Secure by Design

A novel industry practice

CASTOR Software Days 2019

October 15, 2019

Daniel Deogun & Dan Bergh Johnsson

omega point.

Disclaimer

This presentation is purely anecdotical.
But, we would welcome rigorous studies.

About us

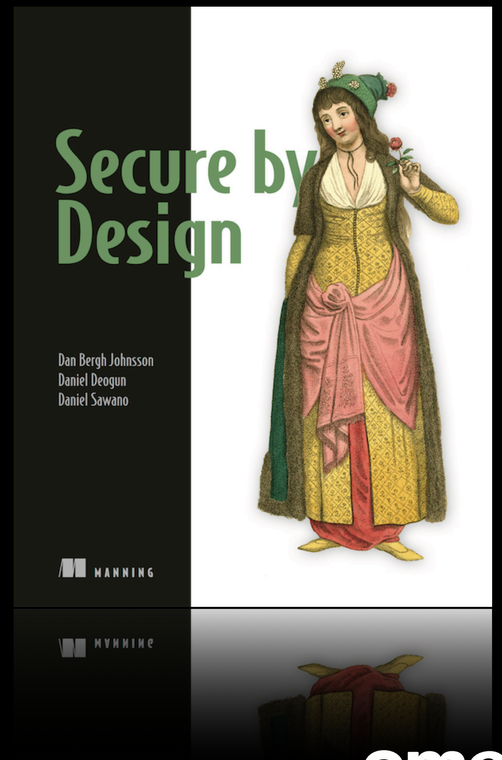
Daniel Deogun
Coder and Quality Defender

Omegapoint

Dan Bergh Johnsson
Secure Domain Philosopher

Take-aways

- Security is important but in practice often neglected
- A lot of security vulnerabilities are due to bugs
- Bugs can be prevented by software design
- *Secure by Design* collects designs that prevents bugs that manifest as security vulnerabilities



@DanielDeogun @danbjson #SecureByDesign

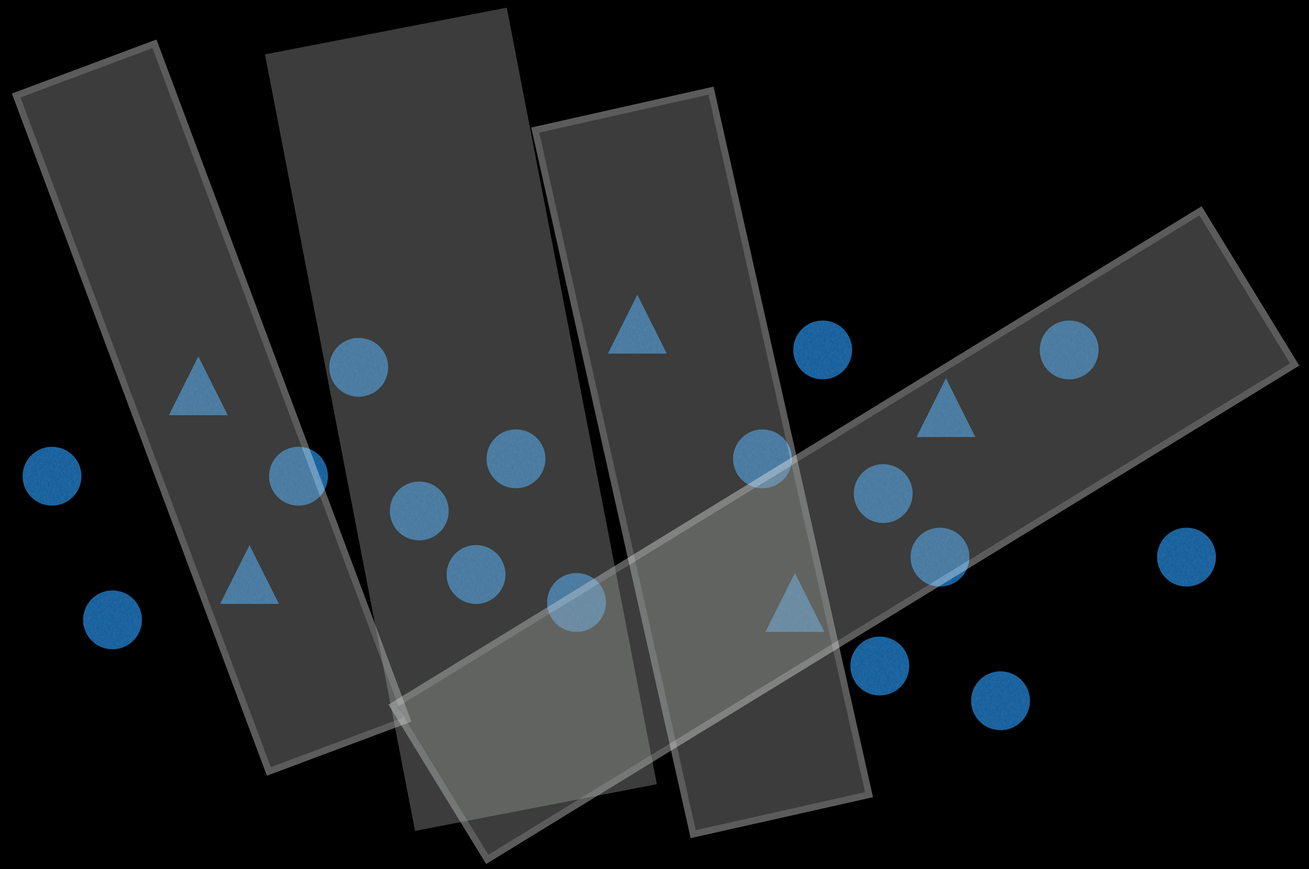
omega point.

