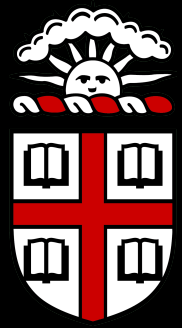
QUACK: Hindering Deserialization Attacks via Static Duck Typing

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?

Background: What's the Deal With Serialization?

Background: What's the Deal With Serialization?

Background: What's the Deal With Serialization?

Background: What's the Deal With Serialization?

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

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

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);
```

Event()
->wrapped_obj=…

nalytics Server

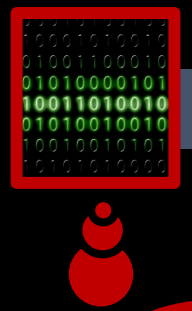
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
recv_event =
    deserialize(ser_event);
```

Exploiting A Deserialization Vulnerability

Analytics Server

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

Exploiting A Deserialization Vulnerability

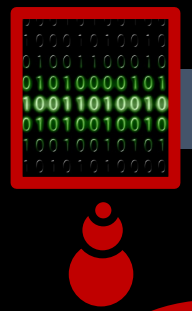


Analytics Server

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

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

Exploiting A Deserialization Vulnerability



MessageLogger()
-> logfile = `.htaccess`

Analytics Server

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

LogginClass.MessageLogger

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

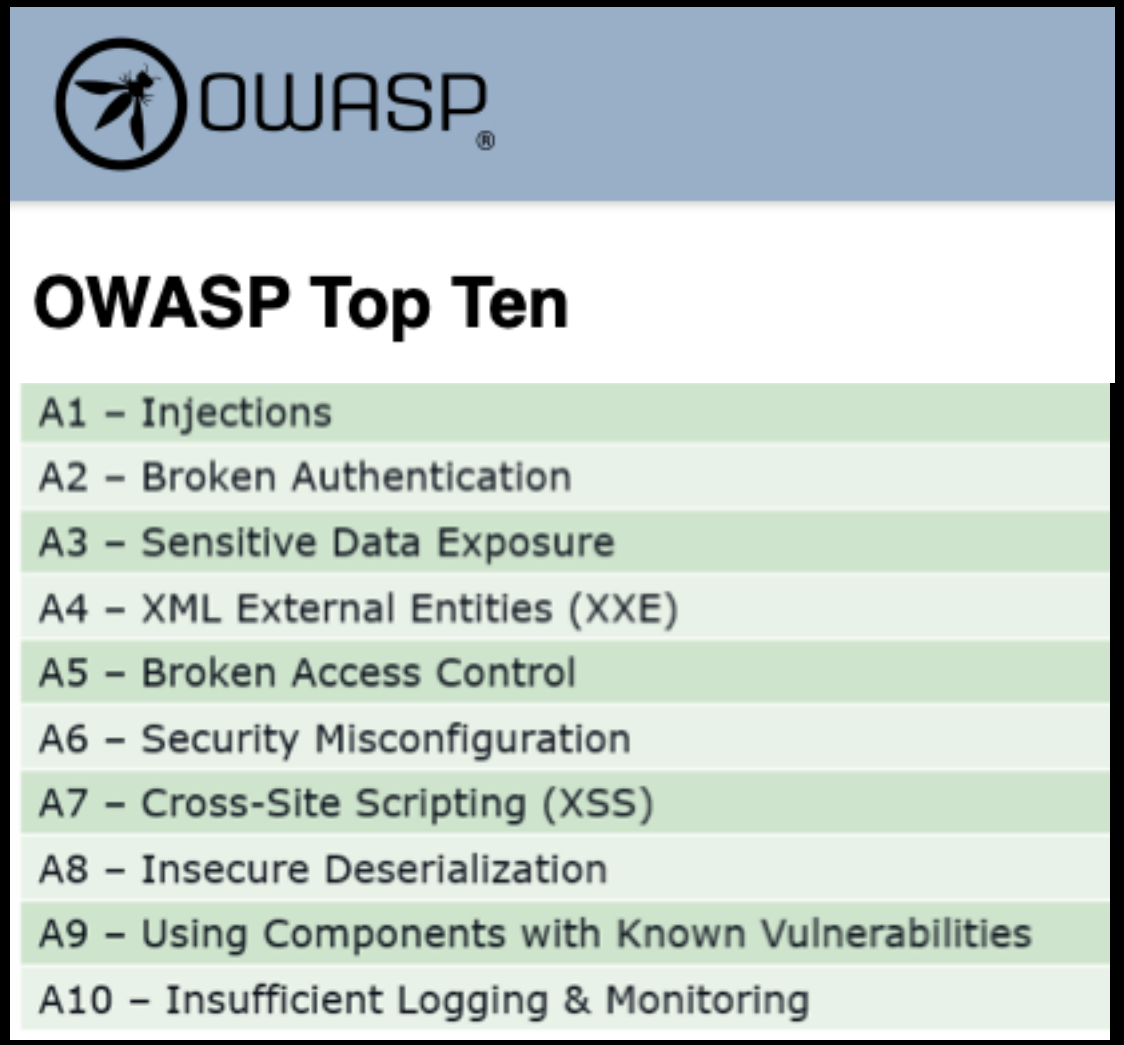
```
recv_event =
    deserialize(ser_event);
```

LogginClass.MessageLogger

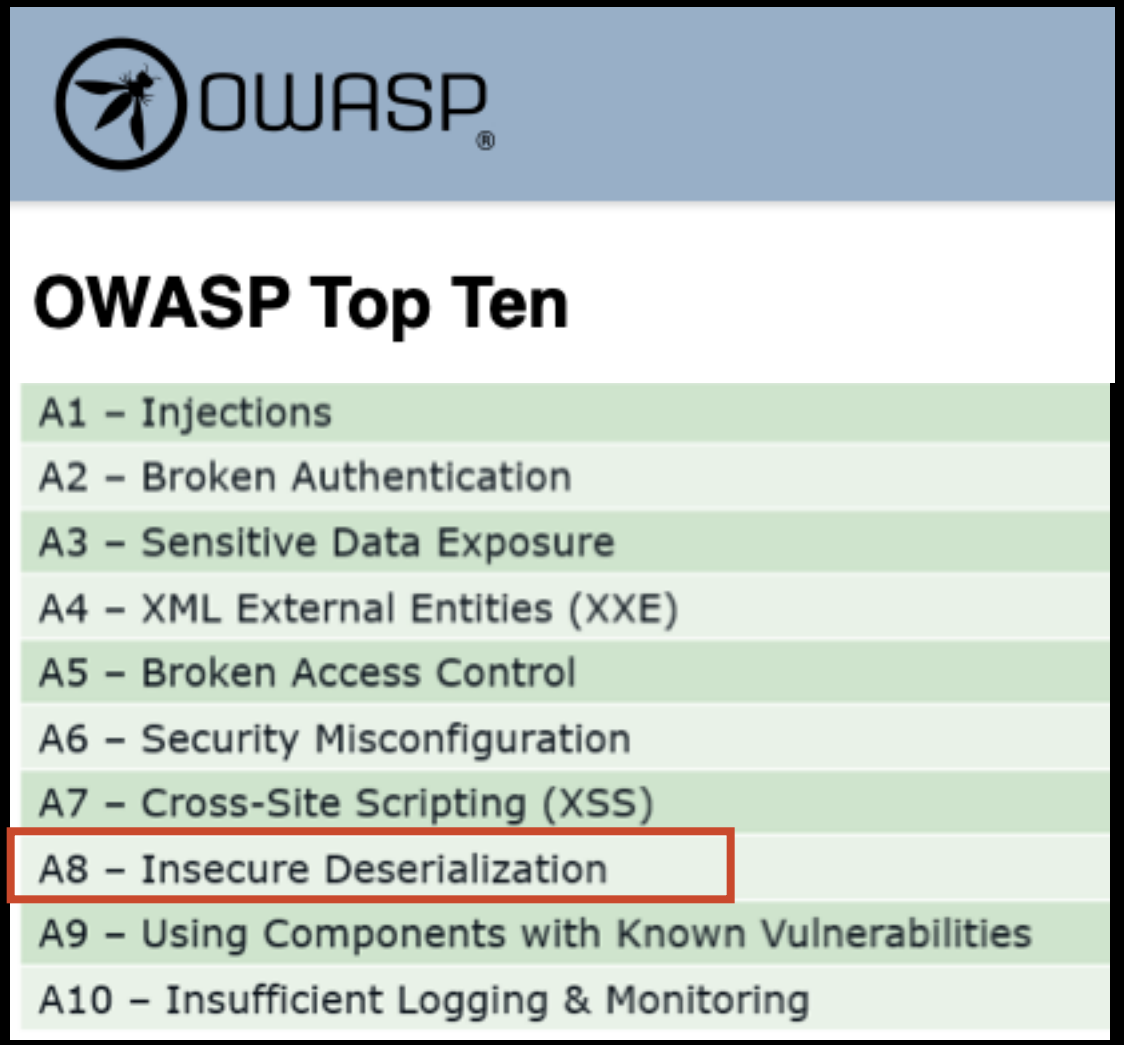
```
MessageLogger()
   -> logfile = `.htaccess`
```

