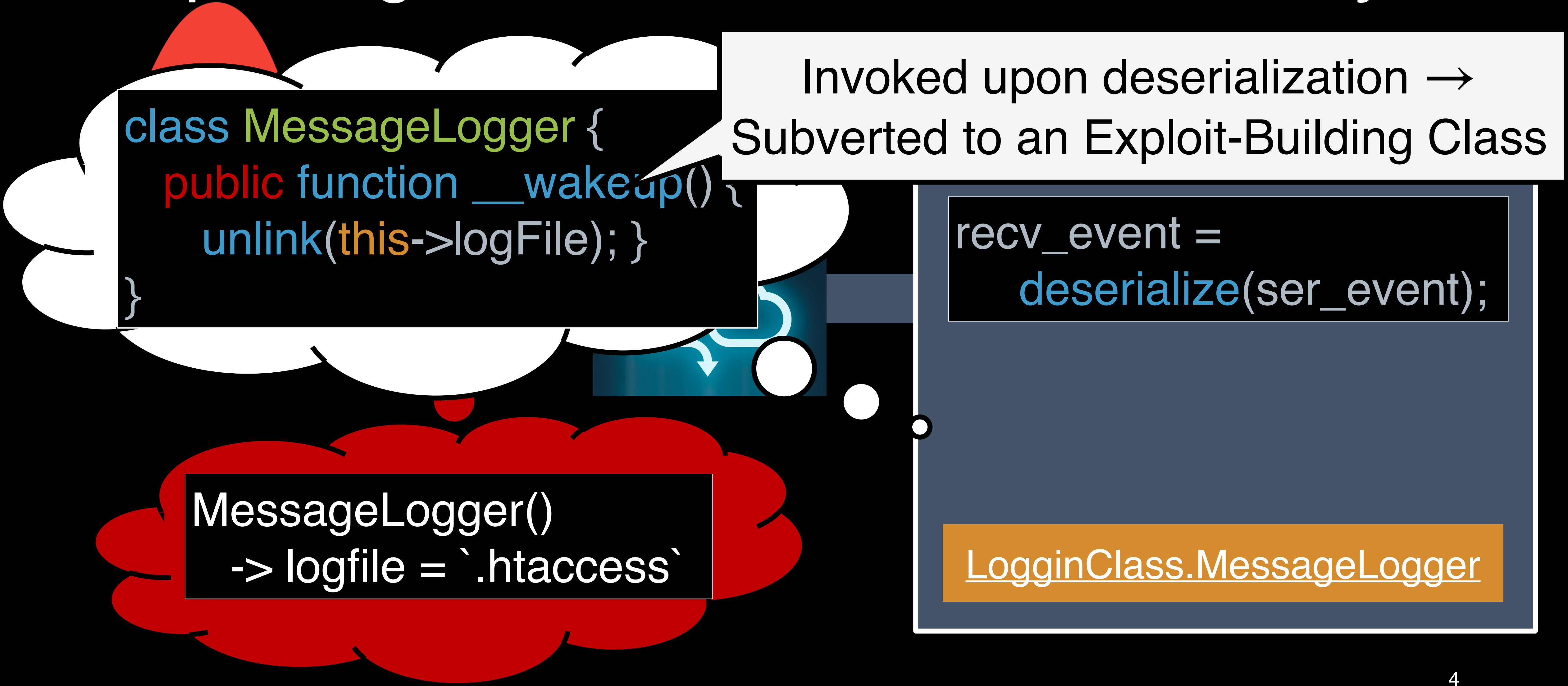
```
MessageLogger()
  -> logfile = `.htaccess`
```

Analytics Server

```
recv_event =
    deserialize(ser_event);
```

LogginClass.MessageLogger

Exploiting A Deserialization Vulnerability



```
class MessageLogger {
  public function __wakeup() {
    unlink(this->logFile); }
}
```

Invoked upon deserialization →
Subverted to an Exploit-Building Class

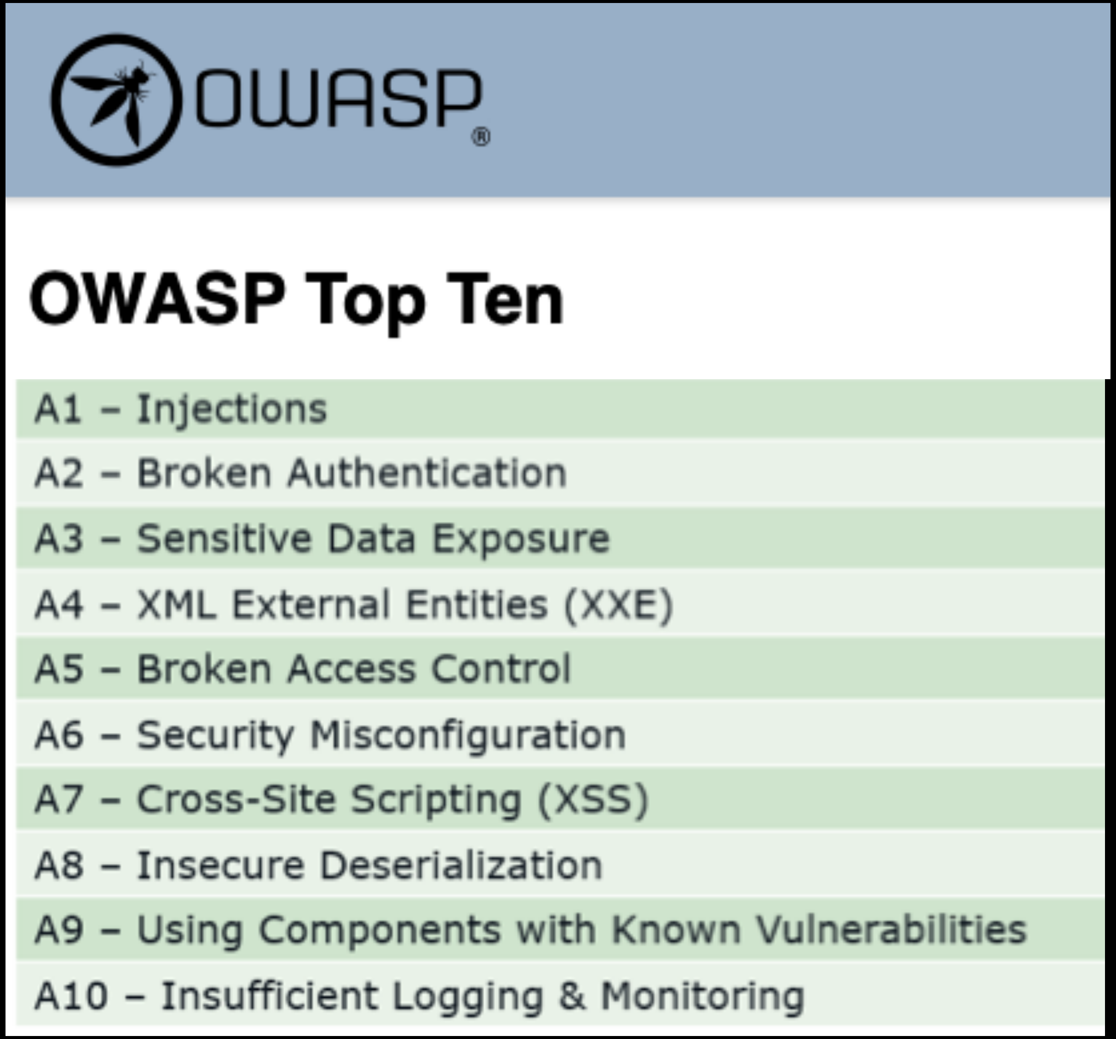
```
recv_event =
    deserialize(ser_event);
```

LogginClass.MessageLogger

```
MessageLogger()
  -> logfile = `.htaccess`
```

Deserialization Attacks Affect Real-World Applications

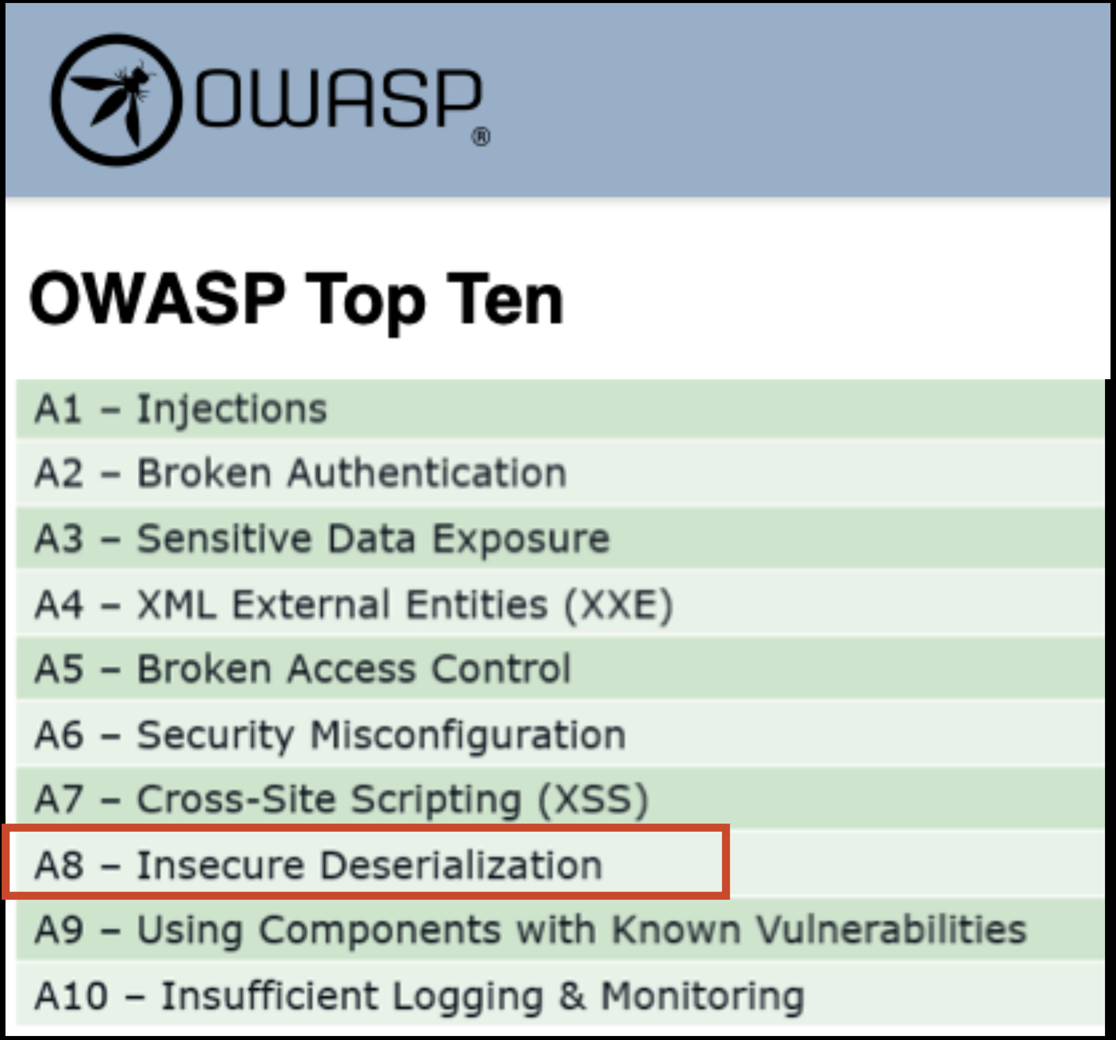
Deserialization Attacks Affect Real-World Applications



OWASP Top Ten

- A1 – Injections
- A2 – Broken Authentication
- A3 – Sensitive Data Exposure
- A4 – XML External Entities (XXE)
- A5 – Broken Access Control
- A6 – Security Misconfiguration
- A7 – Cross-Site Scripting (XSS)
- A8 – Insecure Deserialization
- A9 – Using Components with Known Vulnerabilities
- A10 – Insufficient Logging & Monitoring

Deserialization Attacks Affect Real-World Applications

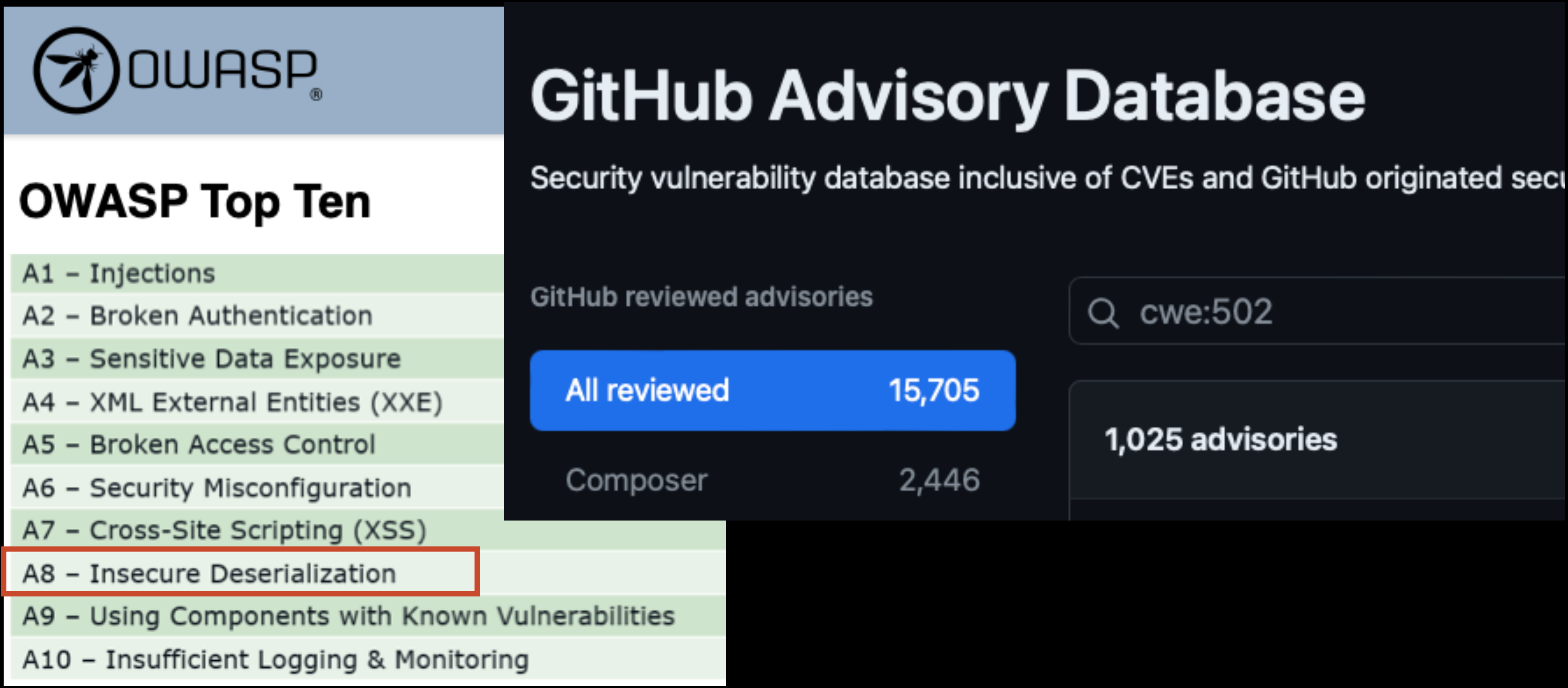


OWASP®

OWASP Top Ten

- A1 – Injections
- A2 – Broken Authentication
- A3 – Sensitive Data Exposure
- A4 – XML External Entities (XXE)
- A5 – Broken Access Control
- A6 – Security Misconfiguration
- A7 – Cross-Site Scripting (XSS)
- A8 – Insecure Deserialization
- A9 – Using Components with Known Vulnerabilities
- A10 – Insufficient Logging & Monitoring