Some security vulnerabilities

- SQL Injection is often a confidentiality problem
- Business integrity problems
- Patch management

Observation: It is hard to think about security all the time - especially when you have work to do.

Secure by Design - as method



Examples of interesting designs

- *Domain Primitives*
- Consistent upon Creation
- Immutability of Objects
- Entity Snapshot
- Testing of extremes
- Taint analysis
- Faults as normal results
- Immutable builds
- Seamless deploy
- Stateless components
- Externalization of configuration
- *3 R's of Enterprise Security*

Domain Primitives

“A value object so precise in its definition that it, by its mere existence, manifests its validity is called a Domain Primitive.”

- Secure by Design

Can only exist if its value is ***valid***

Building block that’s native to ***your domain***

Valid in the ***current context***

Immutable and resemble a value object in DDD

Quantity as a Domain Primitive

```
public final class Quantity {
    private final int value;

    public Quantity(final int value) {
        inclusiveBetween(1, 99, value);

        this.value = value;
    }

    //Domain specific quantity operations...
}
```

Untangle Inside
- Cluttered Entity

```
class Order {
    private ArrayList<Object> items;
    private boolean paid;


    public void addItem(String isbn, int qty) {
        if(this.paid == false) {
            notNull(isbn);
            inclusiveBetween(10, 10, isbn.length());
            isTrue(isbn.matches("[0-9X]*"));
            isTrue(isbn.matches("[0-9]{9}[0-9X]"));

            Book book = bookCatalogue.findByISBN(isbn);

            if (inventory.availableBooks(isbn) >= qty) {
                items.add(new OrderLine(book, qty));
            }
        }
    }
    //Other logic...
}
```

https://flic.kr/p/wdBcT
https://creativecommons.org/licenses/by/2.0/

De-Cluttered Entity

```
class Order {
    private ArrayList<Object> items;
    private boolean paid;

    public void addItem(ISBN isbn, Quantity qty) {
        notNull(isbn);
        notNull(qty);

        if(this.paid == false) {
            Book book = bookCatalogue.findByISBN(isbn);

            if (inventory.availableBooks(isbn).greaterOrEqualTo(qty)) {
                items.add(new OrderLine(book, qty));
            }
        }
    }
    //Other logic...
}
```

https://flic.kr/p/wdBcT
https://creativecommons.org/licenses/by/2.0/

The R's of Enterprise Security

Rotate

Repave

Repair

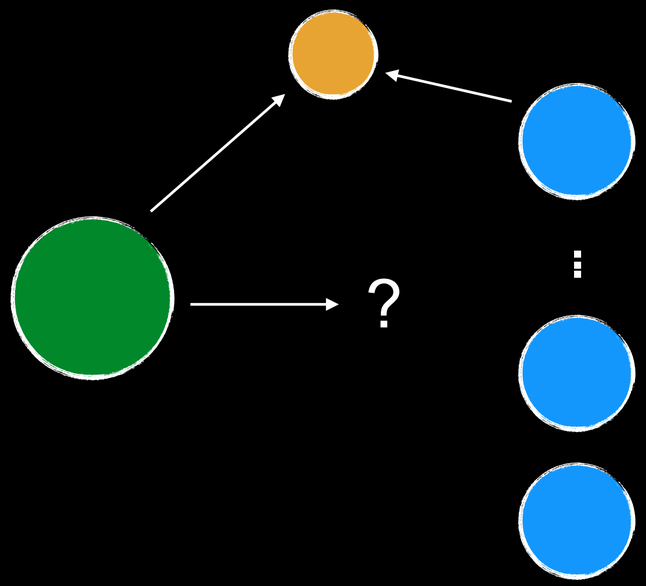
Ref: Justin Smith, Pivotal @justinjsmith

Advanced Persistent Threat (APT)