Deserialization Attacks Affect Real-World Applications

Deserialization Attacks Affect Real-World Applications

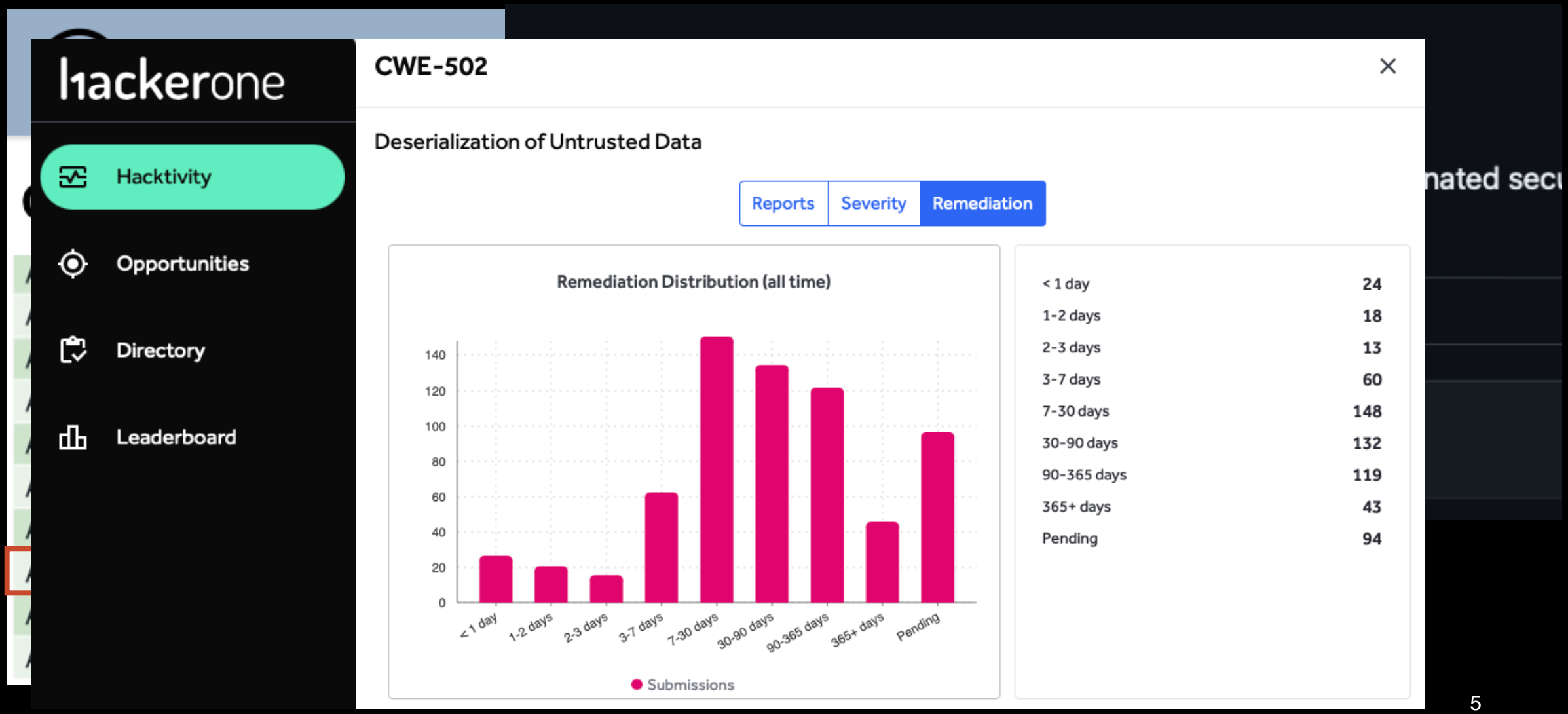


Deserialization Attacks Affect Real-World Applications

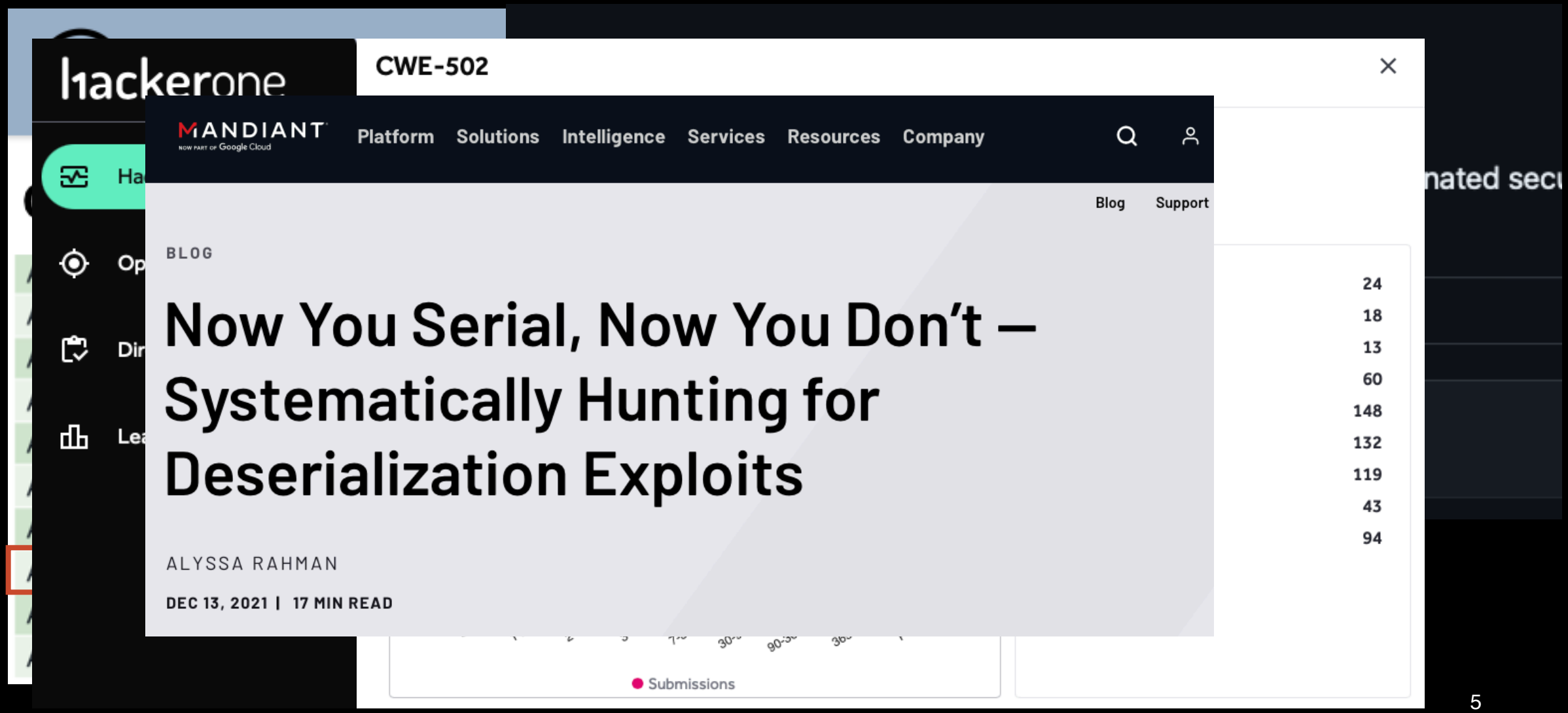


Deserialization Attacks Affect Real-World Applications

Deserialization Attacks Affect Real-World Applications

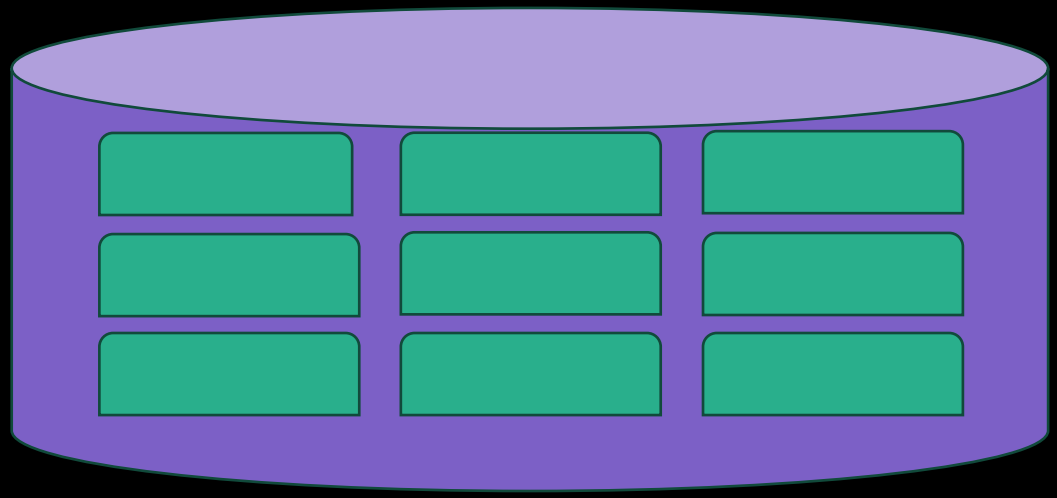


Deserialization Attacks Affect Real-World Applications

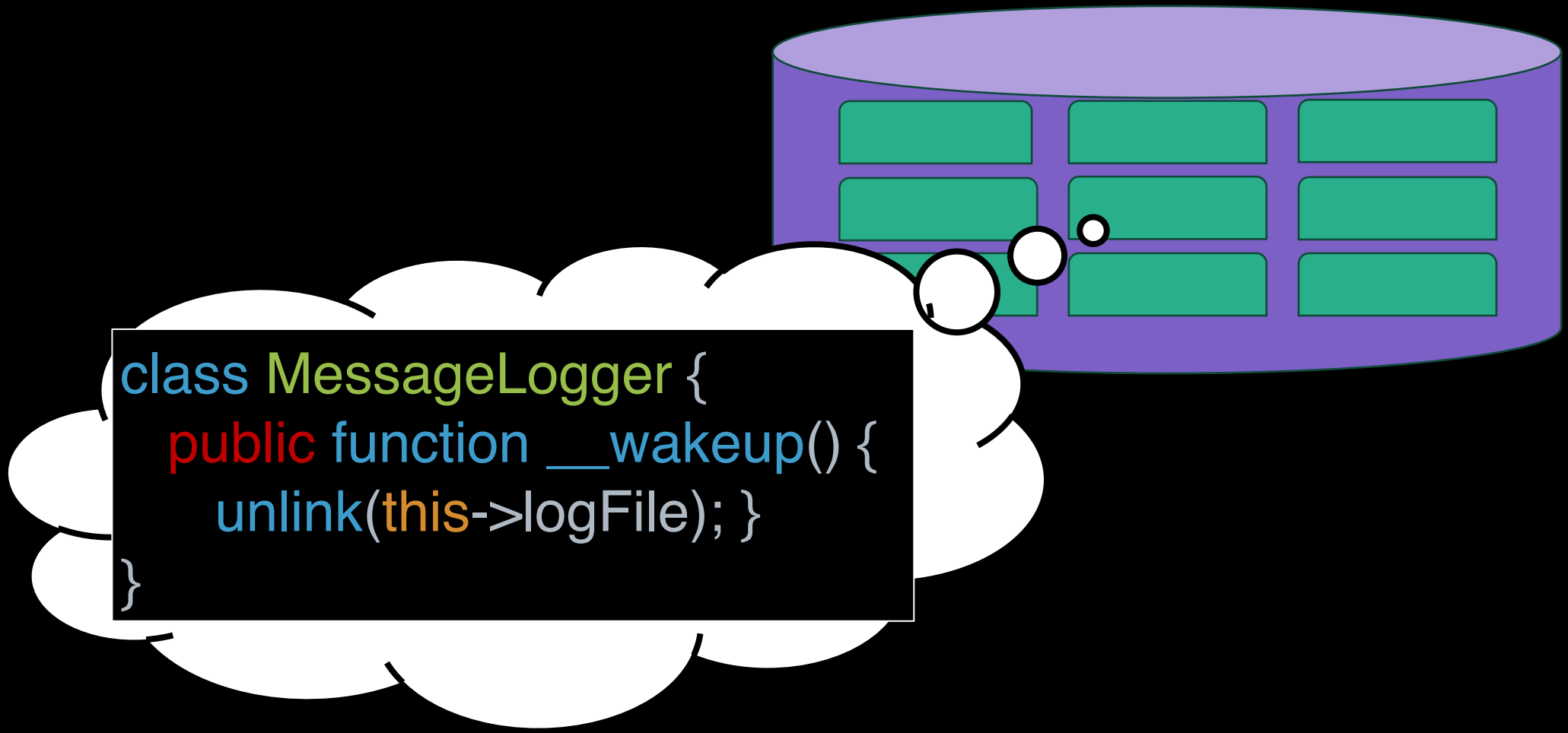


Exploit Generation Techniques Are Improving

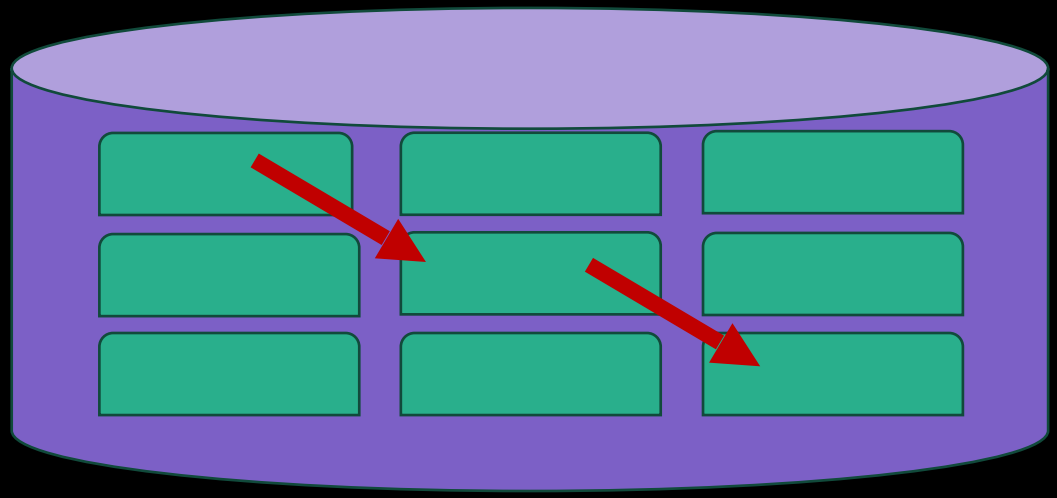
Exploit Generation Techniques Are Improving



Exploit Generation Techniques Are Improving



