Unit 11: Future trends in Secure Software Development

Unit 12: The Great Tanenbaum-Torvalds Debate Revisited

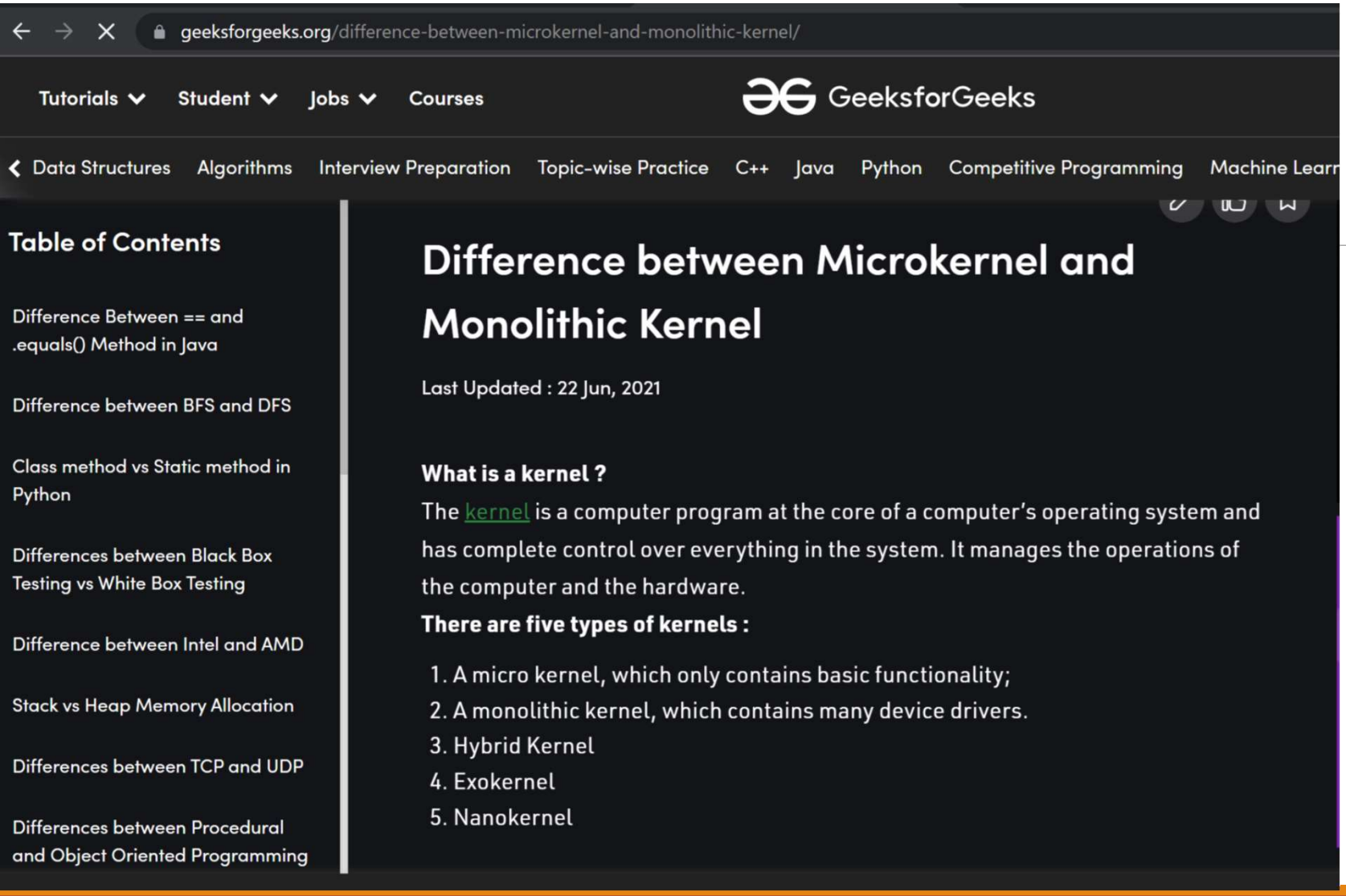
Discussion of one of the great arguments in both system and software development: what is the best approach – monolithic or microkernels and modular?