Deserialization Attacks Affect Real-World Applications



OWASP®

OWASP Top Ten

- A1 – Injections
- A2 – Broken Authentication
- A3 – Sensitive Data Exposure
- A4 – XML External Entities (XXE)
- A5 – Broken Access Control
- A6 – Security Misconfiguration
- A7 – Cross-Site Scripting (XSS)
- A8 – Insecure Deserialization
- A9 – Using Components with Known Vulnerabilities
- A10 – Insufficient Logging & Monitoring

GitHub Advisory Database

Security vulnerability database inclusive of CVEs and GitHub originated secu

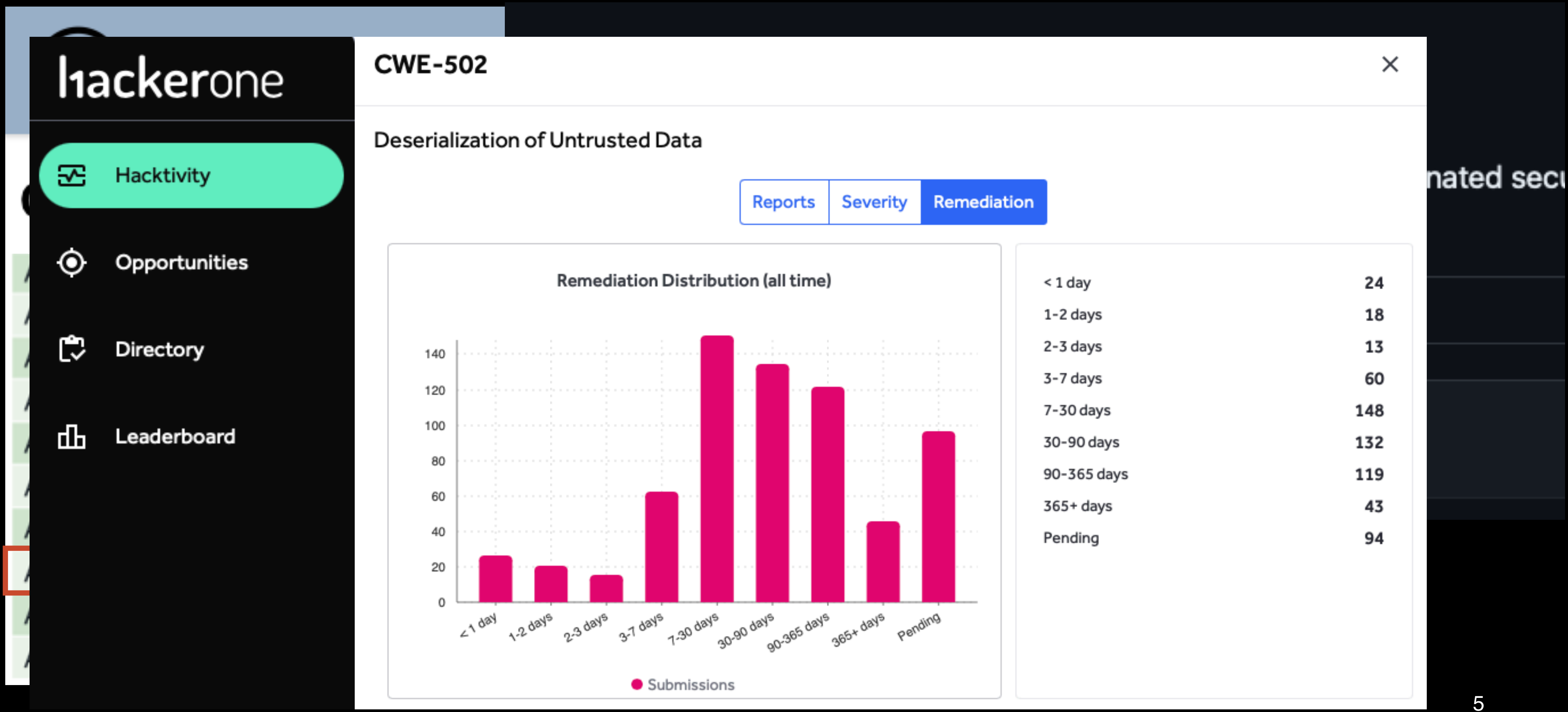
GitHub reviewed advisories

All reviewed	15,705
Composer	2,446

cwe:502

1,025 advisories

Deserialization Attacks Affect Real-World Applications



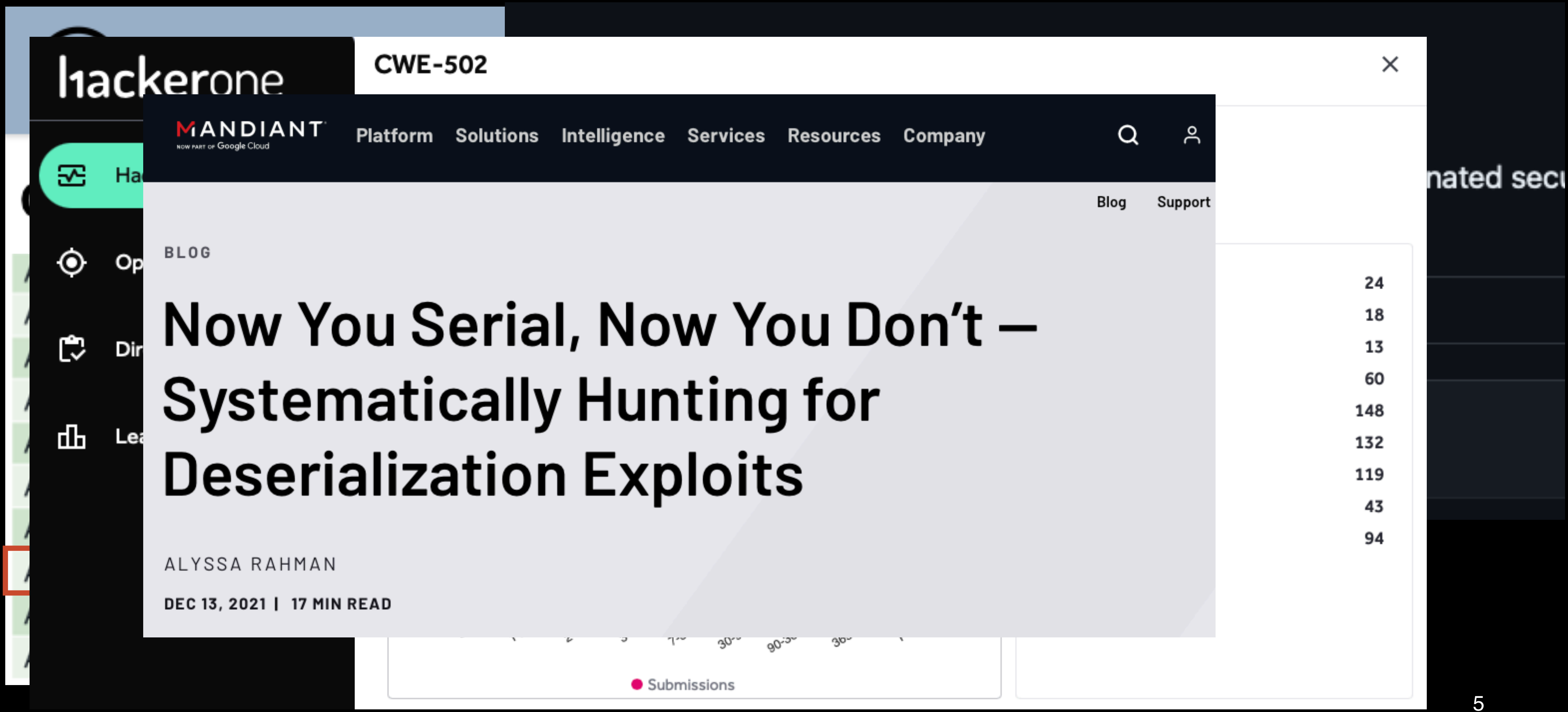
CWE-502

Deserialization of Untrusted Data

Remediation Distribution (all time)

< 1 day	24
1-2 days	18
2-3 days	13
3-7 days	60
7-30 days	148
30-90 days	132
90-365 days	119
365+ days	43
Pending	94

Deserialization Attacks Affect Real-World Applications



BLOG

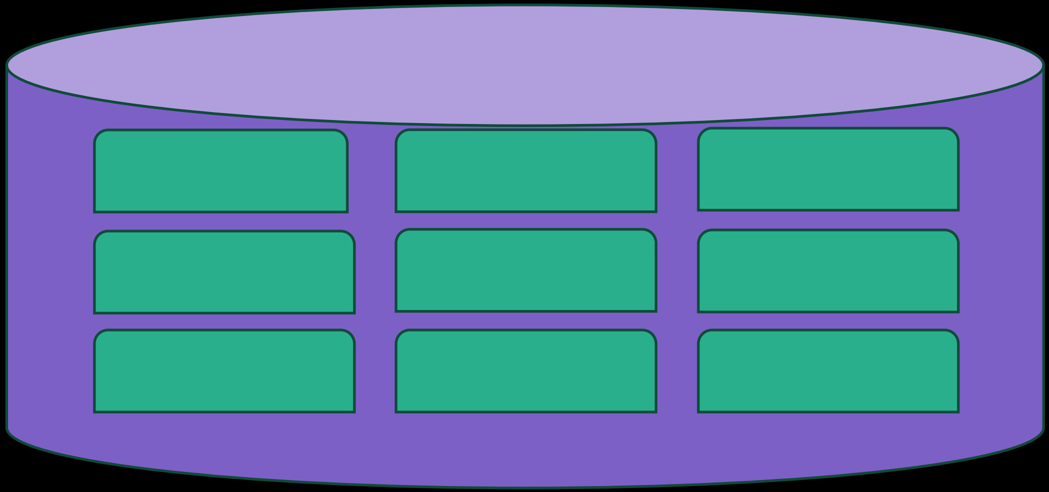
Now You Serial, Now You Don’t — Systematically Hunting for Deserialization Exploits

ALYSSA RAHMAN

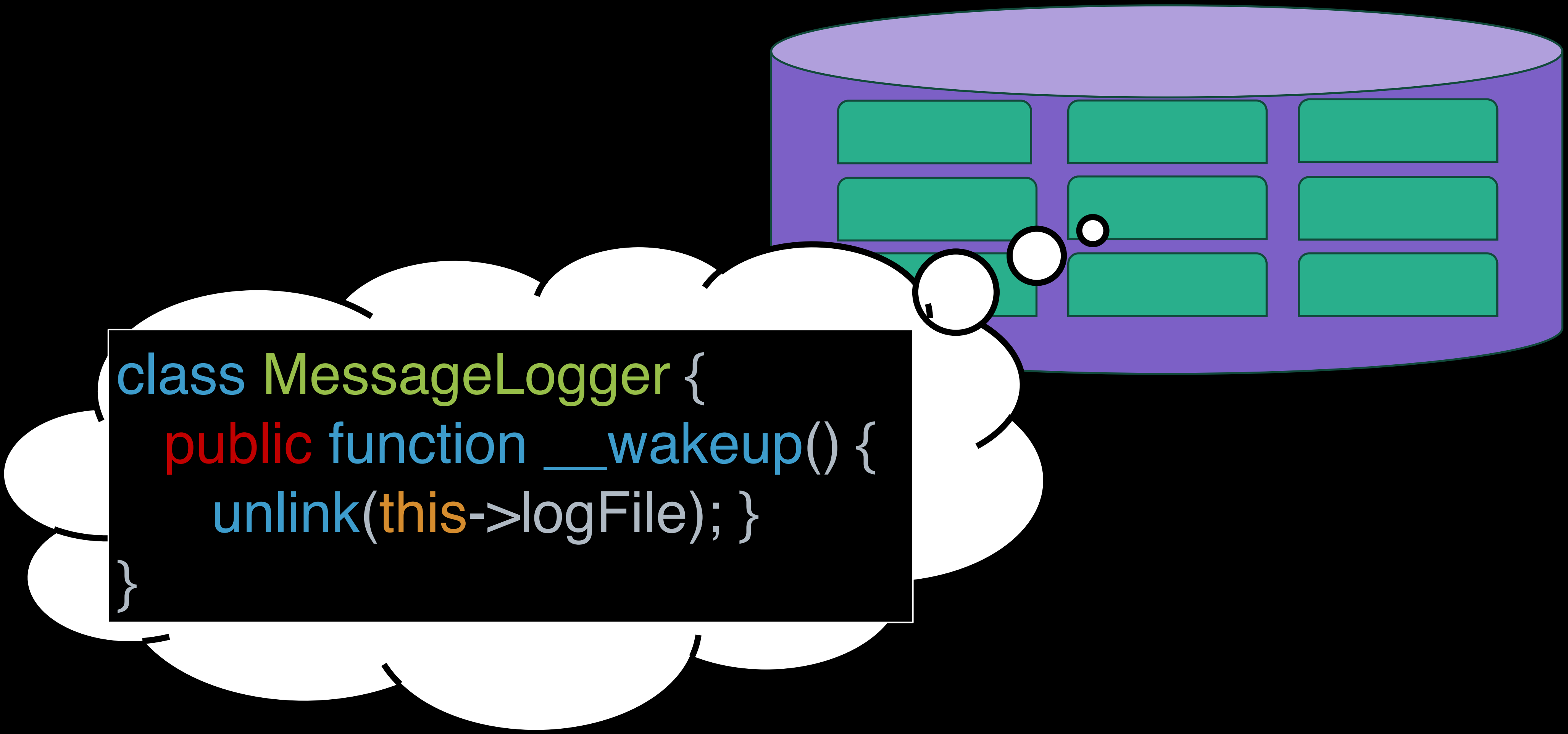
DEC 13, 2021 | 17 MIN READ

Exploit Generation Techniques Are Improving

Exploit Generation Techniques Are Improving

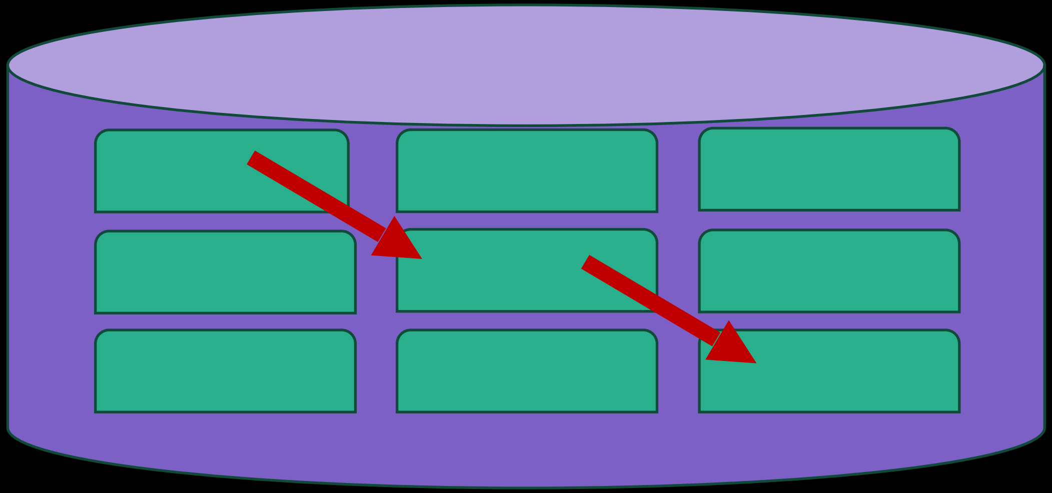


Exploit Generation Techniques Are Improving



```
class MessageLogger {
  public function __wakeup() {
    unlink(this->logFile); }
}
```