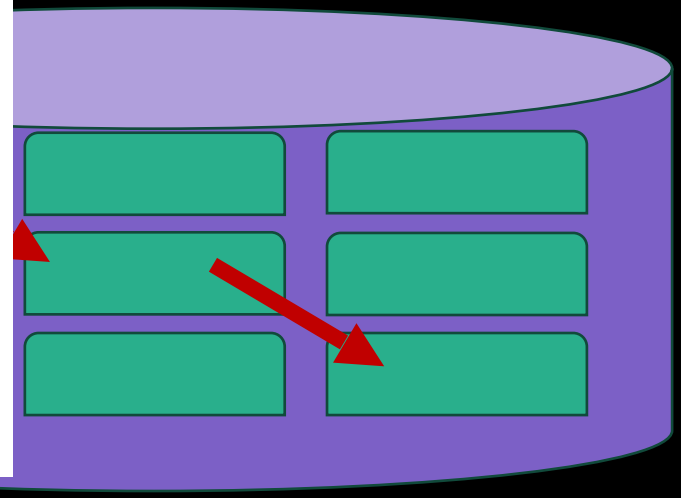
Exploit Generation Techniques Are Improving



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

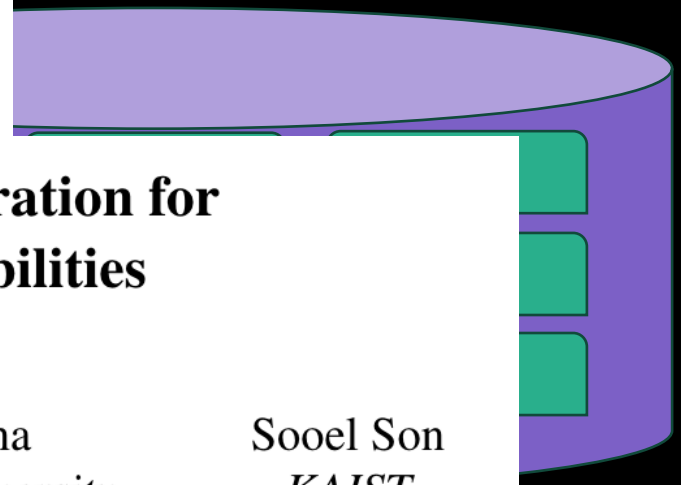
FUGIO: Automatic Exploit Generation for PHP Object Injection Vulnerabilities

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

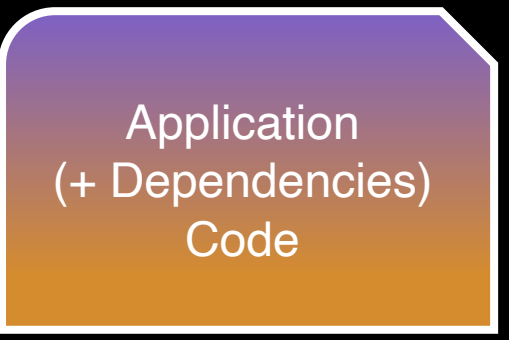
Sunny
KA

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



Quack: Hindering Deserialization Attacks

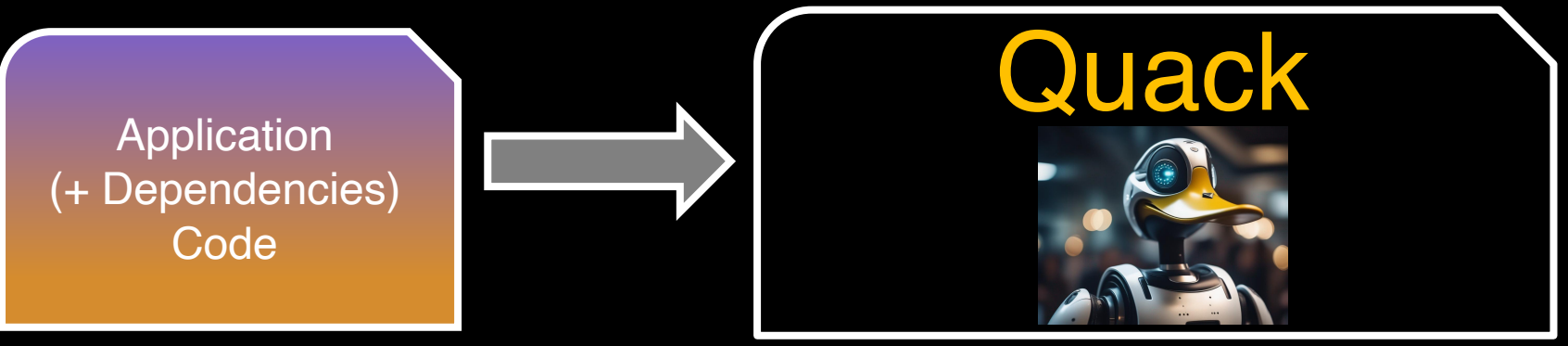


Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

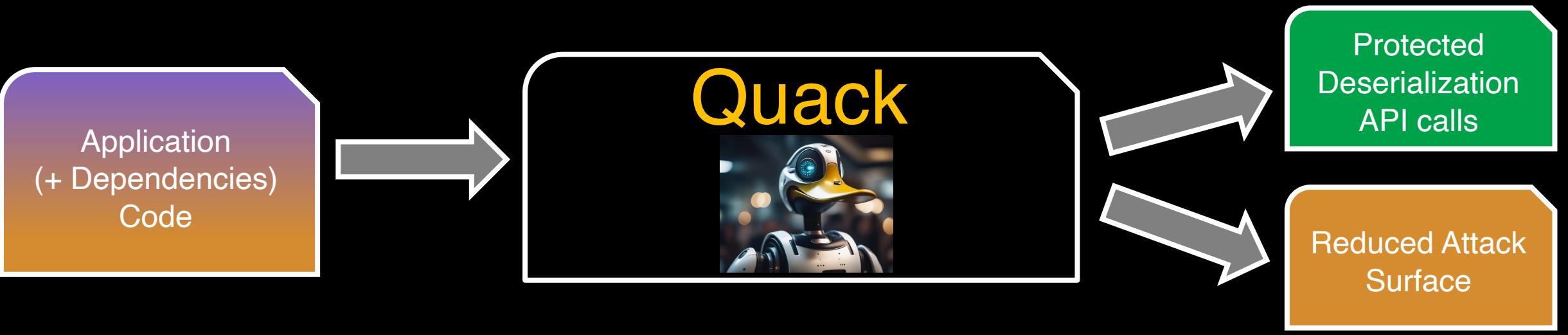
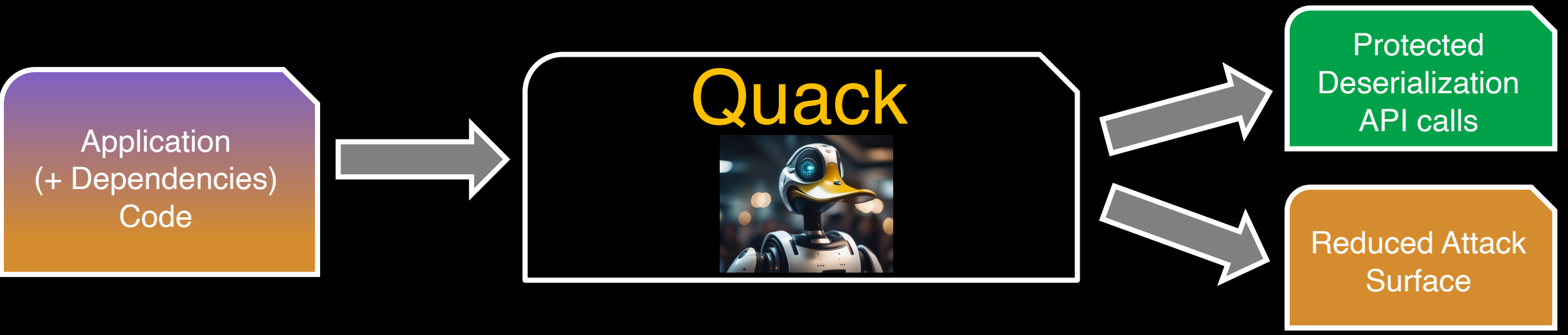


Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks



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

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

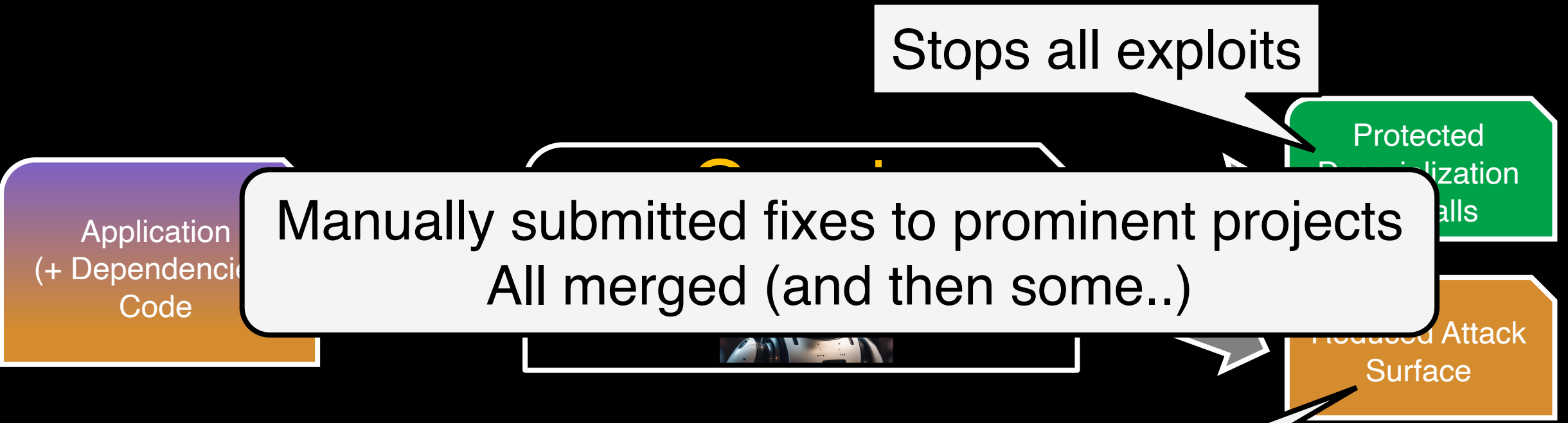
Image Source: Stable Diffusion @ stablediffusionweb.com

Quack: Hindering Deserialization Attacks

Built Quack-php and eva eal applications
against SC

Image Source: Stable Diffusion @ stablediffusionweb.com

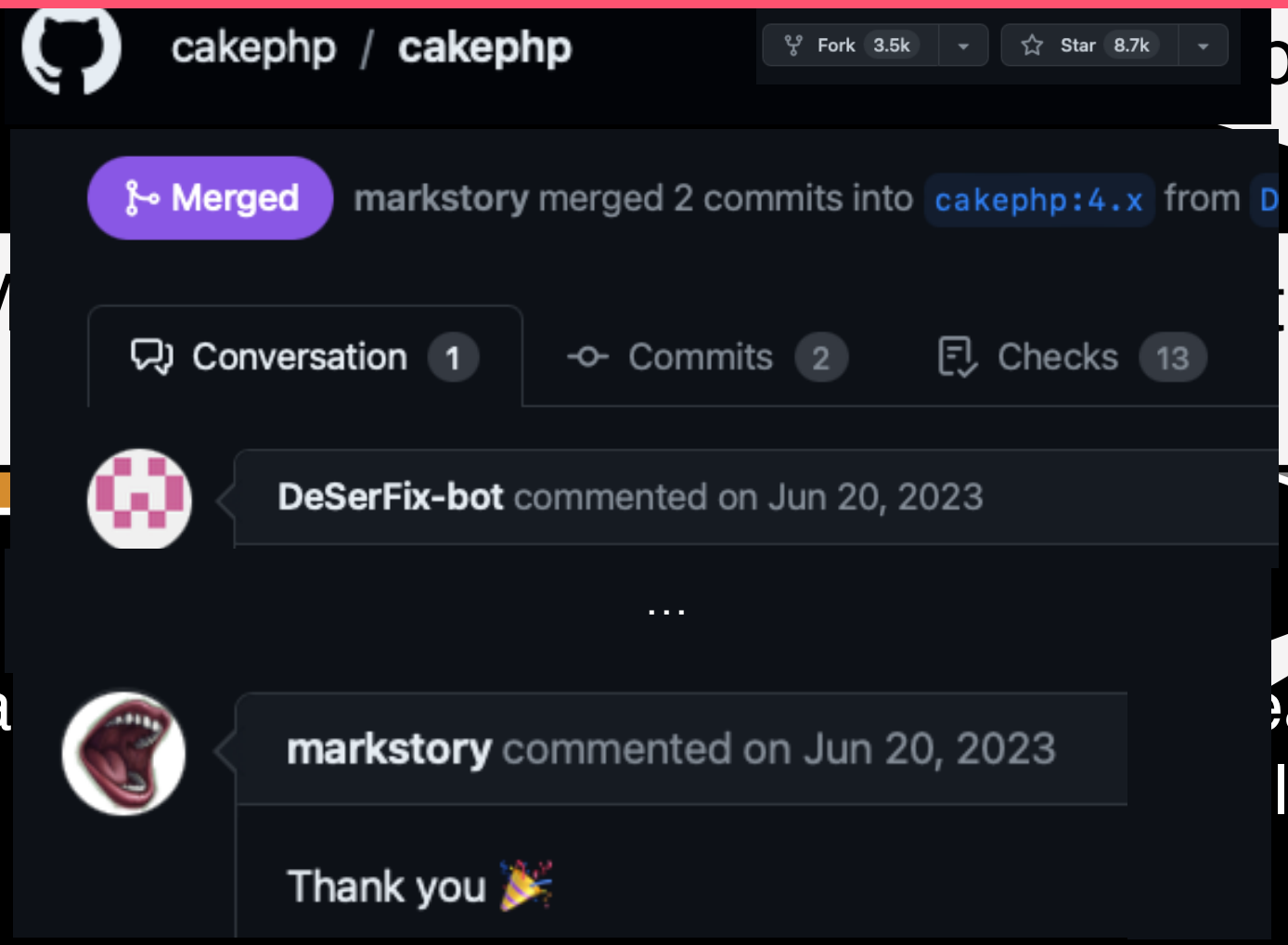
Quack: Hindering Deserialization Attacks



Built Quack-php and eva eal applications against SC

Image Source: Stable Diffusion @ stablediffusionweb.com

Quack:



Application (+ Dependenci Code

M

ploits

projects

Protected
lization
alls

Attack Surface

Built Qua

eal applications

l

Image Source: Stable Diffusion @ stablediffusionweb.com

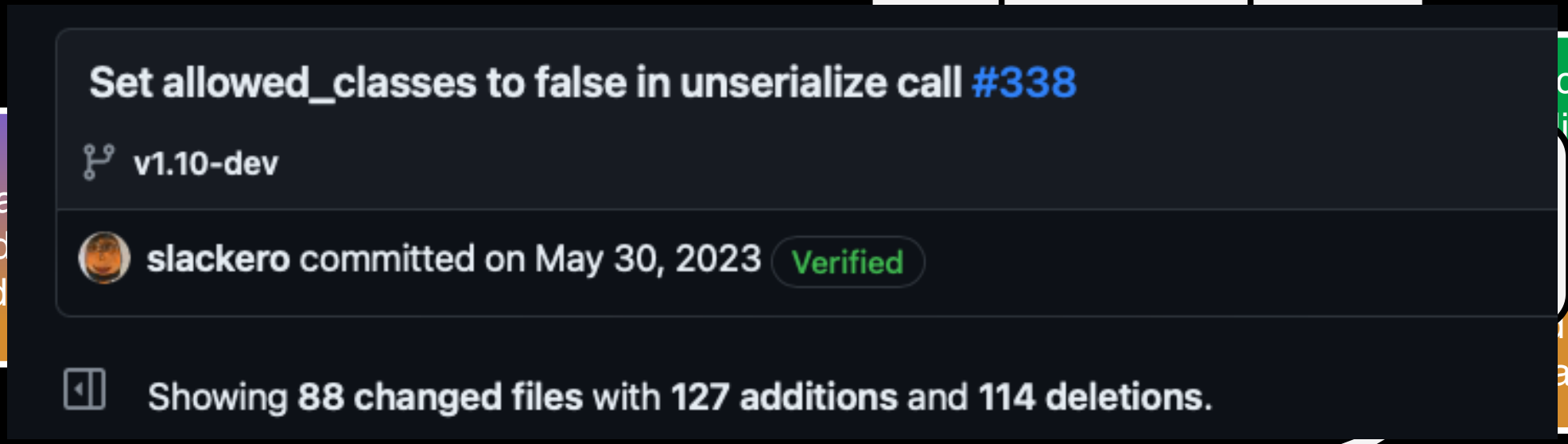
Quack: Hindering Deserialization Attacks

