Complexity vs Simplicity

in web application security

Sergey Ozernikov

sergey@attack.co.nz

Bio: Sergey Ozernikov

- A Russian Hacker
- Leads Security Testing at ATTACK Limited
- 12+ years in Information Security overall
  - 6 years pentesting/offensive security
  - 3 years architecture/defensive security
- Personal interests: Life
- Twitter handle: I don’t remember

Motivation

“Security, Moore's Law, And The Anomaly Of Cheap Complexity” by T. Dullien

https://www.youtube.com/watch?v=q98foLaAfX8

Defence in depth

Use secure defaults

Fail securely

Avoid security by obscurity

Attack surface minimisation

Do not trust by default

Standard security controls

Least privilege

Keep security simple

Avoid security by obscurity

Defer

defaults

Fail secu

curity

Attack

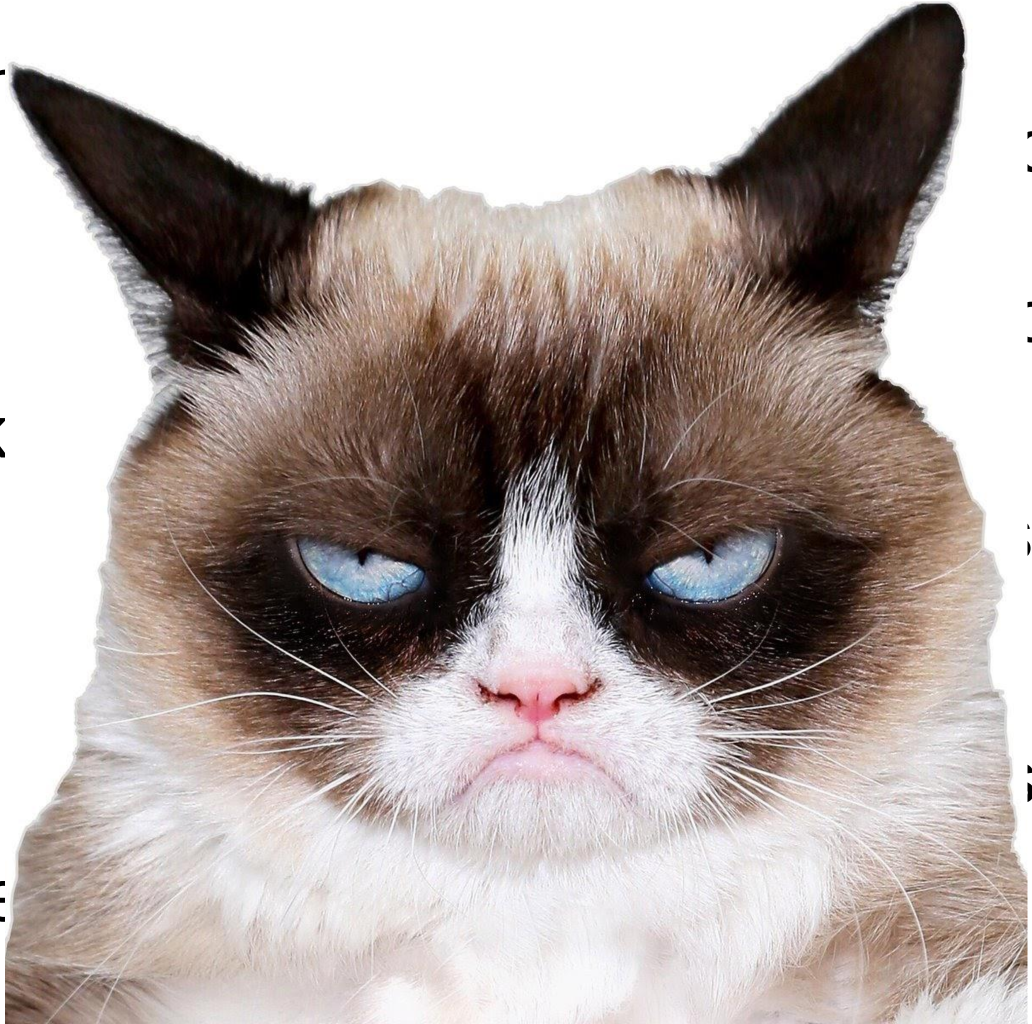
t by default

Standard

Least privil

simple

Avoid se



Defence in depth

Use secure defaults

Fail securely

Avoid security by obscurity

Attack surface minimisation

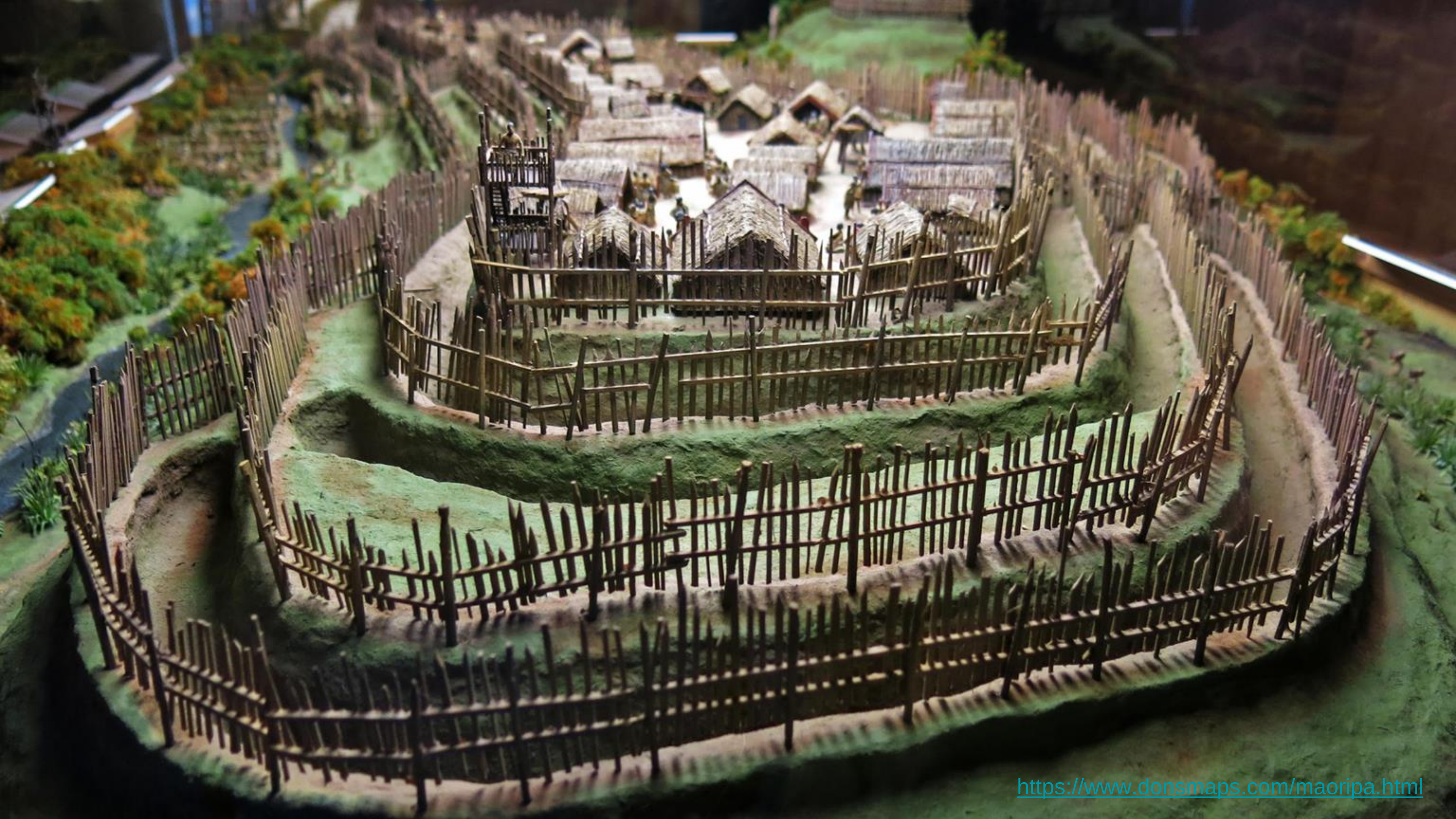
Do not trust by default

Standard security controls

Least privilege

Keep security simple

Avoid security by obscurity



https://www.donsmaps.com/maoripa.html

https://unsplash.com/wallpapers/travel/chicago

Application Security Architecture

Application Security Architecture

Monitoring and Response

Advisory / Assurance

Hardening / Patching

Firewall / WAF

Application

Development Frameworks

- Use frameworks!
- Use security controls provided by a framework
- Not all frameworks are equally good at security
- Try not to build strong dependencies (e.g. Angular 1.x vs Angular 2.0)
- If you’re free to chose - do your homework (e.g. cvedetails.com)
- Patch rigorously

Content Management Systems

- Not all CMS’es are equally good at security (again, do your homework)
- Minimise attack surface – do not install/use funcitonality that you don’t use
- Restrict access to admin interfaces
- Prevent easy external fingerprinting (remove version banners, check with CMS scanners like wpscan, droopscan, etc.)
- Patch rigorously

Cryptographic Mechanisms

- Do not build your own crypto!
- Understand security properties required
- Choose right cryptographic primitives to satisfy desired properties
- Choose right settings/review defaults for your implementation
- https://latacora.micro.blog/2018/04/03/cryptographic-right-answers.html
- Seek further consultation if in doubt