Exploit Generation Techniques Are Improving

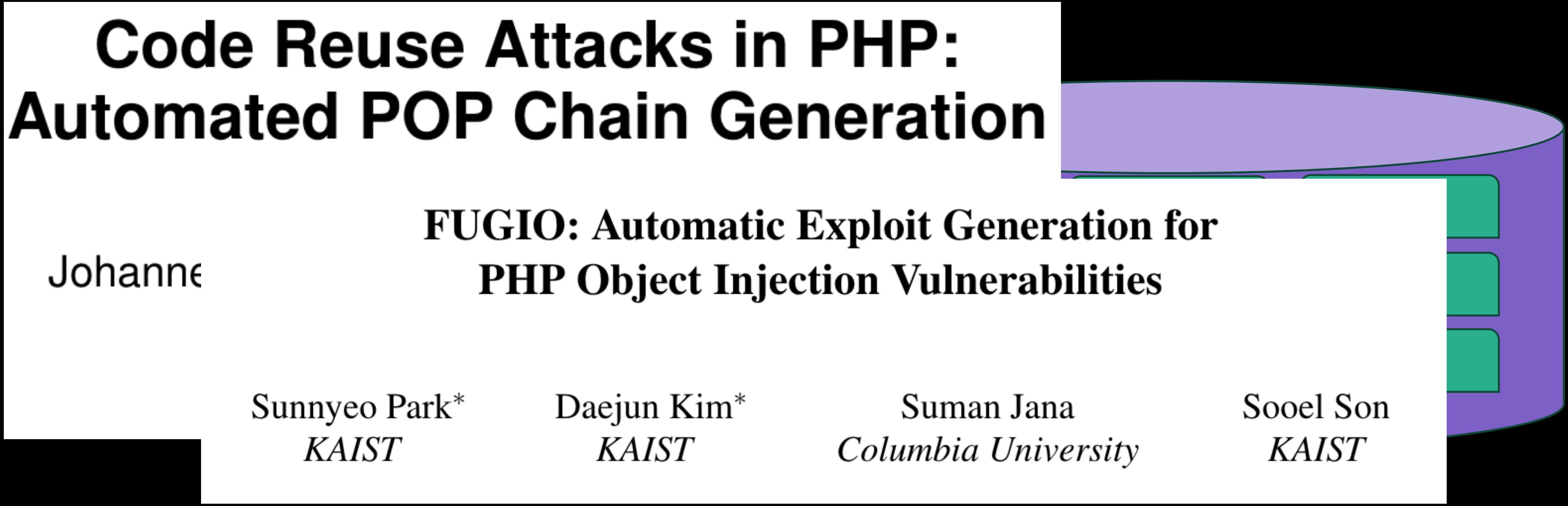


Exploit Generation Techniques Are Improving

Code Reuse Attacks in PHP: Automated POP Chain Generation

Johannes Dahse, Nikolai Krein, and Thorsten Holz
Horst Görtz Institute for IT-Security (HGI)
Ruhr-University Bochum, Germany
{firstname.lastname}@rub.de

Exploit Generation Techniques Are Improving



Code Reuse Attacks in PHP: Automated POP Chain Generation

Johanne

FUGIO: Automatic Exploit Generation for PHP Object Injection Vulnerabilities

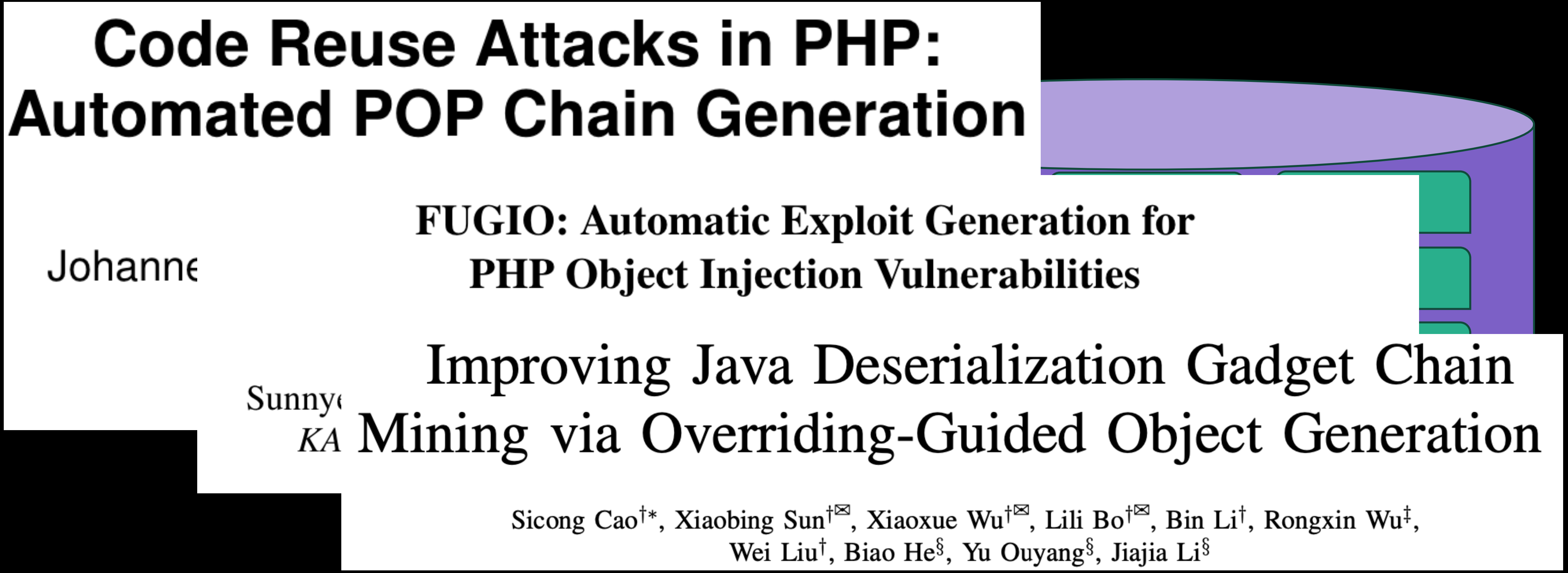
Sunnyeo Park*
KAIST

Daejun Kim*
KAIST

Suman Jana
Columbia University

Sooel Son
KAIST

Exploit Generation Techniques Are Improving



Code Reuse Attacks in PHP: Automated POP Chain Generation

Johanne

FUGIO: Automatic Exploit Generation for PHP Object Injection Vulnerabilities

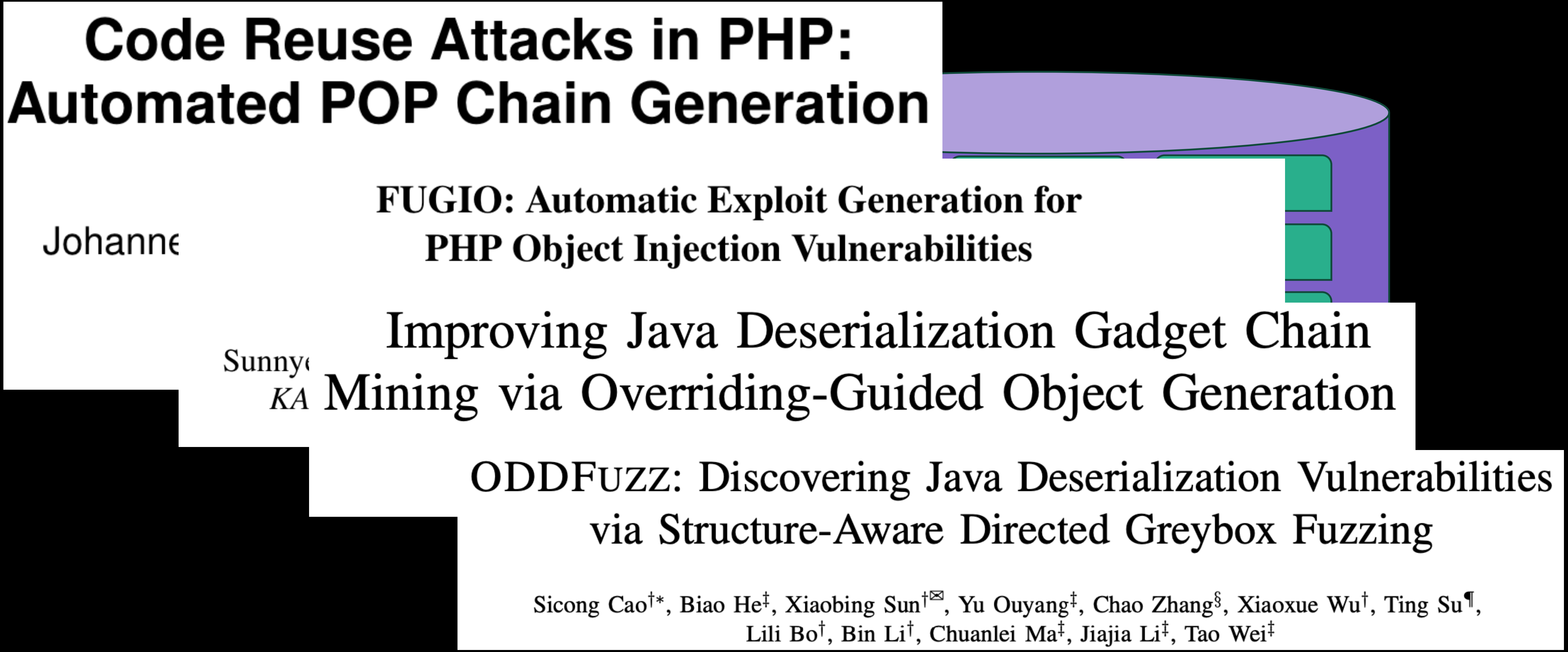
Sunny

KA

Improving Java Deserialization Gadget Chain Mining via Overriding-Guided Object Generation

Sicong Cao[†*], Xiaobing Sun[†✉], Xiaoxue Wu[†✉], Lili Bo[†✉], Bin Li[†], Rongxin Wu[‡],
Wei Liu[†], Biao He[§], Yu Ouyang[§], Jiajia Li[§]

Exploit Generation Techniques Are Improving



Code Reuse Attacks in PHP: Automated POP Chain Generation

Johanne

FUGIO: Automatic Exploit Generation for PHP Object Injection Vulnerabilities

Sunny

KA

Improving Java Deserialization Gadget Chain Mining via Overriding-Guided Object Generation

ODDFuzz: Discovering Java Deserialization Vulnerabilities via Structure-Aware Directed Greybox Fuzzing

Sicong Cao[†*], Biao He[‡], Xiaobing Sun[†✉], Yu Ouyang[‡], Chao Zhang[§], Xiaoxue Wu[†], Ting Su[¶], Lili Bo[†], Bin Li[†], Chuanlei Ma[‡], Jiajia Li[‡], Tao Wei[‡]

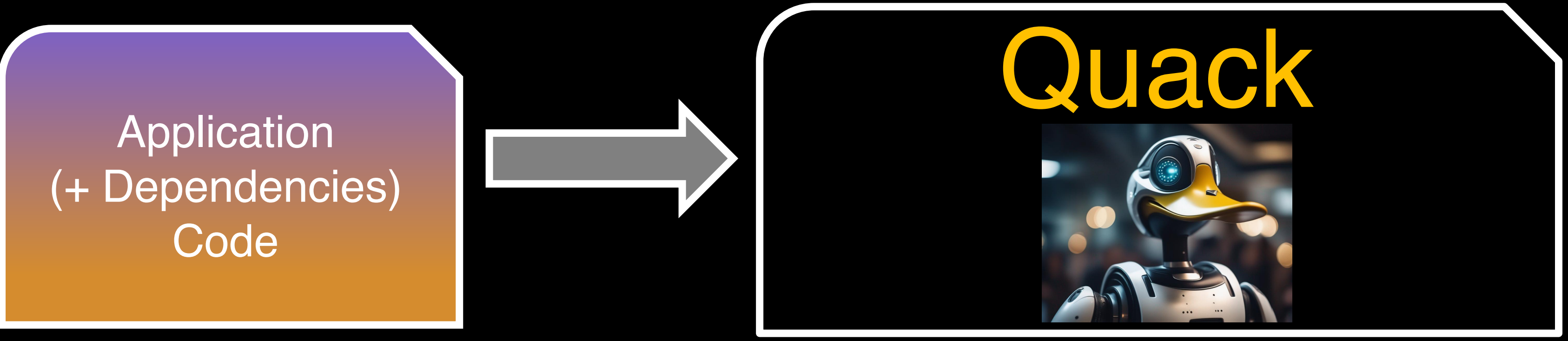
Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**