Stops all exploits

Applica
(+ Depend
Cod

cted
ization
alls

Attack
ace



Built Quack-php and eva
against SC

97% blocked exploit-building classes

eal applications
l

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’]]);
```

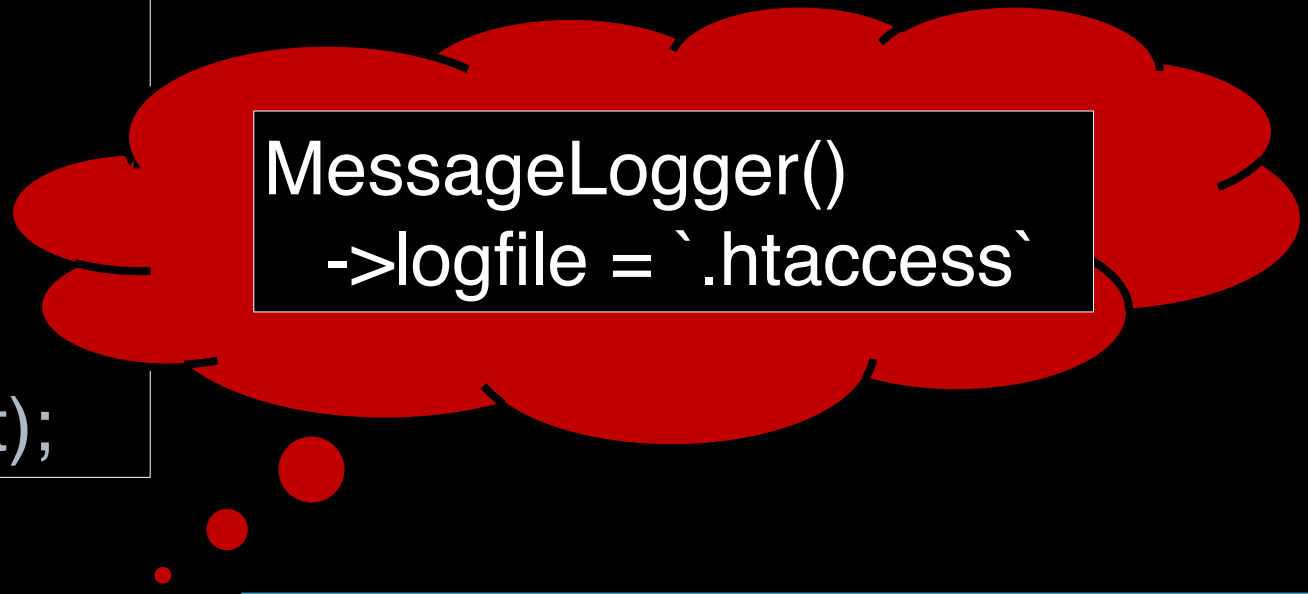
Why Not Limit Deserialized Classes?

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

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

Why Not Limit Deserialized Classes?

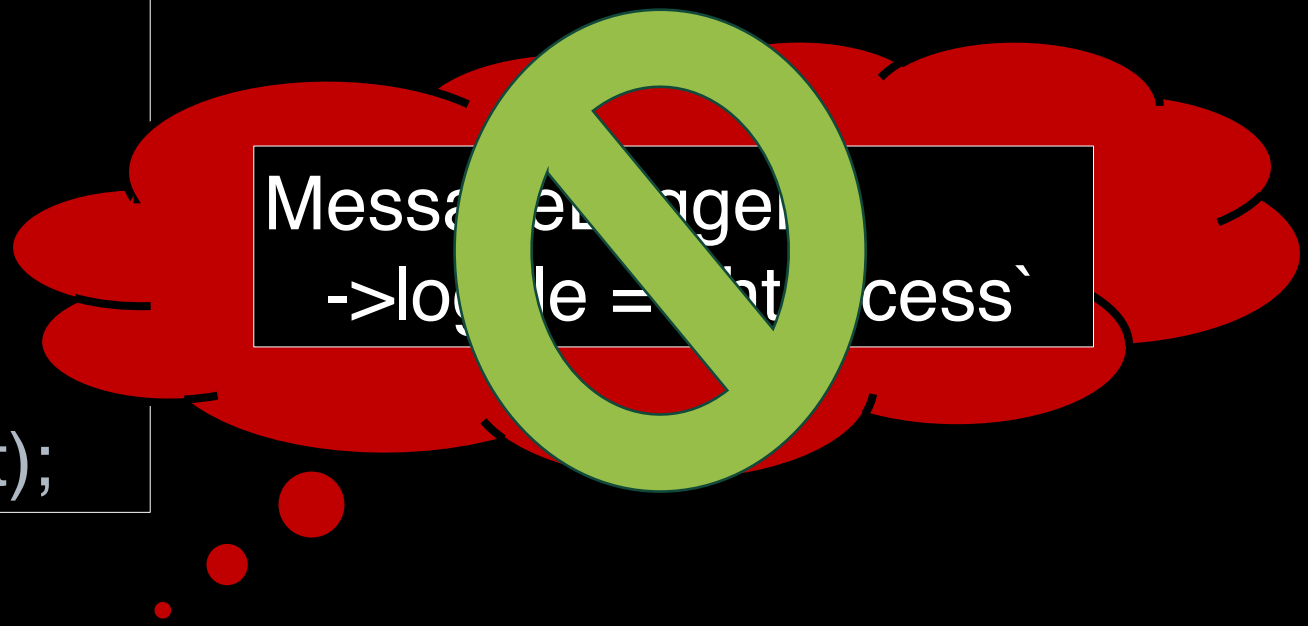
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```




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

Why Not Limit Deserialized Classes?

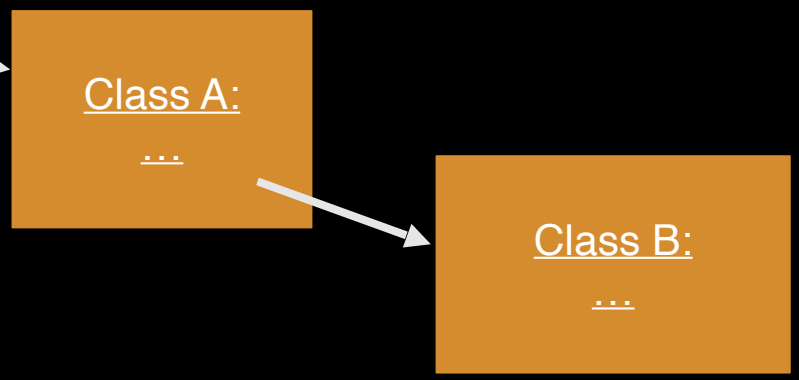
```
class Event {
  private wrapped_obj;
  /* snip */
}
/* snip */
to_send = serialize(event);
```



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

Why Not Limit Deserialized Classes?

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

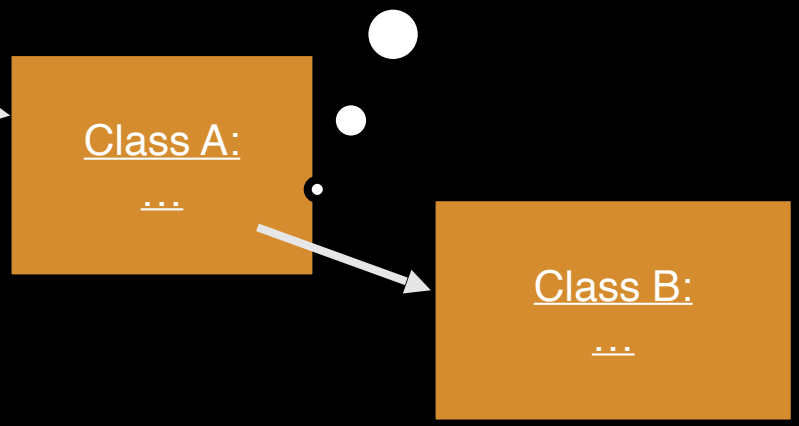



```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized Classes?

What happens if these are updated?

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

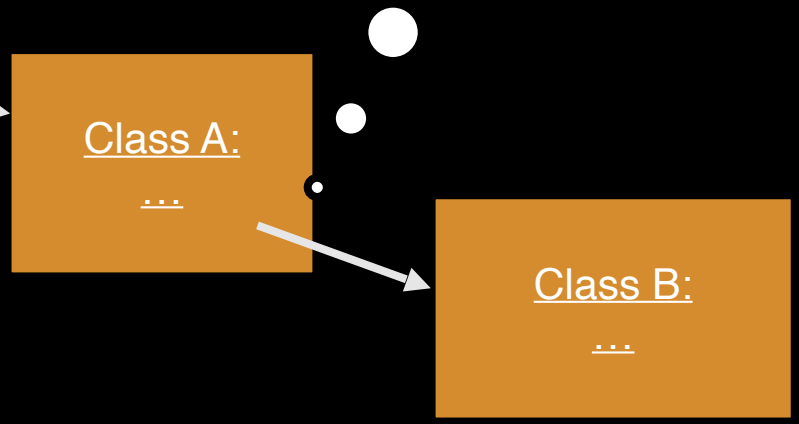



```
recv_event = deserialize(ser_event, [‘allowed_classes’=>[‘Event’]]);
```

Why Not Limit Deserialized Classes?

What happens if these are updated?

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

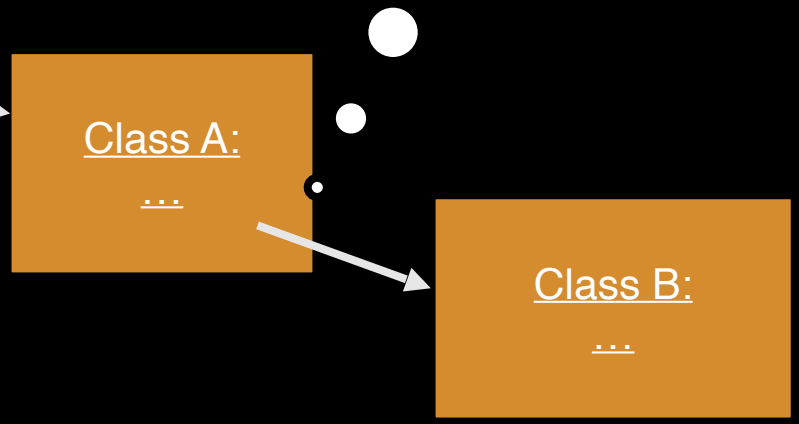



```
recv_event = deserialize(ser_event, ['allowed_classes'=>['Event']]);
```

Why Not Limit Deserialized Classes?

What happens if these are updated?