READING LIST

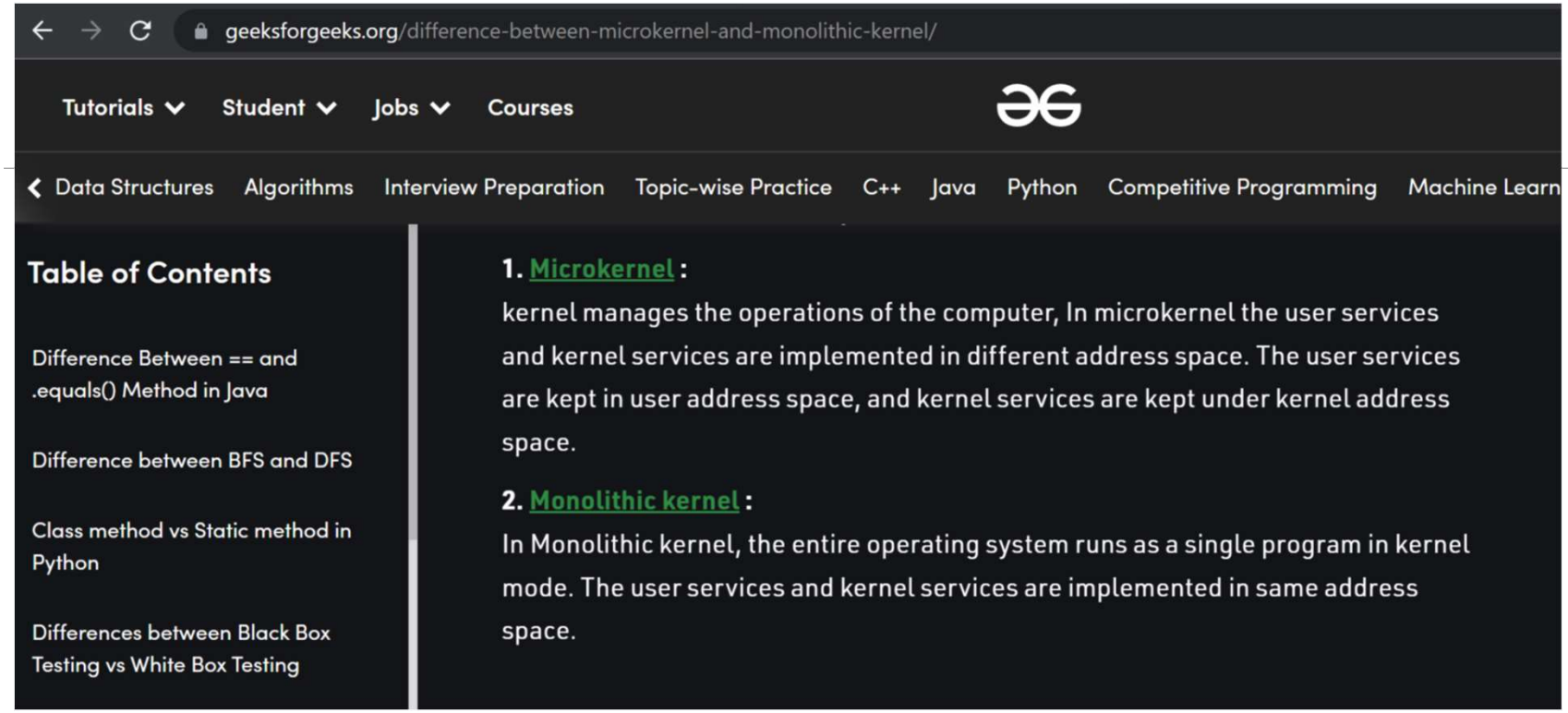
DiBona, C. & Ockman, S. (1999) *Open Sources*. 1st ed. Sebastapol: O'Reilly Media, Inc. [https://www.oreilly.com/openbook/opensources/book/]

Fritzsch J., Bogner J., Zimmermann A. & Wagner S. (2019) From Monolith to Microservices: A Classification of Refactoring Approaches. In: Bruel JM., Mazzara M., Meyer B. (eds) Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment. DEVOPS 2018. Lecture Notes in Computer Science, vol 11350. Springer.

Roman, R. Lopez, J. & Mambo, M. (2016) Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges. *Future Generation Computer Systems* 78(1): 680 - 698.



https://www.geeksforgeeks.org/difference-between-microkernel-and-monolithic-kernel/


geeksforgeeks.org/difference-between-microkernel-and-monolithic-kernel/
Tutorials
Student
Jobs
Courses
Data Structures
Algorithms
Interview Preparation
Topic-wise Practice
C++
Java
Python
Competitive Programming
Machine Learn
Table of Contents
Difference Between == and .equals() Method in Java
Difference between BFS and DFS
Class method vs Static method in Python
Differences between Black Box Testing vs White Box Testing
1. Microkernel :
kernel manages the operations of the computer, In microkernel the user services and kernel services are implemented in different address space. The user services are kept in user address space, and kernel services are kept under kernel address space.
2. Monolithic kernel :
In Monolithic kernel, the entire operating system runs as a single program in kernel mode. The user services and kernel services are implemented in same address space.