Application
(+ Dependencies)
Code

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**



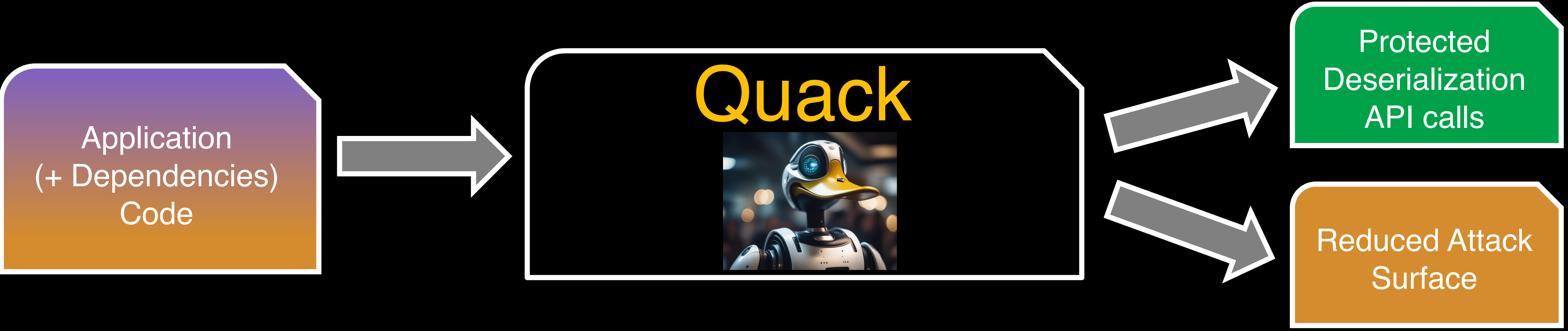
Application (+ Dependencies) Code

Quack

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**



Application (+ Dependencies) Code

Quack

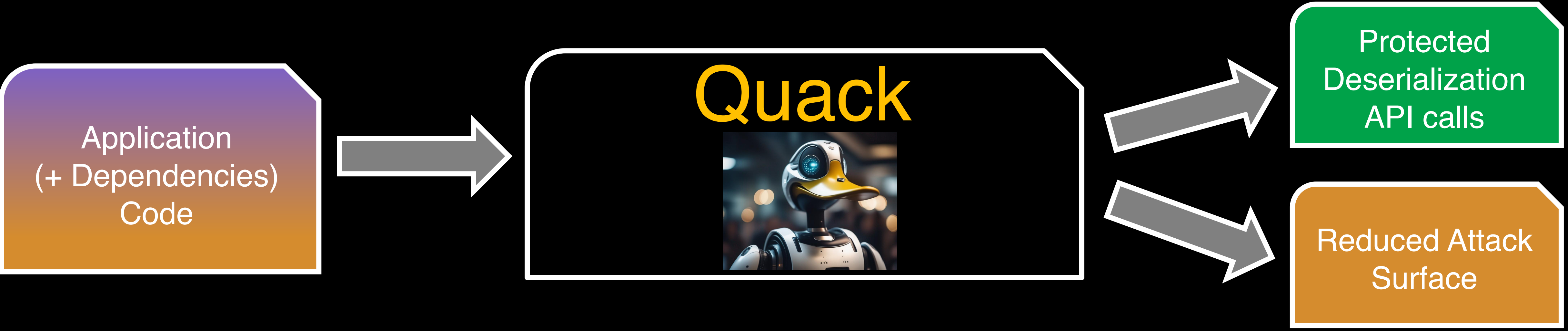
Protected Deserialization API calls

Reduced Attack Surface

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**



Application (+ Dependencies) Code

Quack

Protected Deserialization API calls

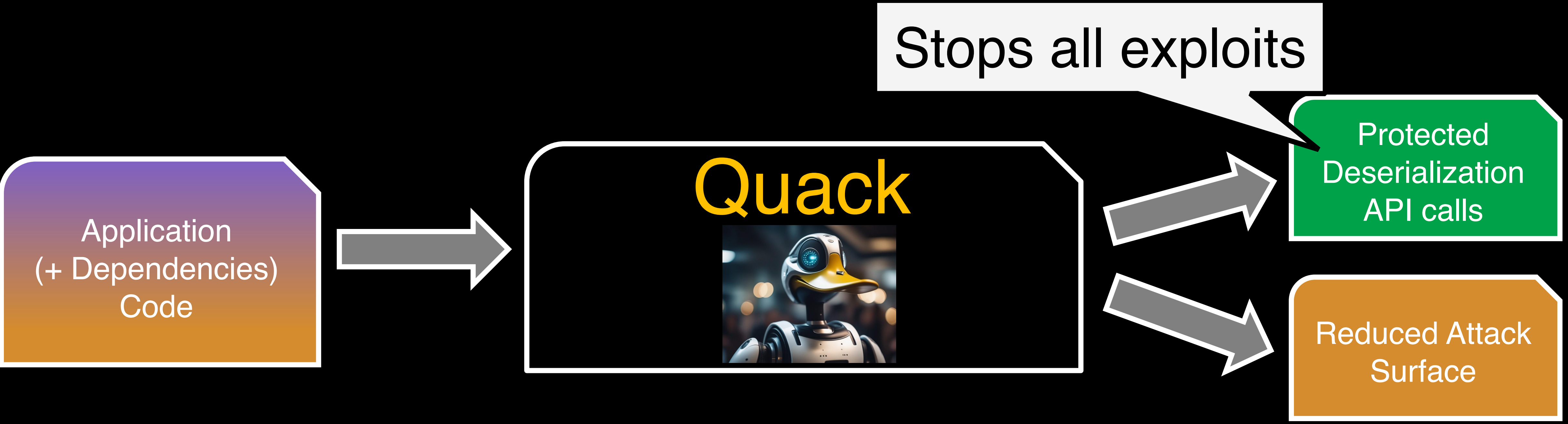
Reduced Attack Surface

Built Quack-php and evaluated on diverse set of real applications against SOTA exploit-generation-tool

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**



Stops all exploits

Application (+ Dependencies) Code

Quack

Protected Deserialization API calls

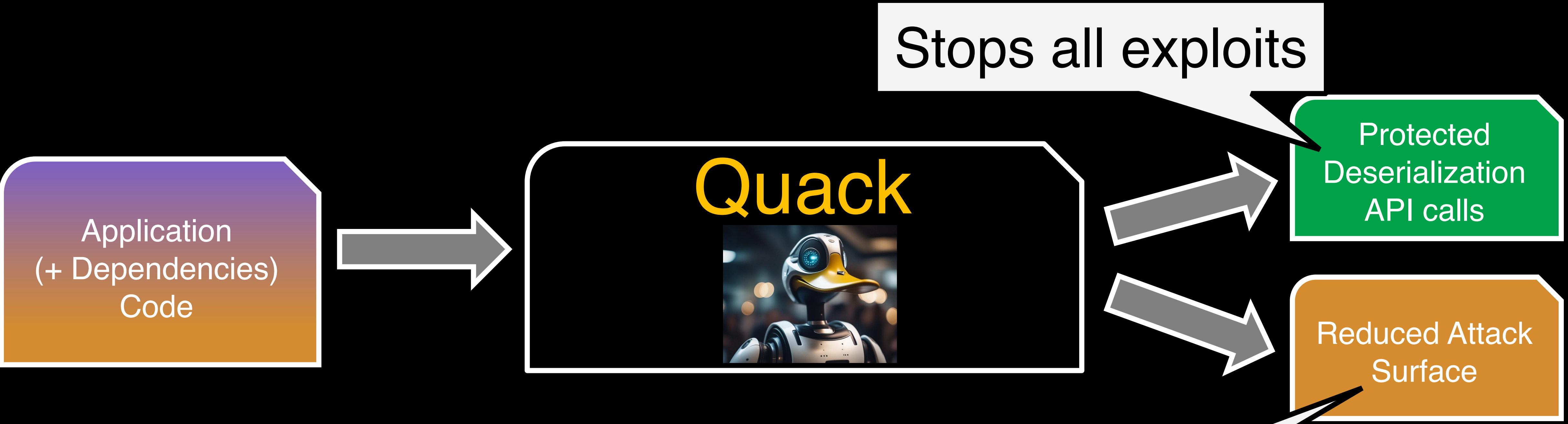
Reduced Attack Surface

Built Quack-php and evaluated on diverse set of real applications against SOTA exploit-generation-tool

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**



Stops all exploits

Application (+ Dependencies) Code

Quack

Protected Deserialization API calls

Reduced Attack Surface

Built Quack-php and eva eal applications against SC l

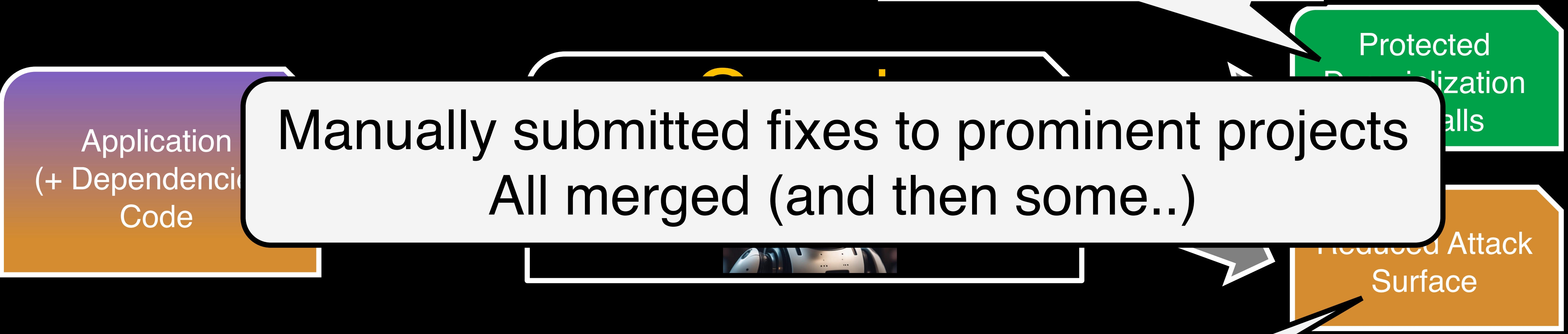
97% blocked exploit-building classes

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**

Stops all exploits



Manually submitted fixes to prominent projects
All merged (and then some..)

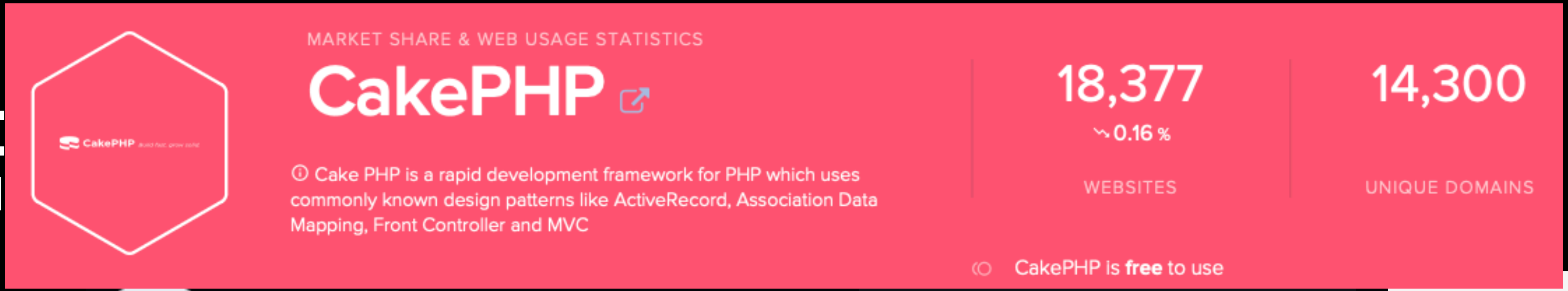
97% blocked exploit-building classes

Built Quack-php and eva eal applications
against SC l

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack:

David et. al



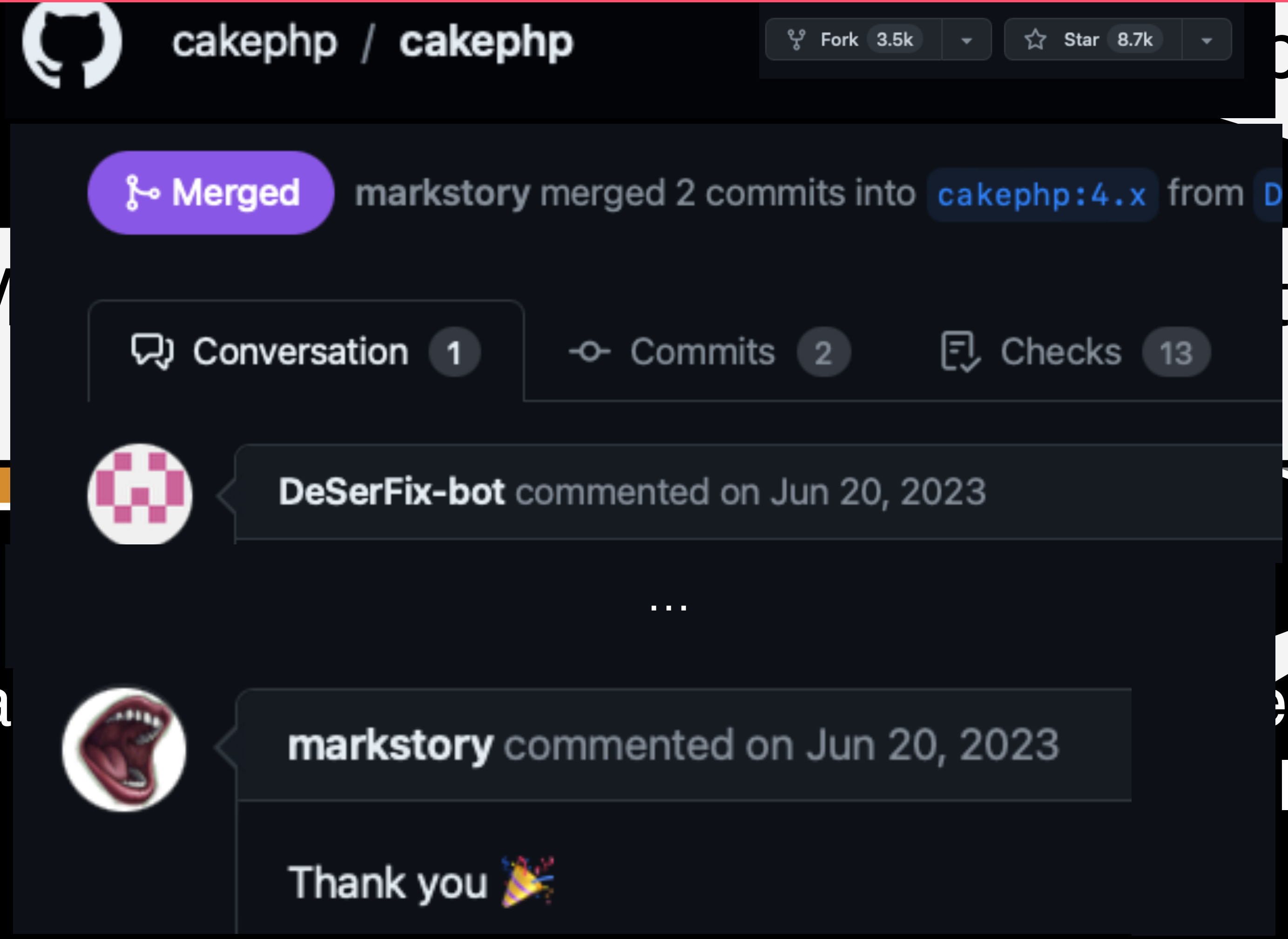
MARKET SHARE & WEB USAGE STATISTICS

CakePHP

ⓘ Cake PHP is a rapid development framework for PHP which uses commonly known design patterns like ActiveRecord, Association Data Mapping, Front Controller and MVC

18,377	14,300
↘0.16 %	
WEBSITES	UNIQUE DOMAINS

CakePHP is **free** to use



cakephp / **cakephp**

Merged markstory merged 2 commits into `cakephp:4.x` from D

DeSerFix-bot commented on Jun 20, 2023

...