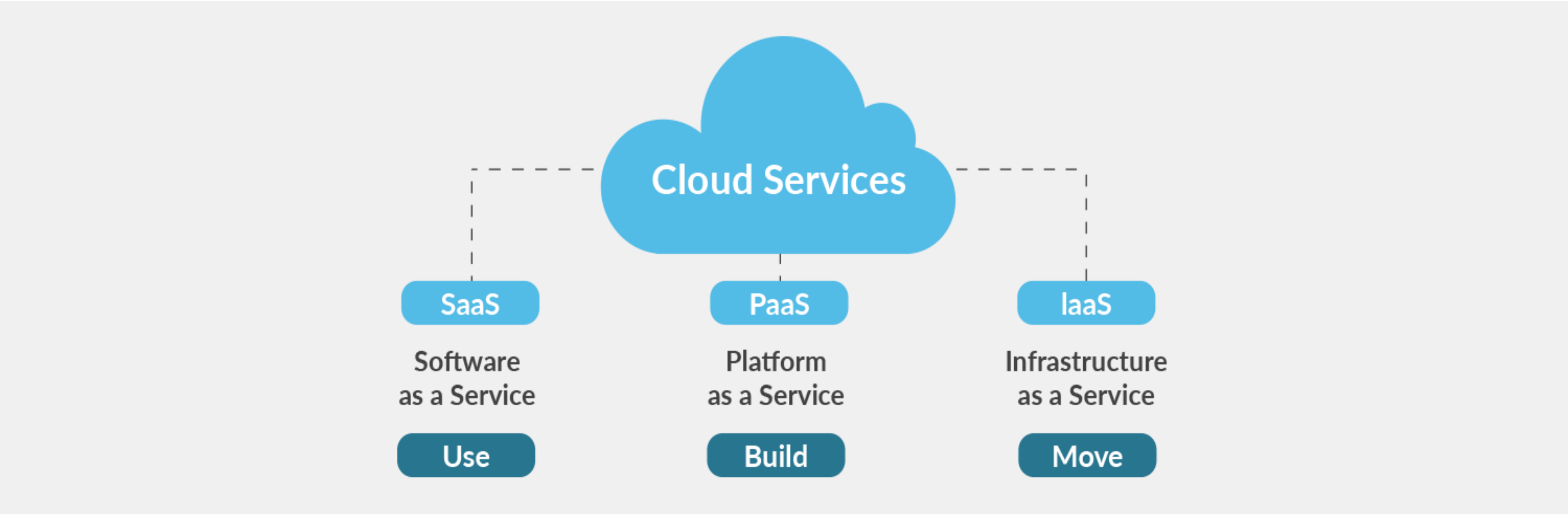
DevOps tooling

- Use automation, use tooling!
- CI/CD tooling – ability to automate is key to security delivery
- Consider its security to be part of the security of the application
- Ensure your tooling setup is:
    - Understood
    - Secured
    - Maintained (patch rigorously)

Security Controls

- Security controls may add to the attack surface themselves

Cloud Adoption Models



https://www.plesk.com/blog/various/iaas-vs-paas-vs-saas-various-cloud-service-models-compared/

Cloud Adoption Models

- Beware of IaaS - pure “Lift and Shift” approach can cause problems
- Combination of traditional and cloud security controls
- Responsibility matrix understanding

Example: CapitalOne breach

● What happened:	a data breach
● When:	from March 2019 to July 2019
● Who was affected:	~106 000 000 customers
● What was leaked:	personal data including names, addresses, phone numbers, self-reported income, SSN
● Who identified it:	a third-party

Example: CapitalOne breach

- Attacker was under arrest after a short investigation
- Indictment with some technical details published
- Public scrutiny followed. As per Brian Krebs, it was a SSRF vulnerability within the WAF (ModSecurity) which escalated to unauthorised access to an S3 bucket due to insecure AWS permissions
- It is unclear if WAF was itself exploited or bypassed / unclear if the WAF was on the same box or separate
- Shortly after the breach AWS announced a change in Metadata endpoint access to prevent SSRF

What should we do about it?

Defence

1. Application security/WAF configuration (better have them on different servers). Or even better using a SaaS WAF.
2. Auditing of AWS IAM roles
    a. Principle of least privilege.
    b. Don’t forget network side of things! E.g. assuming a role to be permitted only from inside a certain VPC/IP addresses (https://aws.amazon.com/premiumsupport/knowledge-center/iam-restrict-calls-ip-addresses/). This would most likely mean that a SSRF won’t help here in passing through this restriction (additional HTTP headers required to use the role).
3. Monitoring
    a. Monitoring of a breach vs. monitoring of a misconfiguration?
    b. Robust initial audit/baseline + AWS Config, Cloudwatch alerts on IAM roles change that are recorded in CloudTrail could have prevented the IAM misconfiguration in the first place.
4. Preventative controls
    a. https://medium.com/netflix-techblog/netflix-information-security-preventing-credential-compromise-in-aws-41b112c15179

Some general takeaways

- Know what security controls reside within which component of the solution
- Assess security posture for a technology before choosing
- Do not deploy/configure/install stuff that is not required
- Disable/restrict access to unused functionality which is already in place
- Monitor not only for attacks but check if crucial security controls are operational

Some general takeaways

- Know what security controls reside within which component of the solution
- Assess security posture for a technology before choosing
- Do not deploy/configure/install stuff that is not required
- Disable/restrict access to unused functionality which is already in place
- Monitor not only for attacks but check if crucial security controls are operational
- Patch and harden rigorously

Who won?

Complexity vs Simplicity

Who won?

Complexity vs ~~Simplicity~~

Who won?

Complexity* vs ~~Simplicity~~

* Controlled

ATTACK™

Thank you!

We’re hiring!

info@attack.co.nz