Differences between Microkernel and Monolithic Kernel :

Microkernel	Monolithic kernel
In microkernel user services and kernel, services are kept in separate address space.	In monolithic kernel, both user services and kernel services are kept in the same address space.
OS is complex to design.	OS is easy to design and implement.
Microkernel are smaller in size.	Monolithic kernel is larger than microkernel.
Easier to add new functionalities.	Difficult to add new functionalities.
To design a microkernel, more code is required.	Less code when compared to microkernel
Failure of one component does not effect the working of micro kernel.	Failure of one component in monolithic kernel leads to failure of entire system.
Execution speed is low.	Execution speed is high.
It is easy to extend Microkernel.	It is not easy to extend monolithic kernel.
Example : Mac OS X.	**Example :** Microsoft Windows 95.

Appendix A
The Tanenbaum-Torvalds Debate (Microkernel vs Monolithic system)
https://www.oreilly.com/openbook/opensources/book/appa.html

- Year 1992
- debate about Linux as an operating system
- Microkernel vs monolithic system

Appendix A

The Tanenbaum-Torvalds Debate

What follows in this appendix are what are known in the community as the Tanenbaum/Linus "Linux is obsolete" debates. Andrew Tanenbaum is a well-respected researcher who has made a very good living thinking about operating systems and OS design. In early 1992, noticing the way that the Linux discussion had taken over the discussion in comp.os.minix, he decided it was time to comment on Linux.

Although Andrew Tanenbaum has been derided for his heavy hand and misjudgements of the Linux kernel, such a reaction to Tanenbaum is unfair. When Linus himself heard that we were including this, he wanted to make sure that the world understood that he holds no animus towards Tanenbaum and in fact would not have sanctioned its inclusion if we had not been able to convince him that it would show the way the world was thinking about OS design at the time.

We felt the inclusion of this appendix would give a good perspective on how things were when Linus was under pressure because he abandoned the idea of microkernels in academia. The first third of Linus' essay discusses this further.

Electronic copies of this debate are available on the Web and are easily found through any search service. It's fun to read this and note who joined into the discussion; you see user-hacker Ken Thompson (one of the founders of Unix) and David Miller (who is a major Linux kernel hacker now), as well as many others.

To put this discussion into perspective, when it occurred in 1992, the 386 was the dominating chip and the 486 had not come out on the market. Microsoft was still a small company selling DOS and Word for DOS. Lotus 123 ruled the spreadsheet space and WordPerfect the word processing market. DBASE was the dominant database vendor and many companies that are household names today--Netscape, Yahoo, Excite--simply did not exist.

```
From: ast@cs.vu.nl (Andy Tanenbaum)
Newsgroups: comp.os.minix
Subject: LINUX is obsolete
Date: 29 Jan 92 12:12:50 GMT

I was in the U.S. for a couple of weeks, so I haven't commented much on
LINUX (not that I would have said much had I been around), but for what
it is worth, I have a couple of comments now.

As most of you know, for me MINIX is a hobby, something that I do in the
evening when I get bored writing books and there are no major wars,
revolutions, or senate hearings being televised live on CNN. My real
job is a professor and researcher in the area of operating systems.
```

Debate

Debate: Microservices and Microkernels

Read Appendix A: the Tanenbaum-Torvalds debate in DiBona & Ockman (1999) then read Fritzsch et al (2019).

The forum has a message that says: “Torvalds has been proven wrong and it only took nearly thirty years. Microservices and microkernels are the future. “

- On the forum post a message either agreeing or disagreeing with the above and give a justification (ideally with an academic reference) supporting your view.
- Outside the forum, discuss your positions in your team and come up with a team stance. This should be shared in Unit 12.

Learning Outcomes

- Critically analyse development problems and determine appropriate methodologies, tools and techniques (including program design and development) to solve them.
- Systematically develop and implement the skills required to be effective member of a development team in a virtual professional environment, adopting real-life perspectives on team roles and organisation.

Read Appendix A: the Tanenbaum-Torvalds debate in DiBona & Ockman (1999) then read Fritzsch et al (2019). The forum has a message that says: “Torvalds has been proven wrong and it only took nearly thirty years. Microservices and microkernels are the future.”