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




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

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

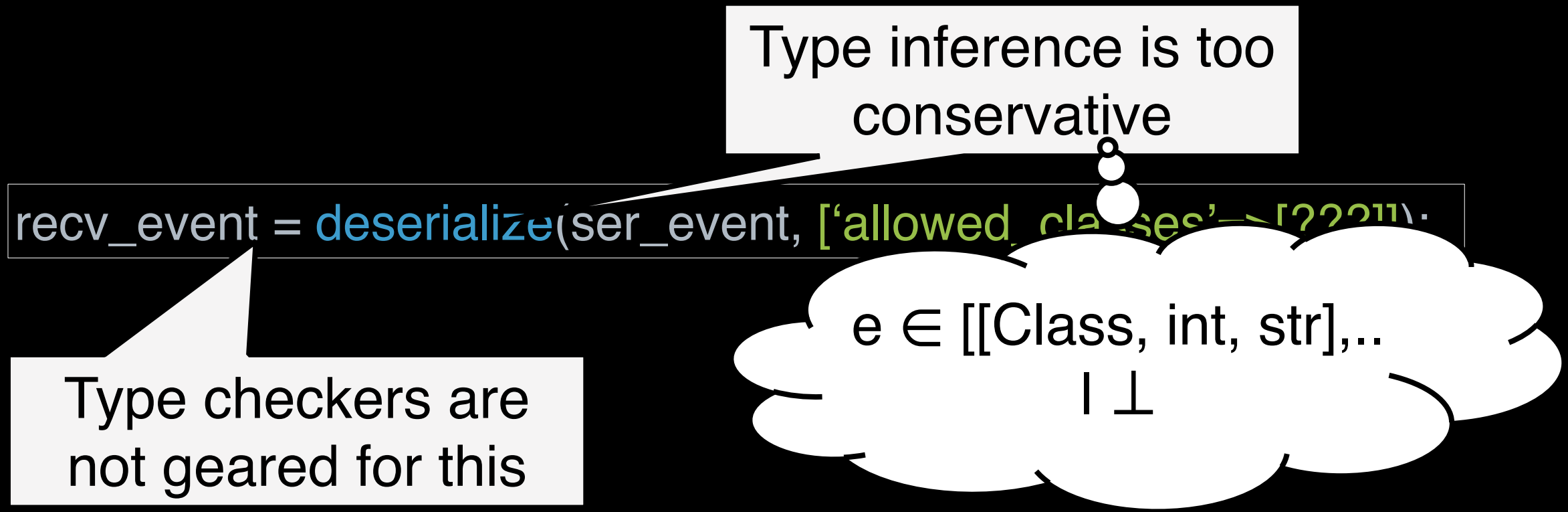
Type inference is too conservative

```
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



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);
```

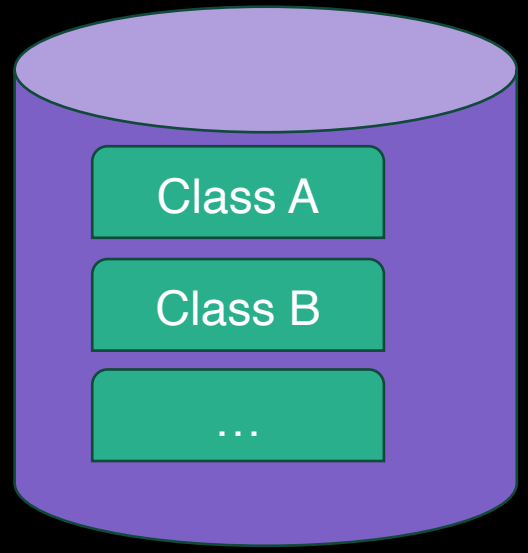
Open Web Analytics v1.5.6 containing the deserialization vulnerability CVE-2014-2294

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

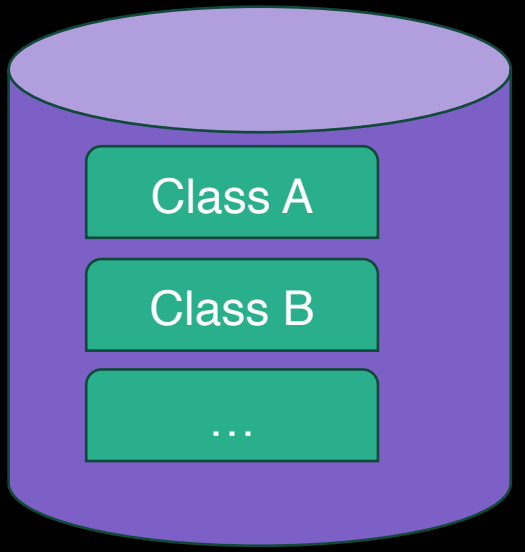
Analyzing Our Motivating Example

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

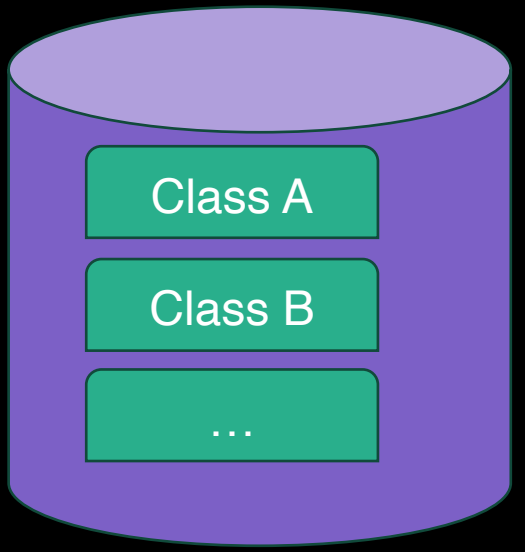
Analyzing Our Motivating Example

Root Object

queue.php

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

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



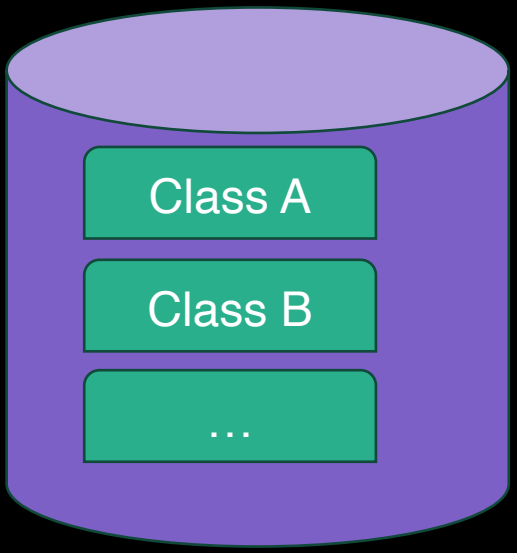
Follow

Open Web Ana ontaining 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());
  }
}
```

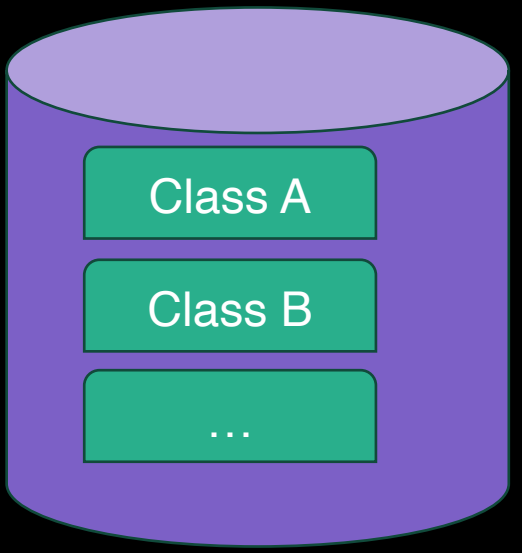


Analyzing Our Motivating Example

eventDispatch.php

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

No Type Hint



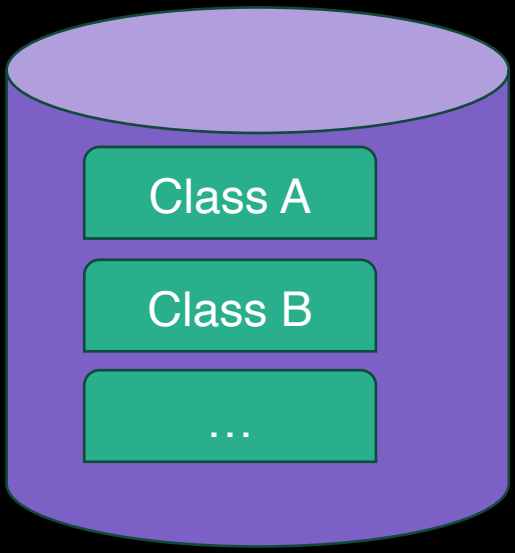
Analyzing Our Motivating Example

eventDispatch.php

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

No Type Hint

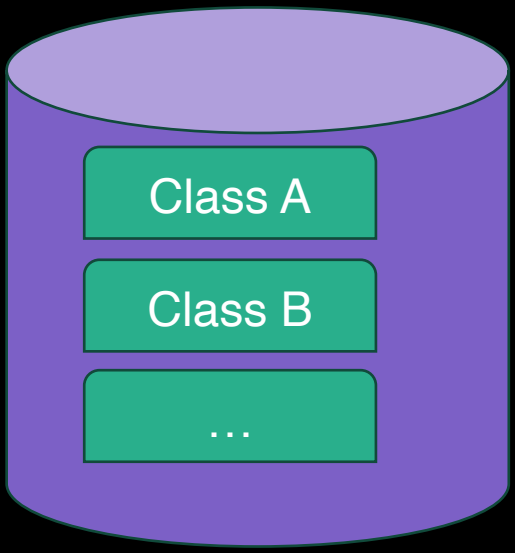
Has method with this name



Analyzing Our Motivating Example

eventDispatch.php

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



No Type Hint

Has method with this name

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

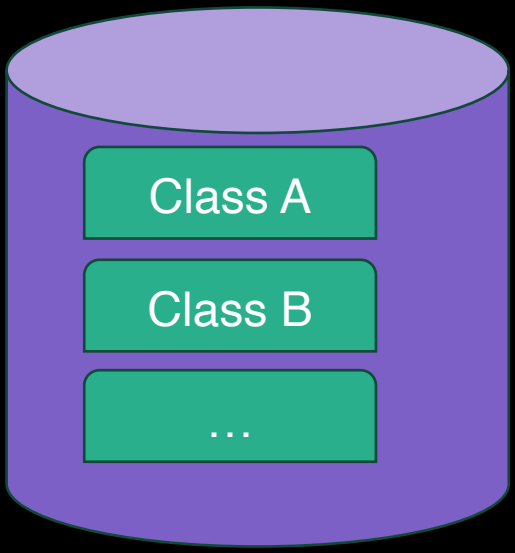
Analyzing Our Motivating Example

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_core::API::debug("Notifying listeners of"
                                    + event->getEventType());
  }
}
```