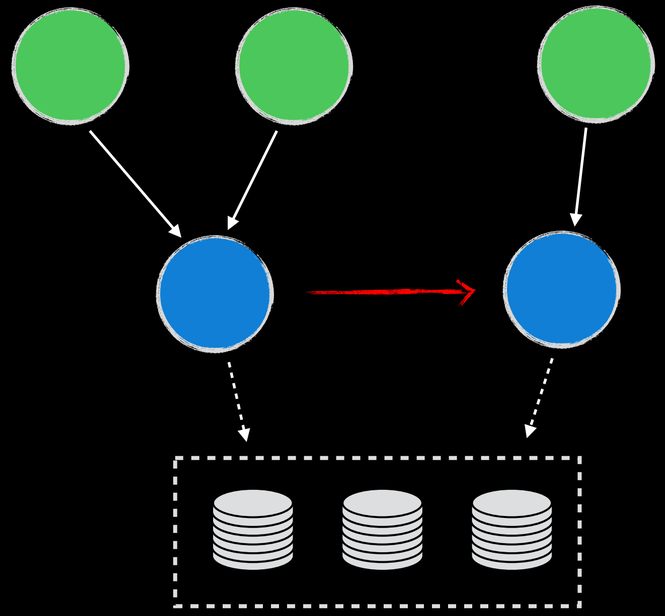
- APT is a type of attack that often result in significant data loss or damage.
- Performed over a long period of time & involves advanced techniques
- Several vulnerabilities in combination are often used to exploit the system
- Exploiting the fact that things seldom change
  - servers
  - credentials
  - ip-addresses
  - non-patched bugs tend to open up for attacks, e.g.
    - Baltimore City Ransomware
    - NotPetya attack on Ukraine
    - Heartbleed in OpenSSL

Richard Patterson https://flic.kr/p/24GcRZQ

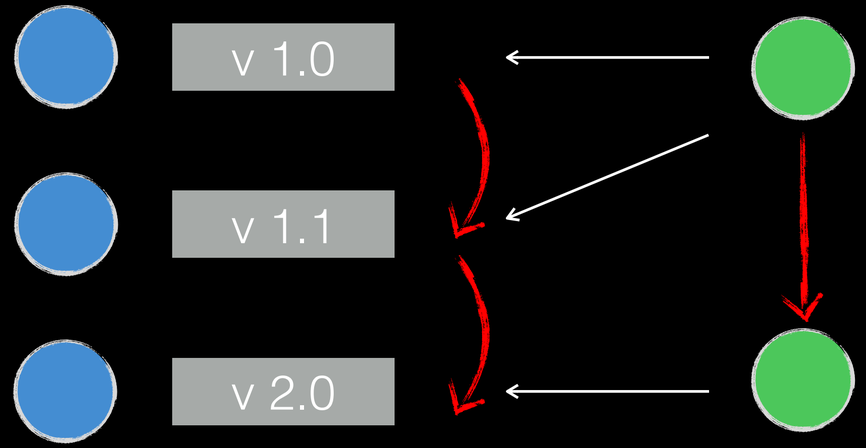
Designs Inspired from the Cloud



Service Discovery



Stateless Services



Expand / Contract APIs

The R's of Enterprise Security

Rotate

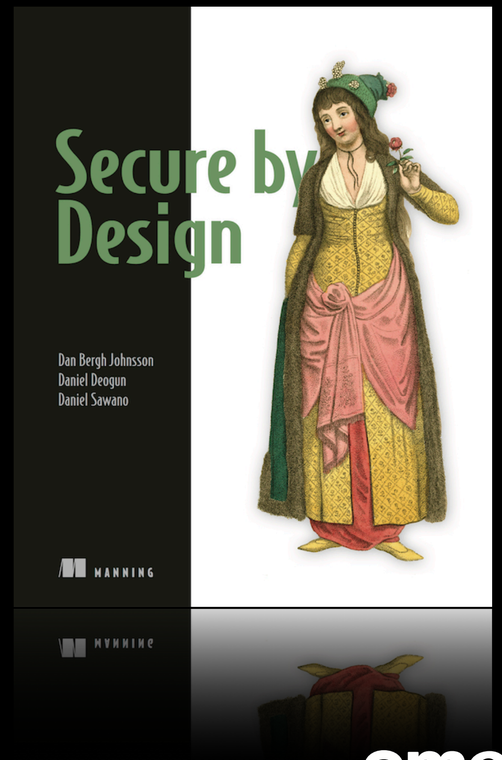
Repave

Repair

Ref: Justin Smith, Pivotal @justinjsmith

Take-aways

- Security is important but in practice often neglected
- A lot of security vulnerabilities are due to bugs
- Bugs can be prevented by software design
- *Secure by Design* collects designs that prevents bugs that manifest as security vulnerabilities



@DanielDeogun @danbjson #SecureByDesign

omega point.

Thanks

 @DanielDeogun @danbjson #SecureByDesign

omega point.