Tanenbaum’s arguments		**Torvalds’ arguments**	
Monolithic system	**Microkernel System**	**Monolithic system**	**Microkernel System**
older OS such as UNIX, MS-DOS…	Windows/NT (not yet released in 1992)	It was available	Not vastly available in 1992, even if nicer
Runs in ‘kernel mode’	Runs as separate processes, mostly outside the kernel		Issues with multitasking in the kernel
	Communicate by message passing	More portable (in the API) – much simpler design	
Performance was the main argument for using Monolithic system	Performance as good as Monolithic system (Mach 3.0 was compared in research papers)		
Costly (especially to make it run – for students)	Modern, free	free	

R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges", *Future Generation Computer Systems*, vol. 78, pp. 680-698, 2018.
http://doi.org/10.1016/j.future.2016.11.009
NICS Lab. Publications: https://www.nics.uma.es/publications

Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges

Rodrigo Roman[a], Javier Lopez[a], Masahiro Mambo[b]

[a] *Computer Science Department, University of Malaga, Ada Byron building, 29071 Malaga, Spain.*
[b] *Faculty of Electrical and Computer Engineering, Institute of Science and Engineering, Kanazawa University, Kakuma Kanazawa 920-1192, Japan.*

Abstract

https://www.nics.uma.es/pub/papers/RomanFog16.pdf

Security vulnerabilities and Operating Systems

Important issue

From Monolithic to Microservices: An experience report

Antonio Bucchiarone*, Nicola Dragoni†, Schahram Dustdar‡, Stephan T. Larsen§, Manuel Mazzara¶
* Fondazione Bruno Kessler, Trento, Italy
bucchiarone@fbk.eu
† Technical University of Denmark and Örebro University, Sweden
ndra@dtu.dk
‡ TU Wien
dustdar@dsg.tuwien.ac.at
§ Danske Bank, Denmark
stephantl@gmail.com
¶ Innopolis University, Russia
m.mazzara@innopolis.ru

***Abstract*—Microservices have seen their popularity blossoming with an explosion of concrete applications in real-life software. Several companies are currently involved in major refactoring of their back-end systems in order to improve scalability. In this paper, we present an experience report of a real world case study in order to demonstrate how scalability is positively affected by re-implementing a monolithic architecture into microservices. The case study is based on the *FX Core* system, a mission critical system of Danske Bank, the largest bank in Denmark and one of the leading financial institutions in Northern Europe.**

***Index Terms*—Microservices, Software Architecture, Scalability.**

microservice is expected to implement a single *business capability*, in fact a very limited system functionality, bringing benefits in terms of service maintainability and extendability. Since each microservice represents a single business capability, which is delivered and updated independently, discovering bugs or adding minor improvements do not have any impact on other services and on their releases. In common practice, it is also expected that a single service can be developed and managed by a single team [10]. The idea to have a team working on a single microservice is rather appealing: to build a system with a modular and loosely coupled design,

https://backend.orbit.dtu.dk/ws/portalfiles/portal/192883777/monolithic_microservices_experience.pdf

V. Conclusions

The re-engineering of the system discussed in this paper led to *reduced complexity*, *lower coupling*, *higher cohesion* and a *simplified integration*. The large components of the monolithic architecture, which were highly coupled, had overlapping responsibilities and integrated in a multitude of ways, have been substituted by several independent microservices. As a direct consequence, the size of the services is now generally smaller when compared to the large components of the monolith. Since services implement focused functionalities, even their names reveal to a large extent their responsibilities, detail which was previously absent. Services have now *reduced feature overlapping*. For example, functionalities such as trade-registration and line-checks in the monolithic architecture were handled by both *ForexAPI* and *RequestService*. The reason of this had to be found in the historical development of the system. After the analysis and the re-engineering, the microservice architecture features a *TradingService* and a *LineCheckService* that independently handle these functions. Furthermore, with a *polyglot architecture*, i.e. not technology dependent, the development team is no longer dependent on the *.NET* platform or MS SQL databases. Instead, services can be implemented in any language.