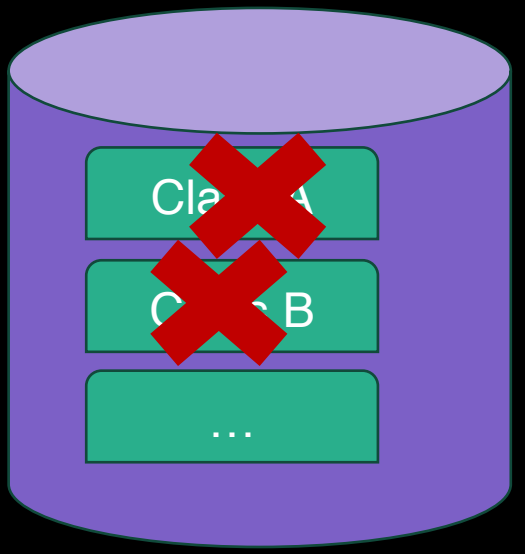


No Type Hint

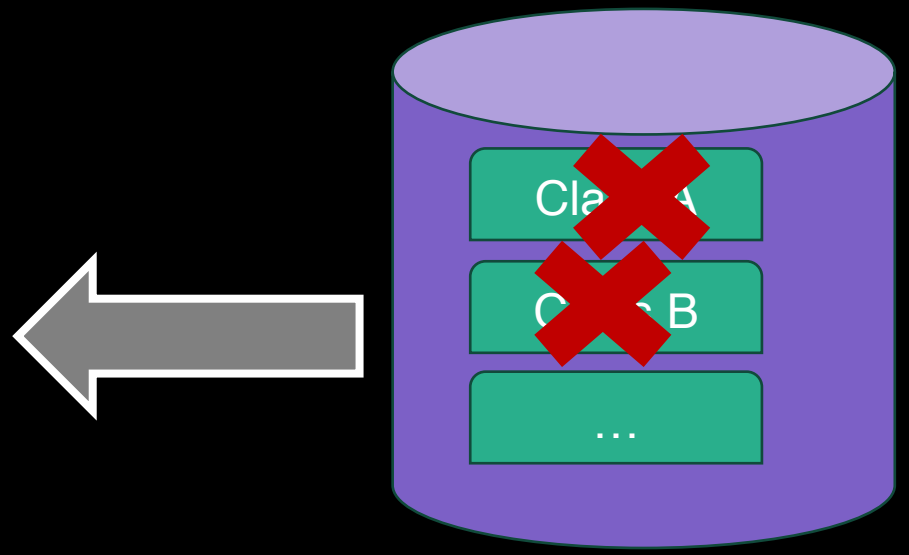
Has method with this name

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

Analyzing Our Motivating Example



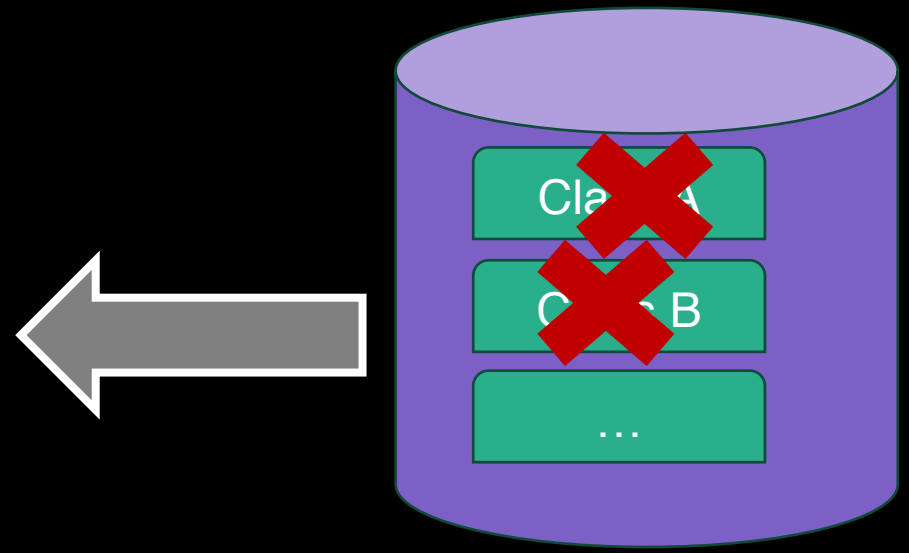
Analyzing Our Motivating Example



Analyzing Our Motivating Example

event.php

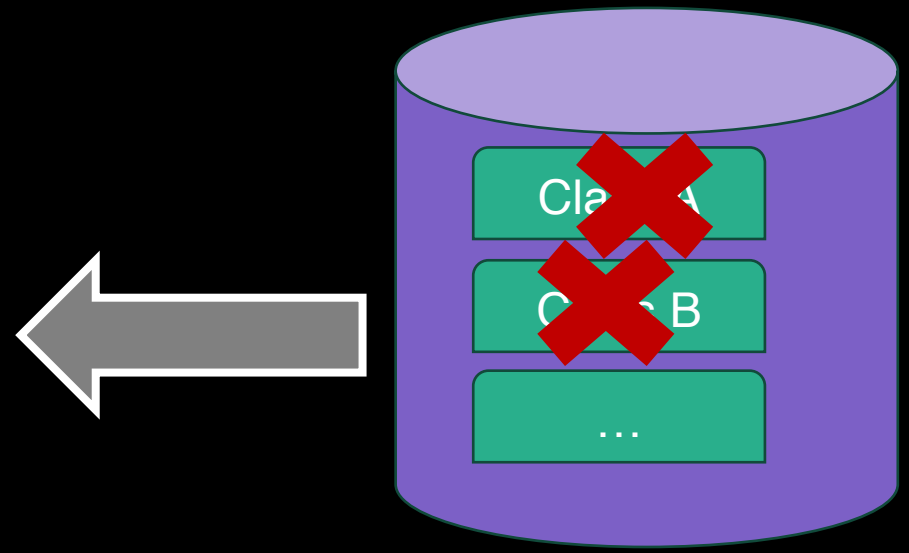
```
class owa_event {
  function getEventType () { /* snip */}
}
```



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 =    >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

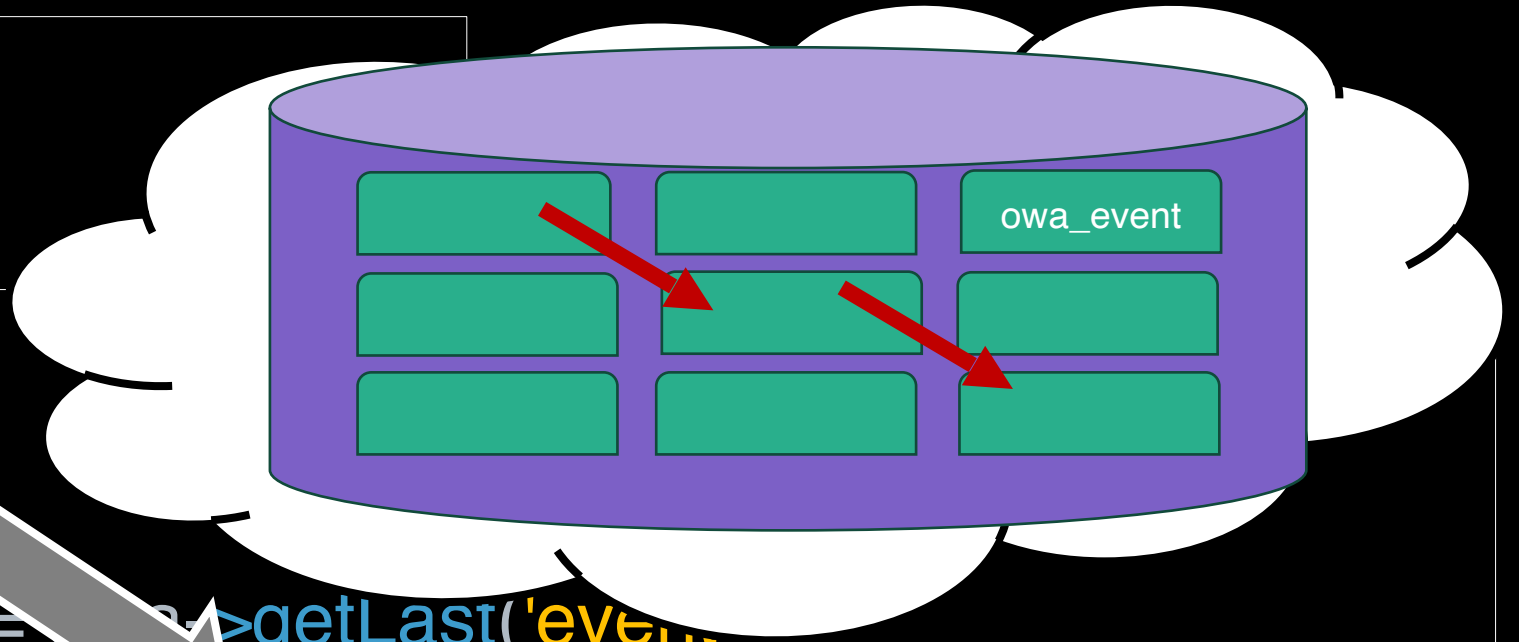
```
/* snip */
raw_event =    >getLast('event');
event = deseria   e(raw_event,
++                  ['allowed_classes' => ['owa_event']]);
owa::getEventDi      ier()->notify(event);
```