markstory commented on Jun 20, 2023

Thank you 🎉

Application
(+ Dependenci
Code

M

ploits

Protected
ization
alls

projects

Attack
Surface

Built Qua

eal applications

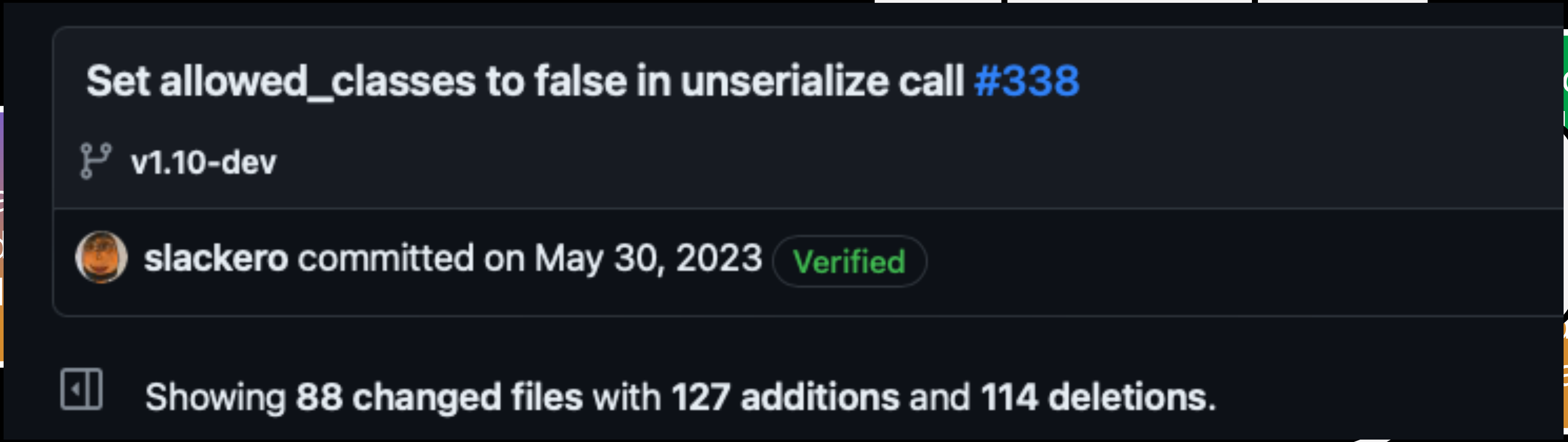
l

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

David et. al., Hindering Deserialization Attacks via Static Duck Typing **[NDSS ‘24]**

Stops all exploits



Set allowed_classes to false in unserialize call #338

v1.10-dev

slackero committed on May 30, 2023 Verified

Showing **88 changed files** with **127 additions** and **114 deletions.**

Applica
(+ Depend
Cod

cted
ization
alls

Attack
ace

Built Quack-php and eva eal applications
against SC l

97% blocked exploit-building classes

Image Source: Stable Diffusion @ stablediffusionweb.com

Why Not Limit Deserialized Classes?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```

Why Not Limit Deserialized Classes?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

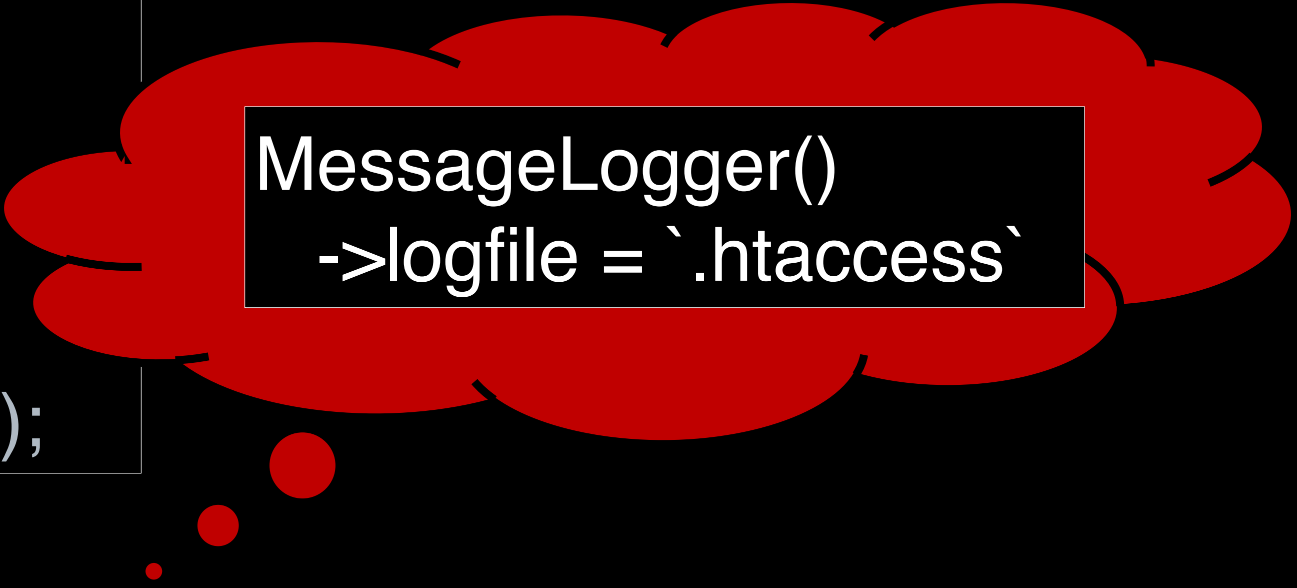
Why Not Limit Deserialized Classes?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized Classes?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```

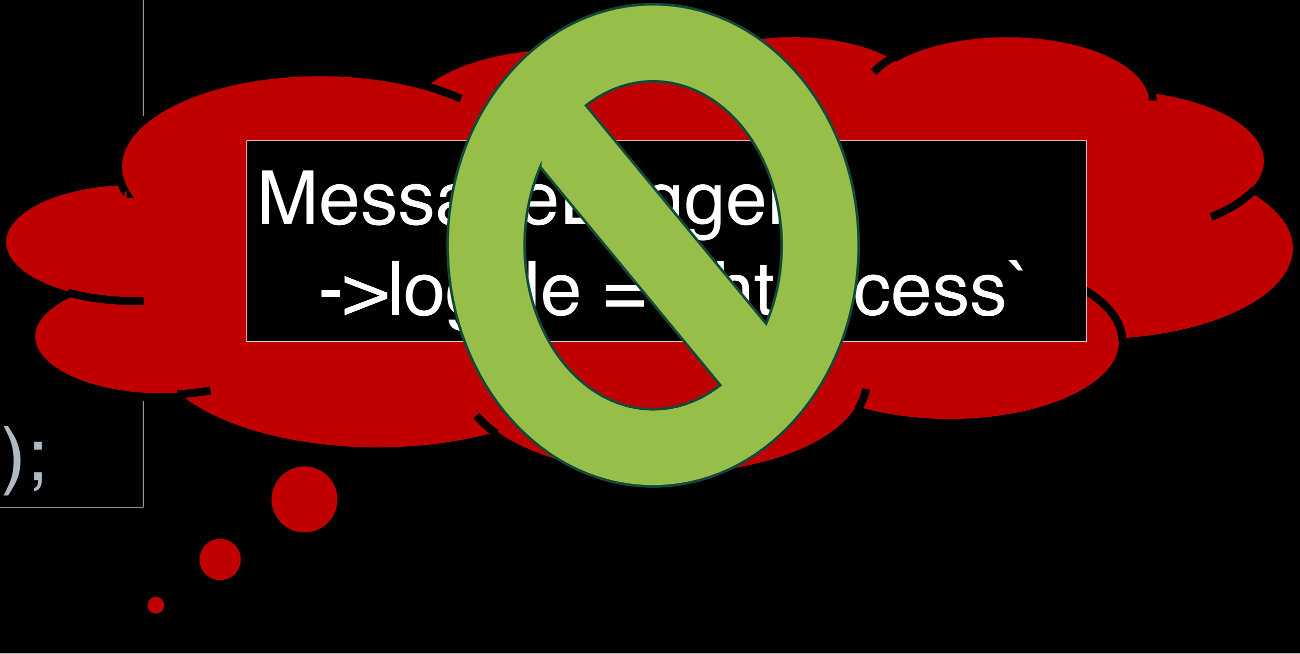


```
MessageLogger()
  ->logfile = `.htaccess`
```

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized Classes?

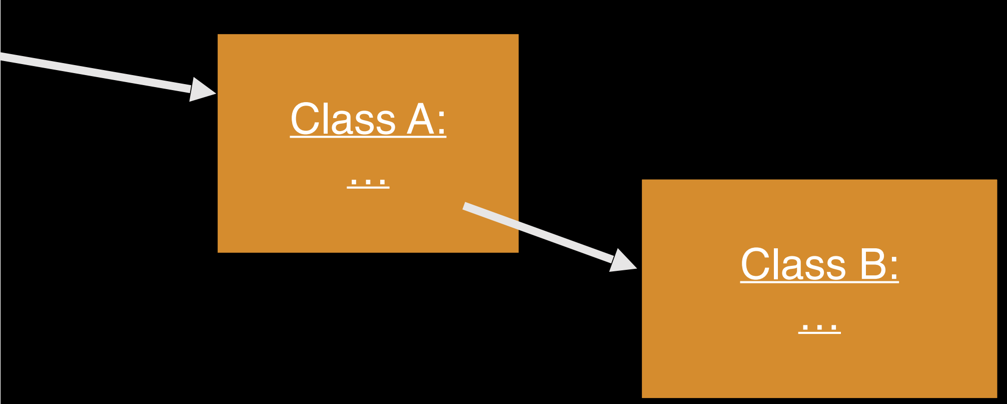
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```



```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized Classes?

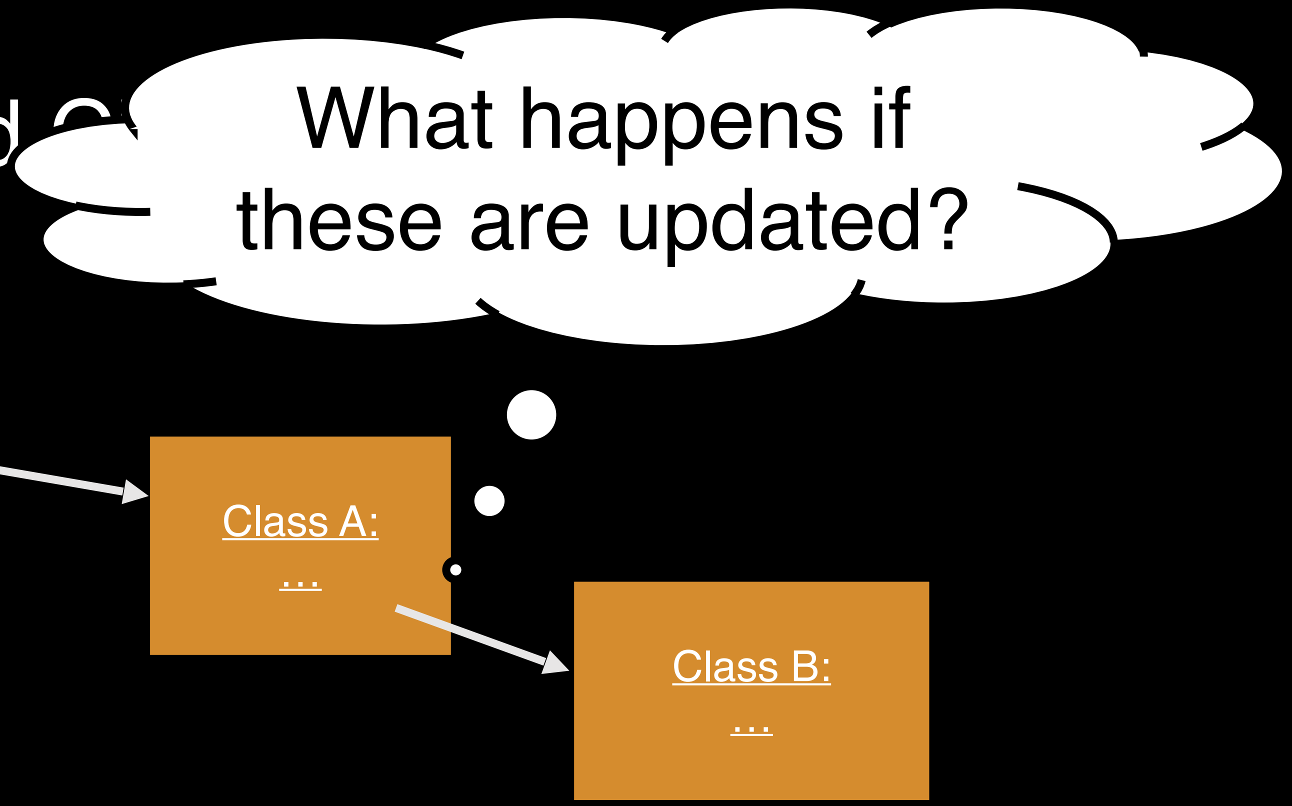
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```



```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized C

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```



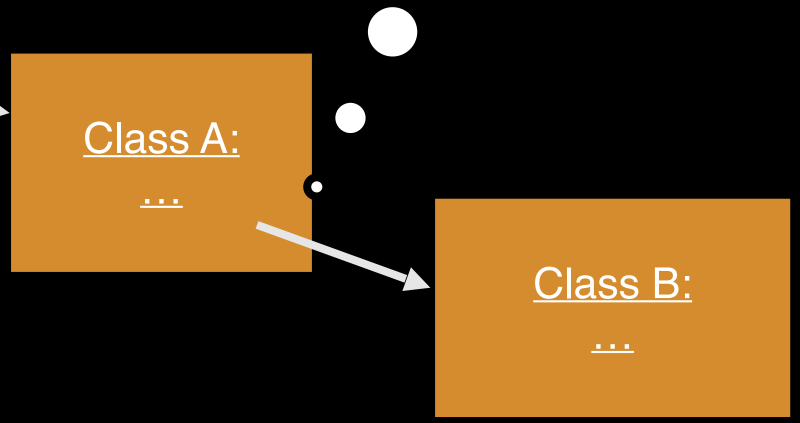
What happens if these are updated?