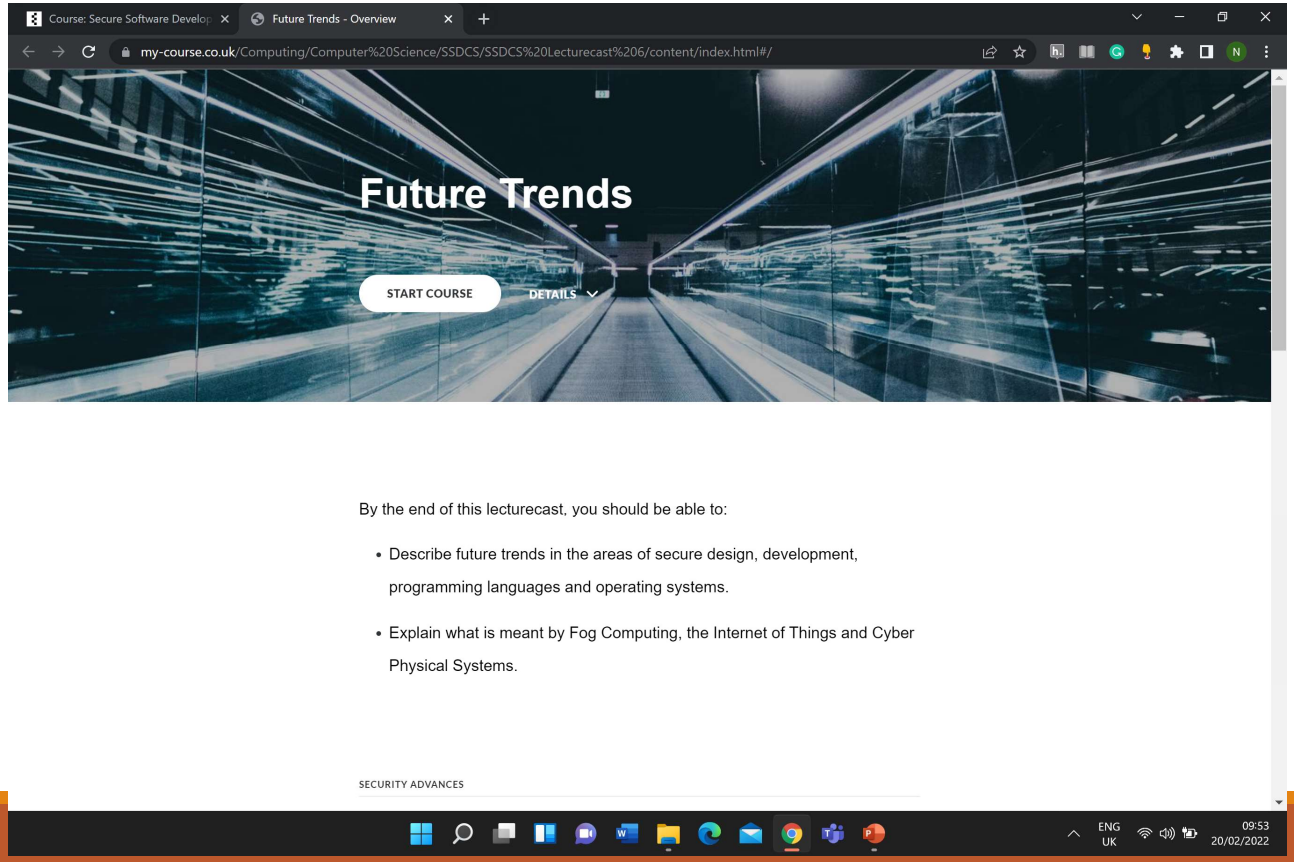
The future will see a growing attention regarding the matters discussed here and the development of new programming languages intended to address the microservice paradigm [5]. Languages for microservices should be able to model microservices in a *uniform way* and at a level of abstraction that also allows for their easy interconnection [2].

Seminar Preparation

Read Biggs et al (2018) and Bucchiarone et al (2018) as examples of modern views and approaches to the Monolithic vs. Microservices/ Microkernel debate.

- Post your team's stance to the forum along with justifications.
- Read all the arguments for each position.
- Choose one team response that disagrees with your team stance and post a message that refutes their argument.
- During this week's seminar session, all students will independently vote for which argument they believe was presented most persuasively.

Screenshots of Lecture Notes (Peoples, 2022, UoEO Lecture Cast) for Unit 12



Course: Secure Software Develop | Future Trends - Security

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/nj2ean9cNQAjgPg5MFA...

Many systems still use woefully inadequate and insecure passwords to access high value online resources even though much better and more secure methods exist, such as public key infrastructure and key exchange, multi-factor authentication or, as a minimum, enforcing strong passwords). Another example is the use of secure protocols across the Internet.

The latest secure HTTP protocol (HTTP/3) is both faster and more secure by default, however current adoption is low (less than 10% of web sites and 3% of browsers support HTTP/3 by default) and a large proportion of sites still default to old and insecure protocols, although many modern browsers block access to these insecure sites (Ghedini, 2018).

From a procedures and processes perspective...

...there needs to be a concerted effort to ensure that Internet protocols use the latest available and secure versions. In addition, all software – but browsers and operating systems in particular - should be secure by design (as they present a large attack surface and can make attacks on other components easier). As mentioned previously, software should also be kept up to date with the latest patches.

ENG UK 10:46 20/02/2022

Course: Secure Software Develop | Future Trends - Blockchain: a sec

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/LUSF3kS96Ioedm5CF15B...

Future Trends

6% COMPLETE

Blockchain is a relatively new technology that brings together encryption, distributed peer-to-peer networks and consensus mechanisms. In its most basic form it is a distributed, secure electronic ledger, which is useful to track exchanges or perform virtual escrow, as discussed later.

Its strength is that it can immutably track transactions and maintain a complete history of them. It consists of a large database of interconnected blocks – each block represents a public key that holds a history of all transactions made by that key's owner. Due to the distributed nature of the network, there are a number of nodes (such as desktop computers, servers and even laptops) each of which retains a complete copy of the blockchain.