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      ier()->notify(event);
```

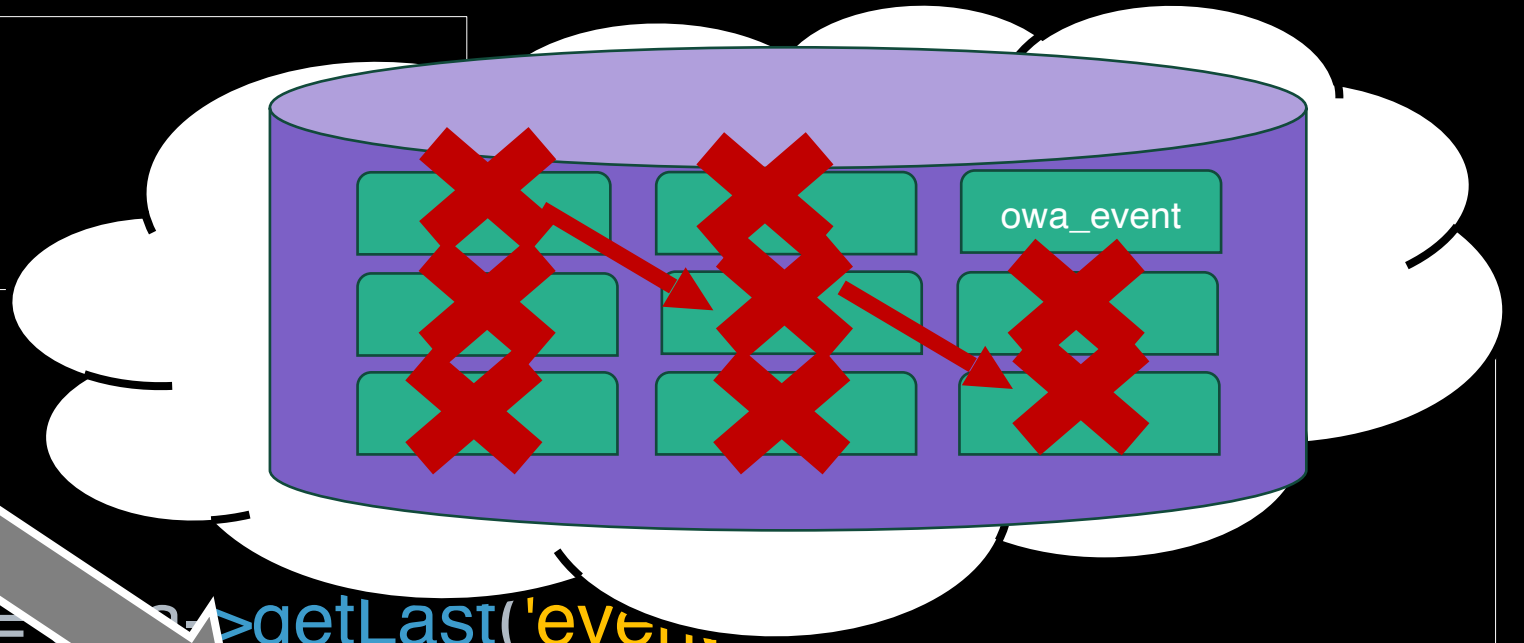
Stops 14 different exploit chains

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...ier()->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	5	79 (55)	100%
VULN202x	10	194(114)	95(8)%

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	5	79 (55)	100%
VULN202x	10	194(114)	95(8)%

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	5	79 (55)	100%
VULN202x	10	194(114)	95(8)%

Quack Automatically Stops Or Hinders Attacks

Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	5	79 (55)	100%
VULN202x	10	194(114)	95(8)%

Protecting Against Recent CVEs

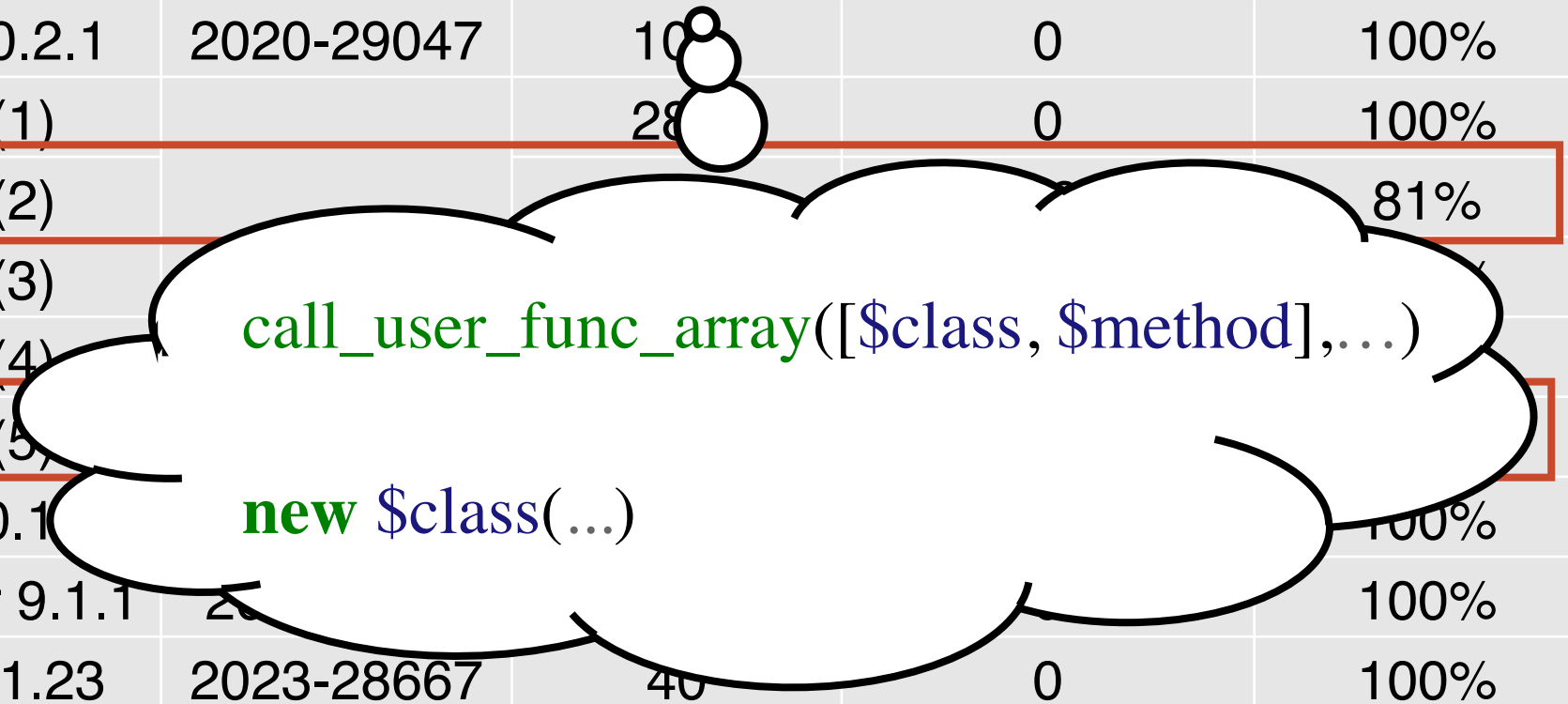
Application	CVE	Exploit-Building Classes		
		#Blocked	#Remaining	%Blocked
ForkCMS 5.8.3	2020-24036	221	23	91%
WP-hotel-booking 10.2.1	2020-29047	103	0	100%
OpenCATS-0.9.5 (1)	2021-25294	288	0	100%
OpenCATS-0.9.5 (2)		232	56	81%
OpenCATS-0.9.5 (3)		288	0	100%
OpenCATS-0.9.5 (4)		288	0	100%
OpenCATS-0.9.5 (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	0	100%
OpenCATS-0.9.5 (1)		28	0	100%
OpenCATS-0.9.5 (2)				81%
OpenCATS-0.9.5 (3)				
OpenCATS-0.9.5 (4)				
OpenCATS-0.9.5 (5)				
WP-AIOSEO 4.1.0.1				100%
WP-booking-calendar 9.1.1	2			100%
WP-lead-generated 1.23	2023-28667	40	0	100%



Performance and Safety

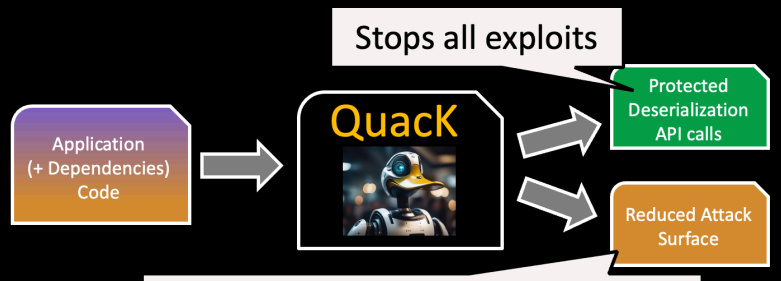
Dataset	#CVEs	Exploit-Building Classes [AVG(STD)]	
		Initial Count	% Blocked
FUGIO	5	79 (55)	100%
VULN202x	10	212(106)	95(8)%

- 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

Quack: Securing Applications Against Deserialization Attacks



Stopping Deserialization Exploits

event.php

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

queue.php

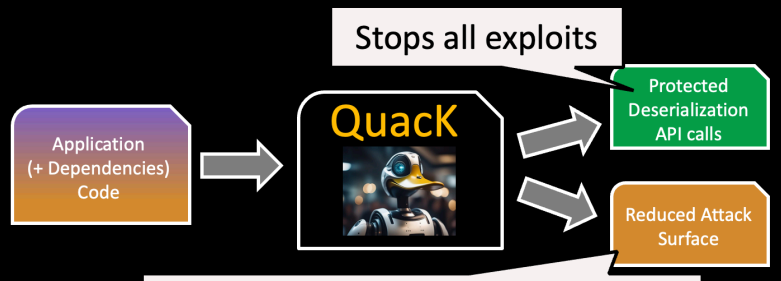
```
/* sn
raw_even  wa->getLast('event');
event = deseri  (base64_decode(raw_event),
++                ['allowed_classes' => ['owa_event']]);
owa::getEv  ber()->notify(event);
```

Stops 14 different exploit chains

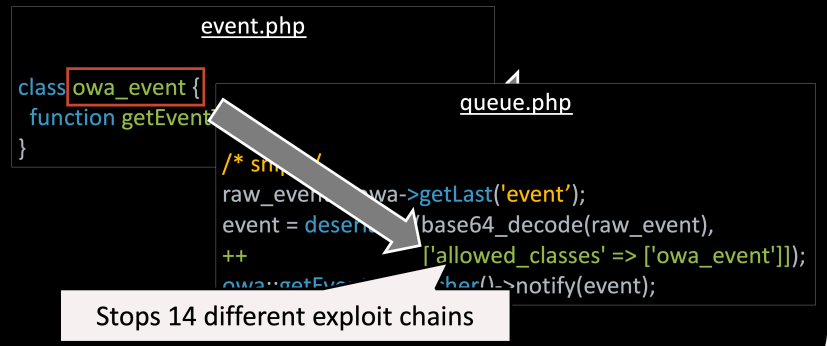
https://github.com/columbia/quack

Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

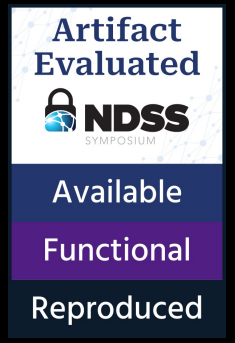
Quack: Securing Applications Against Deserialization Attacks



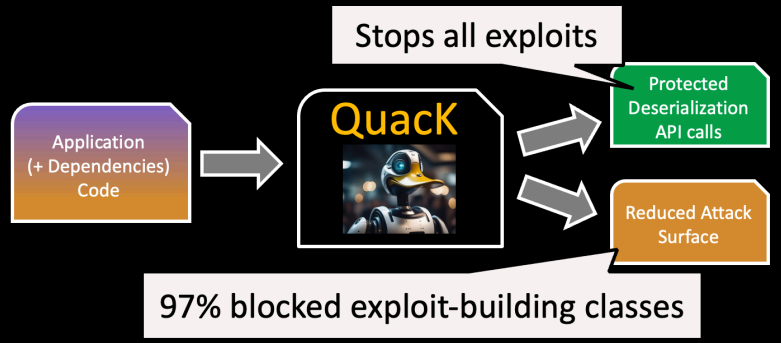
Stopping Deserialization Exploits



https://github.com/columbia/quack



Quack: Securing Applications Against Deserialization Attacks



Stopping Deserialization Exploits