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized C

What happens if these are updated?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```

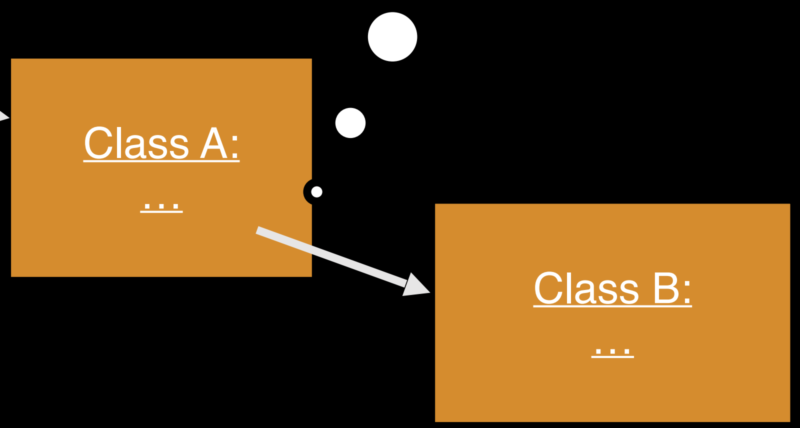


```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘E  nt’]]);
```

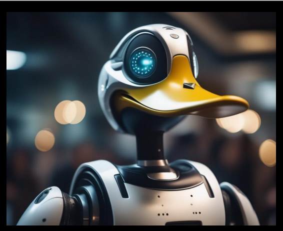
Why Not Limit Deserialized C

What happens if these are updated?

```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```



```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘E✗nt’]]);
```



Key Challenge in Protecting Deserialization API calls

Key Challenge in Protecting Deserialization API calls

- Infer all types for root objects + fields and collections

Key Challenge in Protecting Deserialization API calls

- Infer all types for root objects + fields and collections

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[???]]);
```

Key Challenge in Protecting Deserialization API calls

- Infer all types for root objects + fields and collections

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[???]]);
```

Type checkers are not geared for this

Key Challenge in Protecting Deserialization API calls

- Infer all types for root objects + fields and collections

Type inference is too conservative

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[???]]);
```

Type checkers are not geared for this

Key Challenge in Protecting Deserialization API calls

- Infer all types for root objects + fields and collections

Type inference is too conservative

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[???]]);
```

Type checkers are not geared for this

e ∈ [[Class, int, str],..

| ⊥

Insight: Infer Classes Using Object’s Usage

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>??]);
```

Insight: Infer Classes Using Object’s Usage

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>??]);
```

Type class components ≠ full type
(for security, not optimization)

Insight: Infer Classes Using Object’s Usage

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>??]);
recv_event->my_call();
something = recv_event->my_field
```

Insight: Infer Classes Using Object’s Usage

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>??]);
recv_event->my_call();
something = recv_event->my_fiel
```

Deserialized objects must contain “my_call” and “my_field”

Insight: Infer Classes Using Object’s Usage

Decision Point

```
recv_event = deserialize(ser_event, [‘allowed_classes’=>??]);
recv_event->my_call();
something = recv_event->my_fiel
```

Deserialized objects must contain “my_call” and “my_field”

Key Idea: Targeted On-Demand Class Identity Inference

- Start with the deserializable classes set
    - Conservatively handle dynamic class-loading patterns
- Gather objects (and fields, recursively) usage evidence
    - Over-approximate or know when to stop
- Use collected evidence to filter class set

Key Idea: Targeted On-Demand Class Identity Inference

- Start with the deserializable classes set
  - Conservatively handle dynamic class-loading patterns
- Gather objects (and fields, recursively) usage evidence
  - Over-approximate or know when to stop
- Use collected evidence to filter class set

Key Idea: Targeted On-Demand Class Identity Inference

- Start with the deserializable classes set
  - Conservatively handle dynamic class-loading patterns
- Gather objects (and fields, recursively) usage evidence
  - Over-approximate or know when to stop
- Use collected evidence to filter class set

Inferring structure for input objects is useful for many tasks

Analyzing Our Motivating Example

queue.php

```
/* snip */
raw_event = owa->getLast('event');
event = deserialize(raw_event);

owa::getEventDispatcher()->notify(event);
```

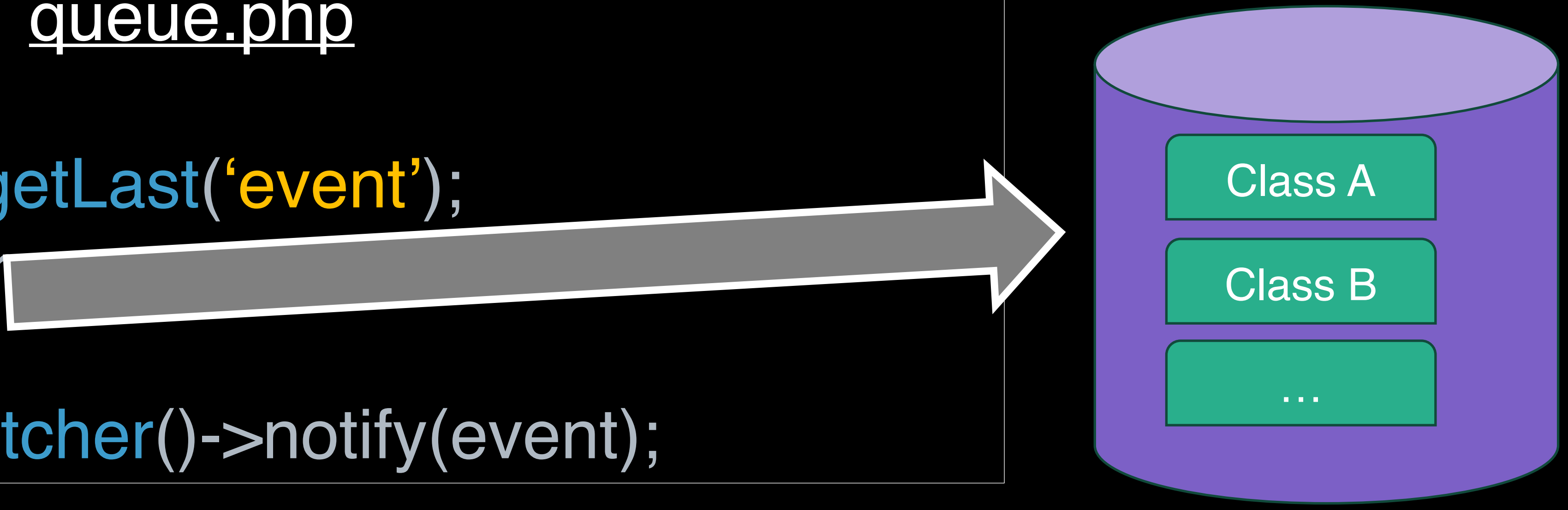
Open Web Analytics v1.5.6 containing the deserialization vulnerability CVE-2014-2294

Analyzing Our Motivating Example

queue.php

```
/* snip */
raw_event = owa->getLast('event');
event = deserialize(r

owa::getEventDispatcher()->notify(event);
```



Open Web Analytics v1.5.6 containing the deserialization vulnerability CVE-2014-2294

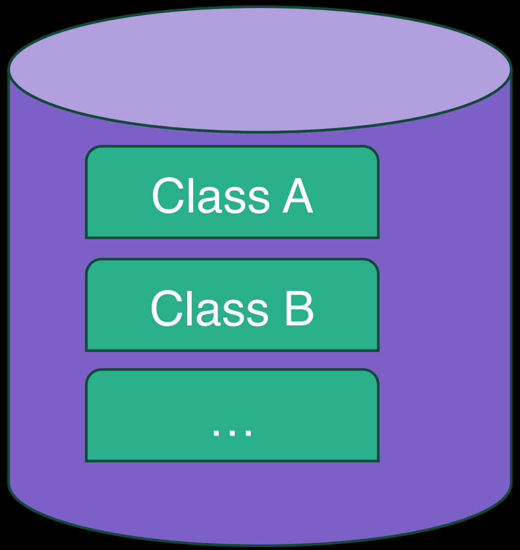
Analyzing Our Motivating Example

Root Object

queue.php

```
/* sni
raw_event = owa->getLast(‘event’);
event = deserialize(r

owa::getEventDispatcher()->notify(event);
```



Open Web Analytics v1.5.6 containing the deserialization vulnerability CVE-2014-2294

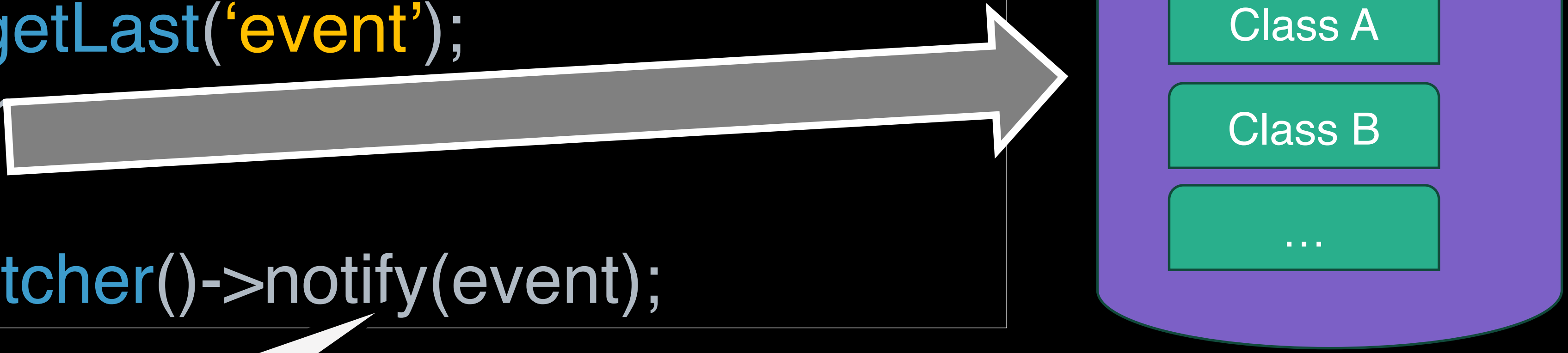
Analyzing Our Motivating Example

Root Object

queue.php

```
/* sni    /
raw_event = owa->getLast('event');
event = deserialize(r

owa::getEventDispatcher()->notify(event);
```



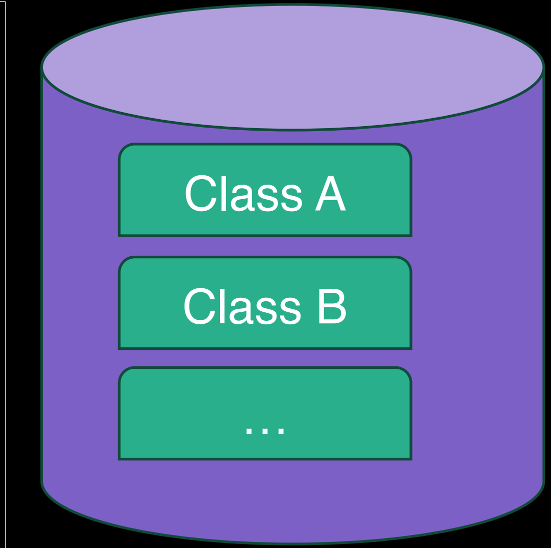
Follow

Open Web Ana containing the deserialization vulnerability CVE-2014-2294

Analyzing Our Motivating Example

eventDispatch.php

```
class owa_eventDispatcher {
 function notify(event) {
       owa_coreAPI::debug("Notifying listeners of"
                                    + event->getEventType());
 }
}
```



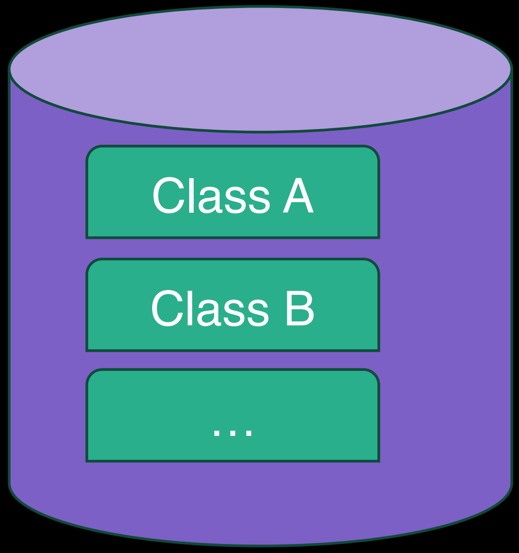
Analyzing Our Motivating Example

eventDispatch.php

```
class owa_eventDispatcher {
  function notify(event) {
        owa_coreAPI::debug("Notifying listeners of"
                                   + event->getEventType());

}
```

No Type Hint



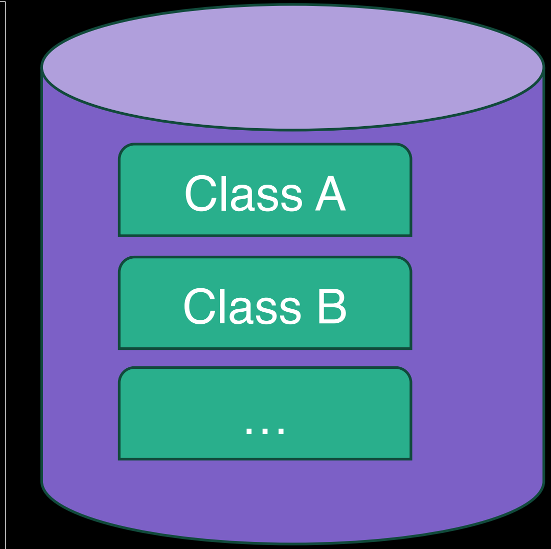
Analyzing Our Motivating Example

eventDispatch.php

```
class owa_eventDispatcher {
  function notify(event) {
        owa_coreAPI::debug("Notifying listeners of"
                                  + event->getEventType());
}
```

No Type Hint

Has method with this name



Analyzing Our Motivating Example

eventDispatch.php

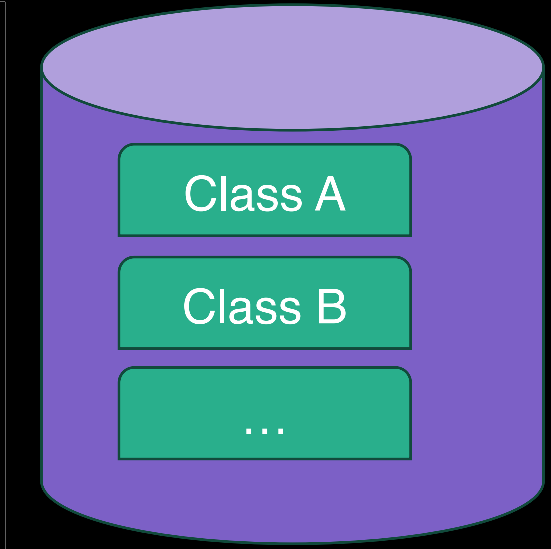
```
class owa_eventDispatcher {
  function notify(event) {
        owa_coreAPI::debug("Notifying listeners of"
                                   + event->getEventType());

}
```

No Type Hint

Has method with this name

Returns *string* or *no-return-type*



Analyzing Our Motivating Example

Rule Type	Partial Statement Matching Rule	Possible Classes
Exact	FunctionX (arg1, argi-1, t, argi+1, ...)	TypeOf(FunctionX's arg;)
	ClassXInstance MethodX (arg1, argi-1, t, argi+1, ...)	TypeOf(ClassX → MethodX's argi)
	(TypeName) t	TypeName
	Expr ? t: a (or symmetric case)	TypeOf(a)
Duck Typing	t->MethodX (...)	Classes with a method named 'MethodX'
	t.Fieldx	Classes with a field named 'FieldX'
	t <BinaryOp>a	Types allowing < BinaryOp>(e.g., "+" or ">=") with TypeOf(a)
	tOp> (or symmetric case)	Types allowing Op (e.g., "++")
	t[offset or key]	Types compatible with slicing
	a <AssignOp> t	Types allowing <AssignOp>(e.g., +=) with TypeOf(a)
	switch (t): case (a)	Types allowing equality check against TypeOf(a)

Analyzing Our Motivating Example

eventDispatch.php

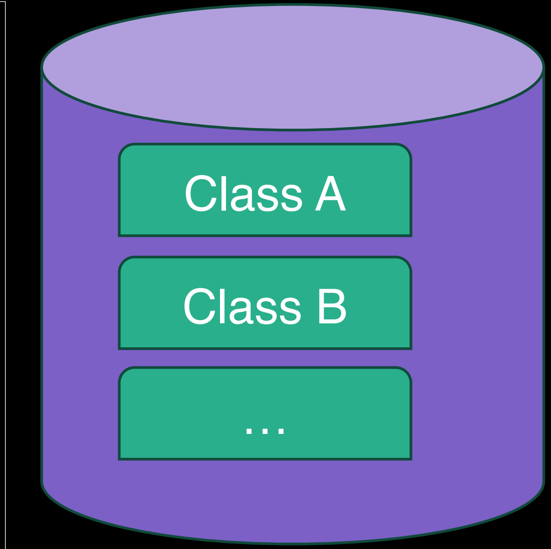
```
class owa_eventDispatcher {
  function notify(event) {
        owa_coreAPI::debug("Notifying listeners of"
                                  + event->getEventType());