When a transaction is carried out...

...a message is broadcast to all the connected nodes active on the network at that time to confirm the validity of the transaction. The consensus algorithm is used to ensure that the transaction is valid based on the agreement of the majority of nodes (where majority is defined as part of the consensus algorithm).

ENG UK 10:48 20/02/2022

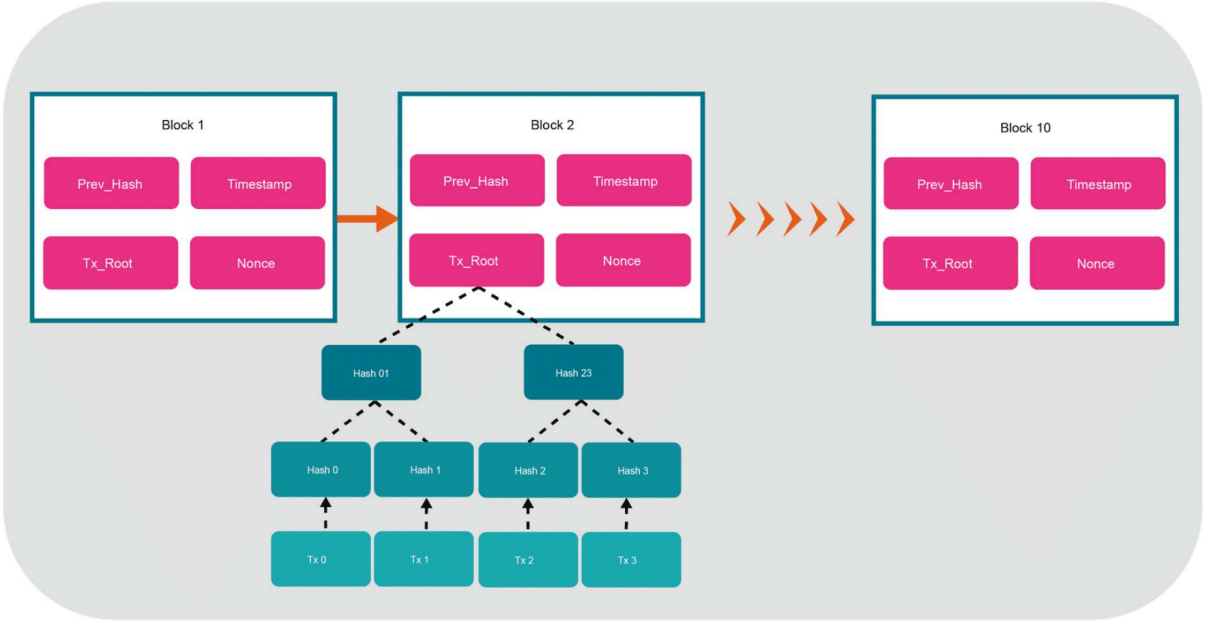
Course: Secure Software Develop | Future Trends - Blockchain: a sec

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/LUSF3kS96Ioedm5CF15B...

Future Trends

6% COMPLETE

The accompanying diagram shows a representation of a blockchain:



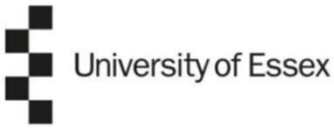
(Hakak et al, 2020)

ENG UK 10:48 20/02/2022

However, blockchain is not a silver bullet. As previously described, depending on the platform, it can be hugely expensive (in terms of hardware and cost of power) to validate transactions. As discussed in the introduction, power usage and efficiency of algorithms is becoming an increasingly important issue in IT systems. including blockchain calculations, and there is ongoing research to make these calculations more efficient (Stoll et al, 2019).

There have been a number of very high-profile incidents (such as the mt.gox security breach and loss of over $450M of bitcoins). Arguably the introduction of cryptocurrencies has also led to an increase in ransomware attacks (as many such attacks demand payment in such currencies) (Goodin, 2013).

CONTINUE



Security in Architecture, Design and the SDLC

Architecturally, design models have moved from the monolithic through to micro-scale (for both OSes and applications) but there needs to be further progression to deal with the integration of Internet of Things (IoT) devices, artificial intelligence (AI) usually hosted in the cloud, and fog computing that ties the two together. There are several design challenges with this new paradigm: should each part be designed in isolation and then connected via loosely coupled network connections? Or should there be a holistic design, treating each component as a small part of the whole, designing the overall system to maximise network throughput and resource utilisation?

Course: Secure Software Develop | Future Trends - Security in Archit

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/UQf7TyoL82rc2O5gVj0w...