}
```

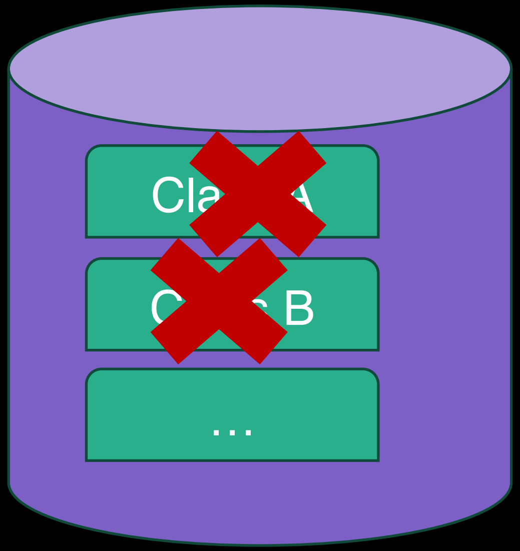
No Type Hint

Has method with this name

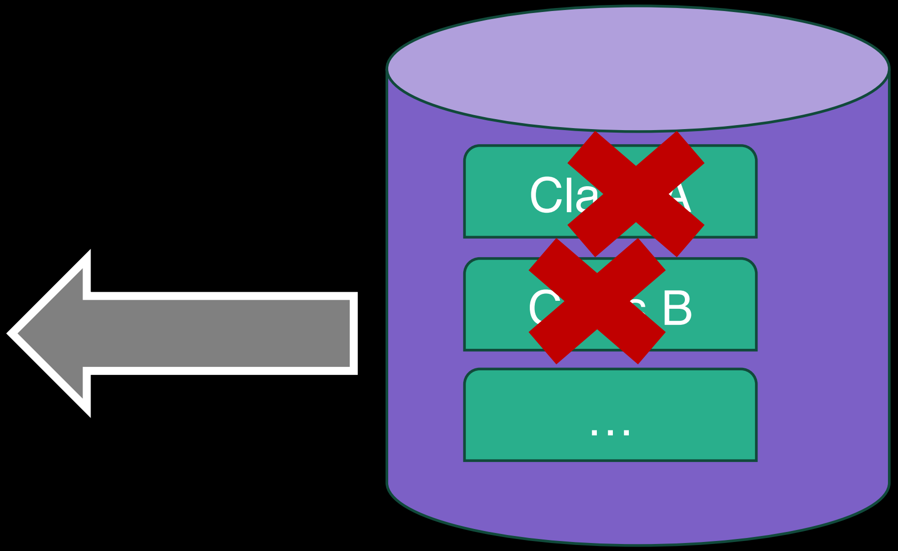
Returns *string* or *no-return-type*



Analyzing Our Motivating Example



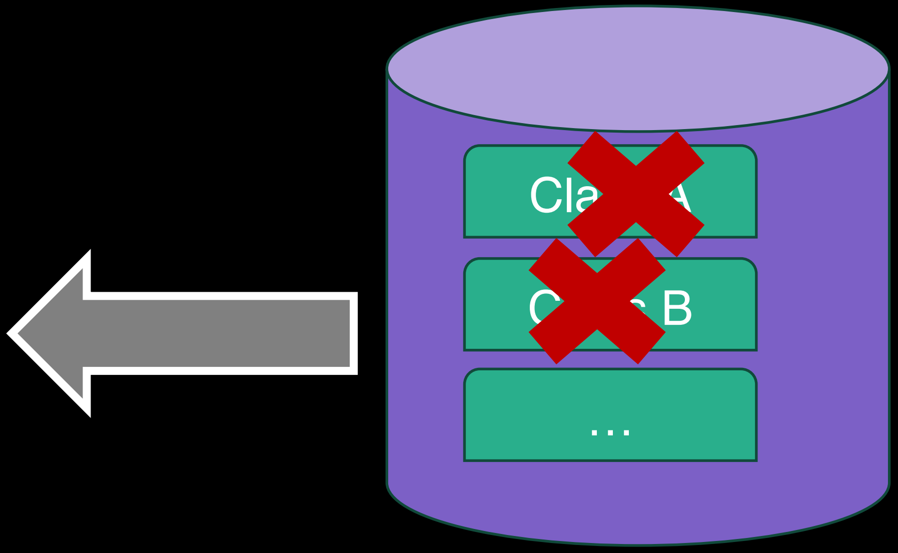
Analyzing Our Motivating Example



Analyzing Our Motivating Example

event.php

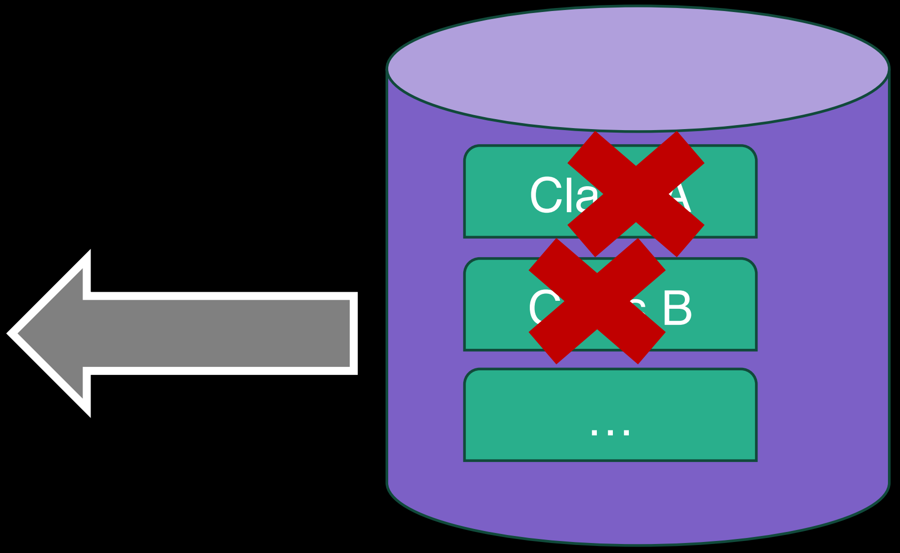
```
class owa_event {
  function getEventType () { /* snip */}
}
```



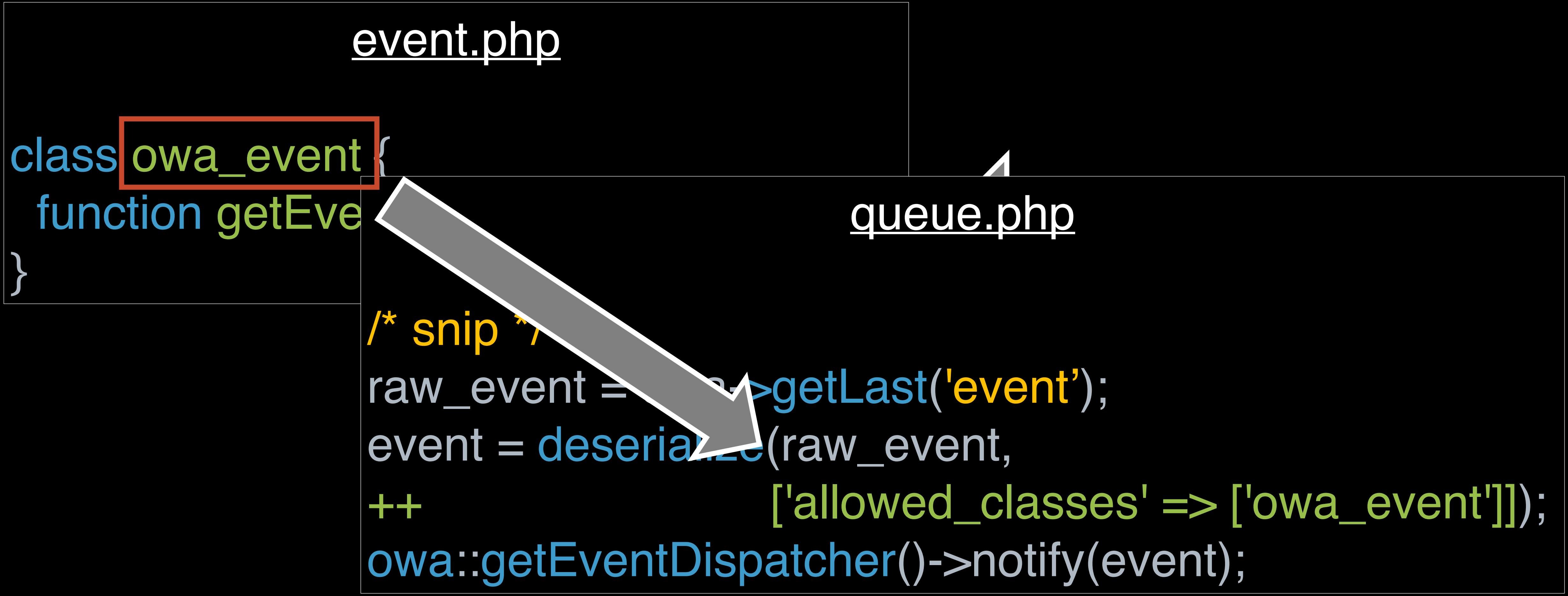
Analyzing Our Motivating Example

event.php

```
class owa_event {
  function getEventType () { /* snip */}
}
```



Analyzing Our Motivating Example



event.php

```
class owa_event {
  function getEve
}
```

queue.php

```
/* snip */
raw_event = n->getLast('event’);
event = deserialize(raw_event,
++                  ['allowed_classes' => ['owa_event']]);
owa::getEventDispatcher()->notify(event);
```

Analyzing Our Motivating Example

event.php

```
class owa_event {
  function getEve
}
```

queue.php

```
/* snip */
raw_event = [illegible]->getLast('event’);
event = deserialize(raw_event,
++                  ['allowed_classes' => ['owa_event']]);
owa::getEventDispatcher()->notify(event);
```

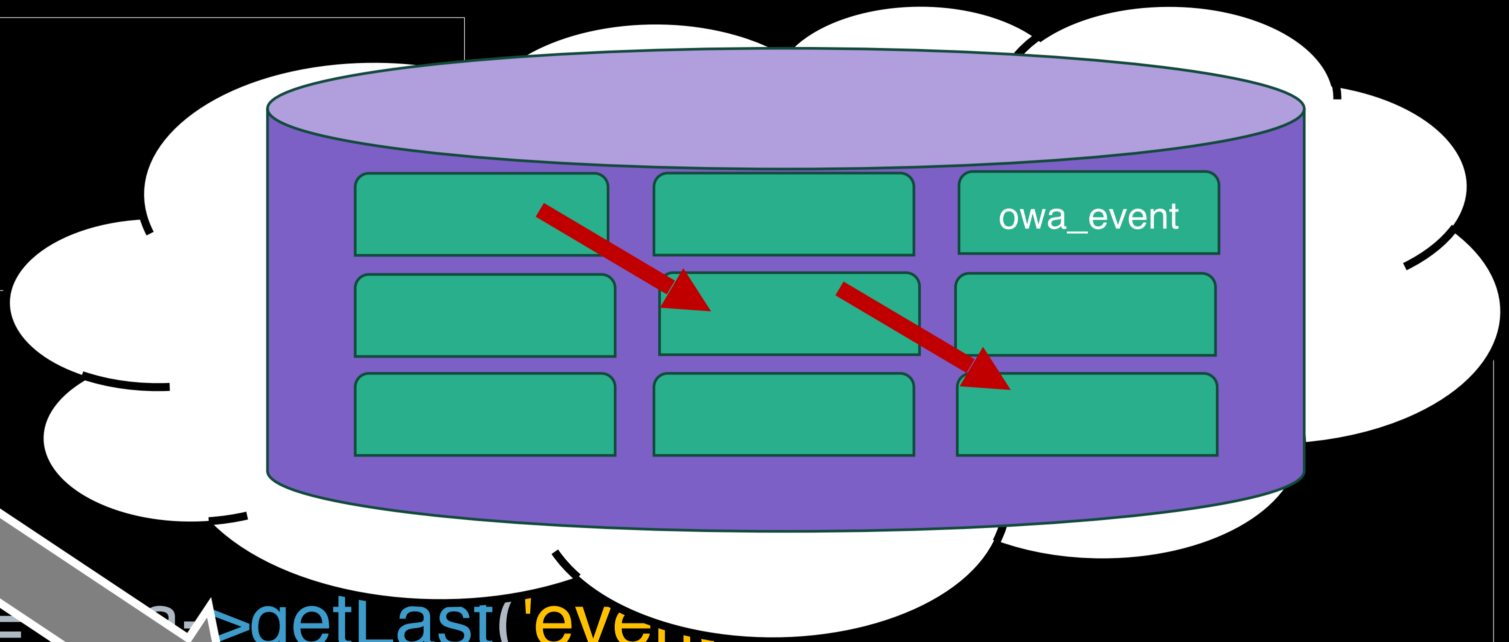
Stops 14 different exploit chains

Analyzing Our Motivating Example

event.php

```
class owa_event {
  function getEve
}
```

```
/* snip */
raw_event =     ->getLast('eve
event = deserial  (raw_ev   ,
++                 ['allowed_classes' => ['owa_event']]);
owa::getEventDi     er()->notify(event);
```



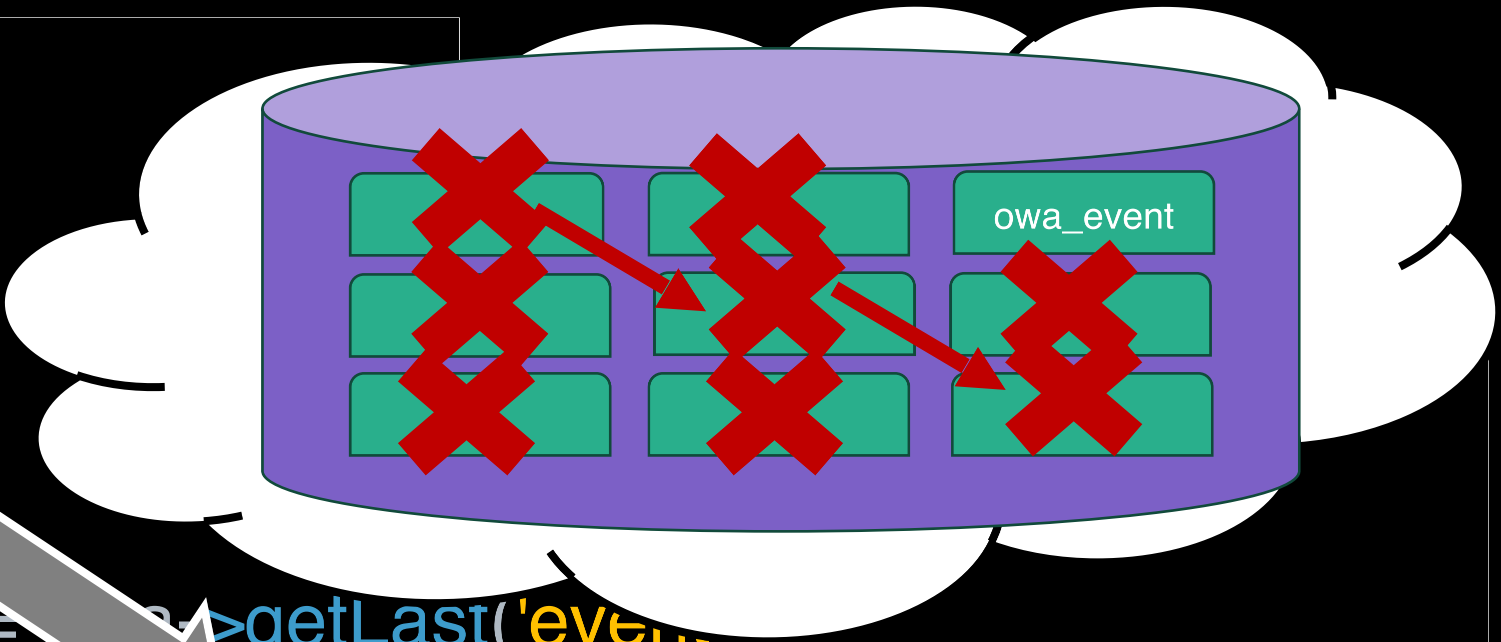
Stops 14 different exploit chains

Analyzing Our Motivating Example

event.php

```
class owa_event {
  function getEve
}
```

```
/* snip */
raw_event =  ->getLast('event');
event = deserialize(raw_ev ,
++                 ['allowed_classes' => ['owa_event']]);
owa::getEventDi   er()->notify(event);
```



Stops 14 different exploit chains

Evaluation

- Compare against **FUGIO**, SOTA automatic exploit generation tool
  - Dynamically collect available classes and composes them into a gadget-chain

- Vulnerability Datasets
  - **FUGIO** – all vulnerable php applications used by previous papers
  - **VULN202X** – A sample of PHP deserialization vulnerabilities published at/after 2020

- Measure:
  - Exploit-building classes blocked (“Positive”)
  - Classes wrongfully excluded (“Negative”)

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	7	79 (55)	100%
VULN202x	10	194(114)	97(7)%

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	7	79 (55)	100%
VULN202x	10	194(114)	97(7)%

Protecting Applications Against FUGIO

Application	CVE	# FUGIO-Generated Exploits	
Piwik 0.4.5	CVE-2009-4137		
Joomla-3.0.2	CVE-2013-1453		
CubeCart 5.2.0	CVE-2013-1465		
Contao CMS 3.2.4	CVE-2014-1860		
Open Web Analytics	CVE-2014-2294		

Protecting Applications Against FUGIO

Application	CVE	# FUGIO-Generated Exploits
		Original Version
Piwik 0.4.5	CVE-2009-4137	1
Joomla-3.0.2	CVE-2013-1453	2
CubeCart 5.2.0	CVE-2013-1465	1
Contao CMS 3.2.4	CVE-2014-1860	5
Open Web Analytics	CVE-2014-2294	14

Protecting Applications Against FUGIO

Application	CVE	# FUGIO-Generated Exploits	
		Original Version	Quack-Protected Version
Piwik 0.4.5	CVE-2009-4137	1	0
Joomla-3.0.2	CVE-2013-1453	2	0
CubeCart 5.2.0	CVE-2013-1465	1	0
Contao CMS 3.2.4	CVE-2014-1860	5	0
Open Web Analytics	CVE-2014-2294	14	0

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	7	79 (55)	100%
VULN202x	10	194(114)	97(7)%

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	7	79 (55)	100%
VULN202x	10	194(114)	97(7)%

Protecting Against Recent CVEs

Application	CVE	Exploit-Building Classes		
		#Blocked	#Remaining	%Blocked
ForkCMS 5.8.3	2020-24036	221	23	91%
WP-hotel-booking 10.2.1	2020-29047	103	0	100%
OpenCATS-0.9.5 (1)	2021-25294	288	0	100%
OpenCATS-0.9.5 (2)		232	56	81%
OpenCATS-0.9.5 (3)		288	0	100%
OpenCATS-0.9.5 (4)		288	0	100%
OpenCATS-0.9.5 (5)		232	56	81%
WP-AIOSEO 4.1.0.1	2021-24307	23	0	100%
WP-booking-calendar 9.1.1	2022-1463	96	0	100%
WP-lead-generated 1.23	2023-28667	40	0	100%

Protecting Against Recent CVEs

Application	CVE	Exploit-Building Classes		
		#Blocked	#Remaining	%Blocked
ForkCMS 5.8.3	2020-24036	221	23	91%
WP-hotel-booking 10.2.1	2020-29047	103	0	100%
OpenCATS-0.9.5 (1)	2021-25294	288	0	100%
OpenCATS-0.9.5 (2)		232	56	81%
OpenCATS-0.9.5 (3)		288	0	100%
OpenCATS-0.9.5 (4)		288	0	100%
OpenCATS-0.9.5 (5)		232	56	81%
WP-AIOSEO 4.1.0.1	2021-24307	23	0	100%
WP-booking-calendar 9.1.1	2022-1463	96	0	100%
WP-lead-generated 1.23	2023-28667	40	0	100%

Protecting Against Recent CVEs

Application	CVE	Exploit-Building Classes		
		#Blocked	#Remaining	%Blocked
ForkCMS 5.8.3	2020-24036	221	23	91%
WP-hotel-booking 10.2.1	2020-29047	10[illegible]	0	100%
OpenCATS-0.9.5 (1)		28[illegible]	0	100%
OpenCATS-0.9.5 (2)		[illegible]	[illegible]	81%
OpenCATS-0.9.5 (3)	[illegible]	[illegible]	[illegible]	[illegible]
OpenCATS-0.9.5 (4)	[illegible]	[illegible]	[illegible]	[illegible]
OpenCATS-0.9.5 (5)	[illegible]	[illegible]	[illegible]	[illegible]
WP-AIOSEO 4.1.0.1	[illegible]	[illegible]	[illegible]	100%
WP-booking-calendar 9.1.1	20[illegible]	[illegible]	[illegible]	100%
WP-lead-generated 1.23	2023-28667	40	0	100%

```
call_user_func_array([$class, $method],...)

new $class(...)
```

Performance and Safety

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	7	79 (55)	100%
VULN202x	10	212(106)	97(7)%

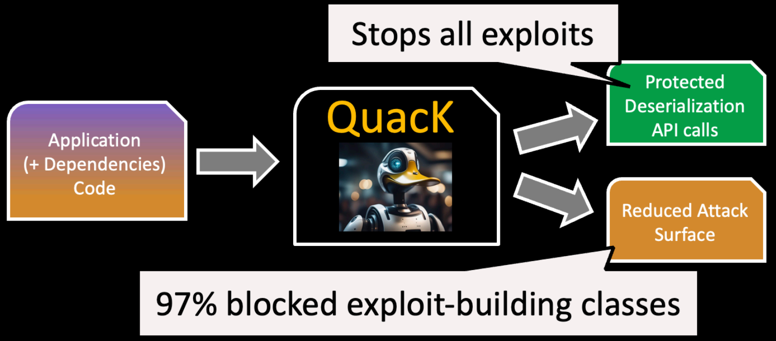
- Favor soundness → **no Negatives**
- Offline project scan: **< 7 minutes**
- Enforcement incurs **negligible overheads**

https://github.com/columbia/quack

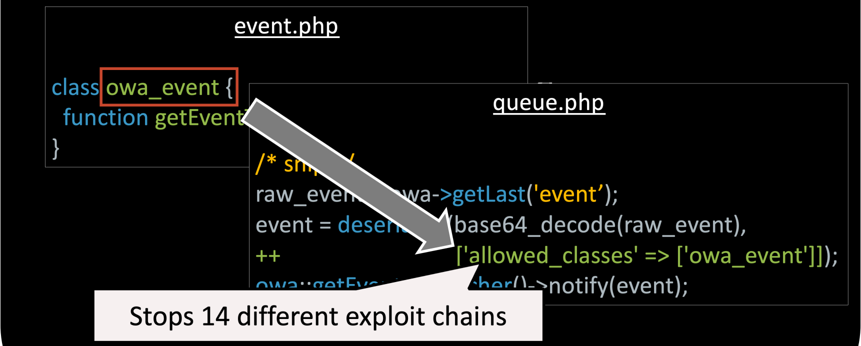
Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

Thank You!
Questions?

Quack: Securing Applications Against Deserialization Attacks


Stops all exploits
Application (+ Dependencies) Code
QuacK
Protected Deserialization API calls
Reduced Attack Surface
97% blocked exploit-building classes

Stopping Deserialization Exploits



event.php

```
class owa_event {
 function getEvent
}
```

queue.php

```
/* sn     /
raw_even      wa->getLast('event’);
event = deser      (base64_decode(raw_event),
++                  ['allowed_classes' => ['owa_event']]);
owa::getEv       her()->notify(event);
```

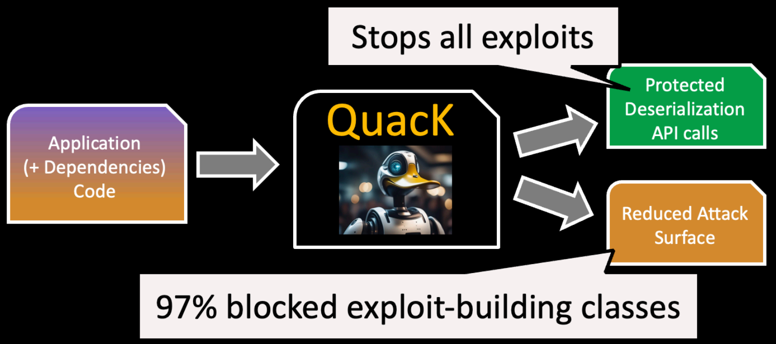
Stops 14 different exploit chains

https://github.com/columbia/quack

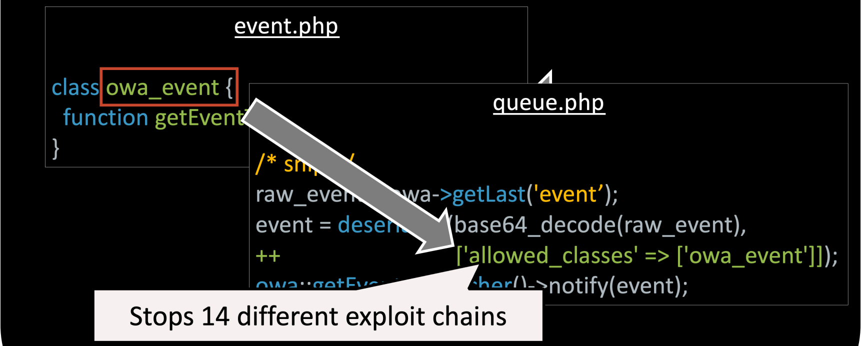
Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

Thank You!
Questions?

Quack: Securing Applications Against Deserialization Attacks



Stopping Deserialization Exploits


event.php

```
class owa_event {
 function getEvent
}
```

queue.php

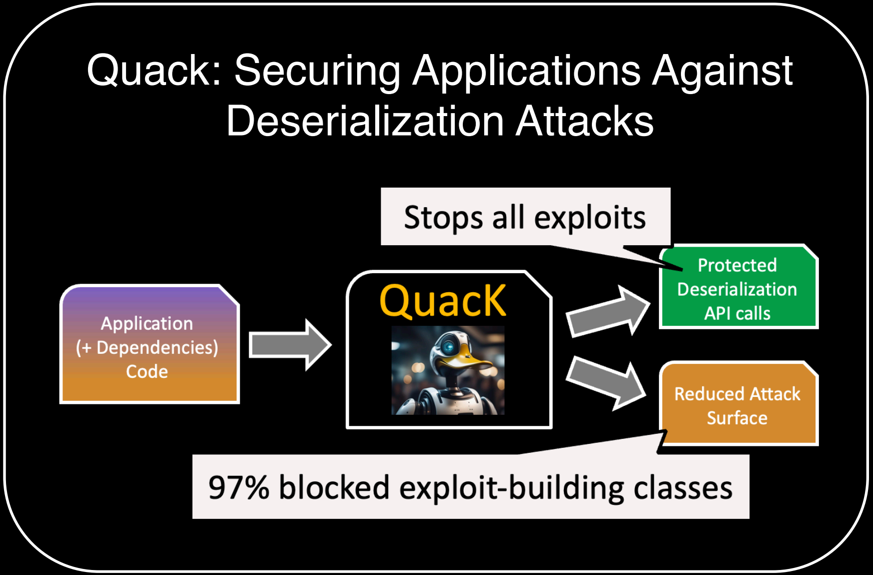
```
/* sn   /
raw_even    wa->getLast('event’);
event = deseri     (base64_decode(raw_event),
++                ['allowed_classes' => ['owa_event']]);
owa::getEve       cher()->notify(event);
```

Stops 14 different exploit chains

https://github.com/columbia/quack

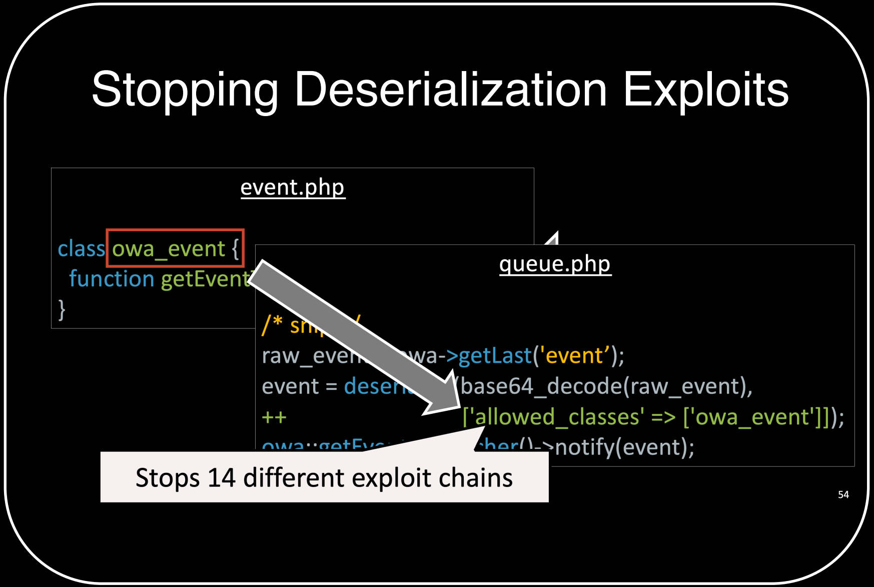
Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

Thank You!
Questions?



Quack: Securing Applications Against Deserialization Attacks

Stops all exploits
Application (+ Dependencies) Code
QuacK
Protected Deserialization API calls
Reduced Attack Surface
97% blocked exploit-building classes



Stopping Deserialization Exploits

event.php

```
class owa_event {
 function getEvent
}
```

queue.php

```
/* sn   /
raw_even    wa->getLast('event’);
event = deseri    (base64_decode(raw_event),
++                  ['allowed_classes' => ['owa_event']]);
owa::getEv         her()->notify(event);
```

Stops 14 different exploit chains

54