SECURITY ADVANCES

- Security
- Blockchain: a security panacea?
- Security in Architecture, Design & the SDLC

ARCHITECTURE & DESIGN

- Cyber Physical Systems (CPSs)
- Fog Computing
- Security & SDLC
- DevOps

Finally...

...security has always been a major concern for users and operations staff, if not always for designers.

Although there have been add-ons and extensions for architectural frameworks (such as SABSA for TOGAF), and even SDLC methodologies designed with security built-in (such as Spiral, one of the first waterfall methods with integrated security, and still in use for some projects today), the flexibility of many of the agile methodologies has meant that security considerations have taken a backseat, often to the detriment of the overall system. However, with the advent of DevOps, the introduction of DevSecOps may mean that security at last gets full consideration in the lifecycle/ pipeline.

CONTINUE

ENG UK 10:54 20/02/2022

Course: Secure Software Develop | Future Trends - Cyber Physical Sy

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/mvMF-Z6bC-iqEsOeKMT...

Future Trends

18% COMPLETE

SECURITY ADVANCES

- Security
- Blockchain: a security panacea?
- Security in Architecture, Design & the SDLC

ARCHITECTURE & DESIGN

- Cyber Physical Systems (CPSs)
- Fog Computing
- Security & SDLC
- DevOps

Lesson 4 of 17

Cyber Physical Systems (CPSs)

Increasingly...

...many of the systems we use today can be considered to be Cyber Physical Systems (CPSs) – that is they were designed to interact with, and to include humans as part of the system.

There are a number of attributes and characteristics that identify a CPS:

ENG UK 10:54 20/02/2022

Course: Secure Software Develop | Future Trends - Cyber Physical Sy

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/mvMF-Z6bC-iqEsOeKMT...

Future Trends

18% COMPLETE

- SECURITY ADVANCES
  - Security
  - Blockchain: a security panacea?

This combination of characteristics needs a very different set of design tools than would be used to design a traditional web application or a database application. They will need to include or integrate aspects of model-based design (MBD), data driven design (DDD), machine learning (ML – that includes the capability for proving system correctness), as well as advanced security threat models.

Examples of CPSes include:

1. Smart energy systems.
2. Advanced driver assistance systems (ADAS).

(Seshia et al, 2017)

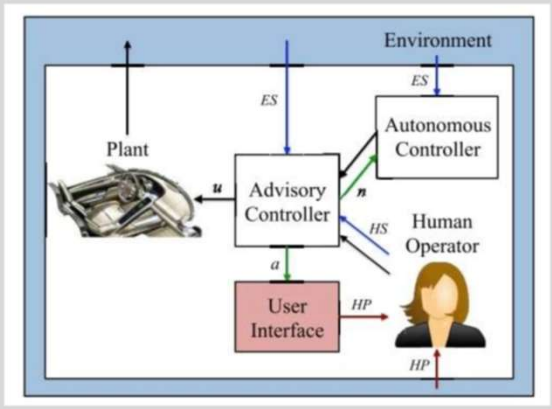


Diagram sourced from: Seshia et al, 2017. https://par.nsf.gov/servlets/purl/10040007

An example of a human-in-the-loop controller is illustrated in the diagram below, where ES is environmental sensing/ sensor; HS is human sensing; HP is human perception; u is the control input to the vehicle; a is an advisory signal to the user; n is a notification from the advisory controller to the autonomous controller (Seshia et al, 2017). An example of this in use is when a driver is in a semi-autonomous vehicle. Here, the autonomous controller can manage the vehicle in an emergency, but the driver can take control at any time while the advisory controller mediates between the two modes of operation.

ENG UK 10:56 20/02/2022

Course: Secure Software Develop | Future Trends - Fog Computing

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/TbLhdZt-pEogozc5j-or6...

Fog computing is a term that brings together the vast compute and storage resources of the cloud, the flexibility and security of edge nodes, and the distributed, micro-powered pervasiveness of the internet of things (IoT). The Fog Computing concept was introduced in a paper by Bonomi et al (2012).

The diagram below shows the different layers of the conceptual model:

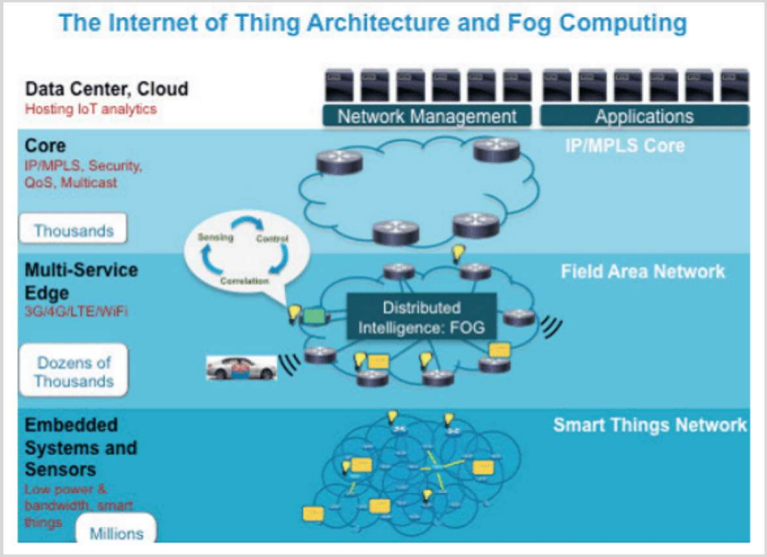


Diagram sourced from: Bonomi et al, 2012 paper https://doi.org/10.1145%2F2342509.2342513

ENG UK 10:58 20/02/2022

Venticinque & Amato (2018) present a methodology for deploying IoT applications in a fog architecture. Cooja (Gaurav, 2017) is an IoT network simulator that can be used to design, model and test IoT networks.

Security & privacy are quite possibly the two major concerns around fog computing deployments.

A typical fog system consists of;

- cloud,
- edge,
- and IoT components.

This means that the attack surface is vast.

Course: Secure Software Develop | Future Trends - Security & SDLC

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/JDHEZQfGiLJUytPVEi28...

Future Trends

29% COMPLETE

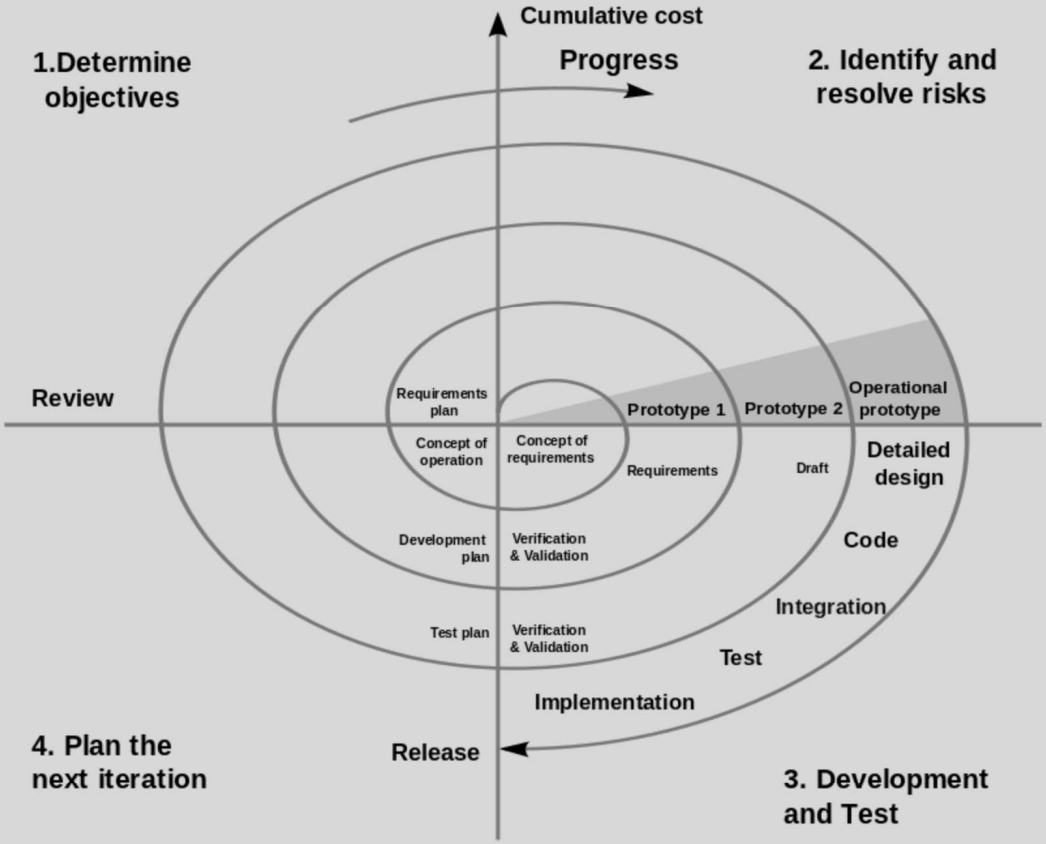


Diagram 1: Spiral Model Diagram

ENG UK 11:03 20/02/2022

OWASP CLASP

The Open Web Application Security Project (OWASP) has been discussed several times during this module as a framework for design, development and testing. In addition, OWASP provide the Comprehensive Lightweight Application Security Process (CLASP) which was originally developed by Viega (2005).

The basic CLASP methodology consists of:

STEP 1

Identify system roles and resources.

STEP 2

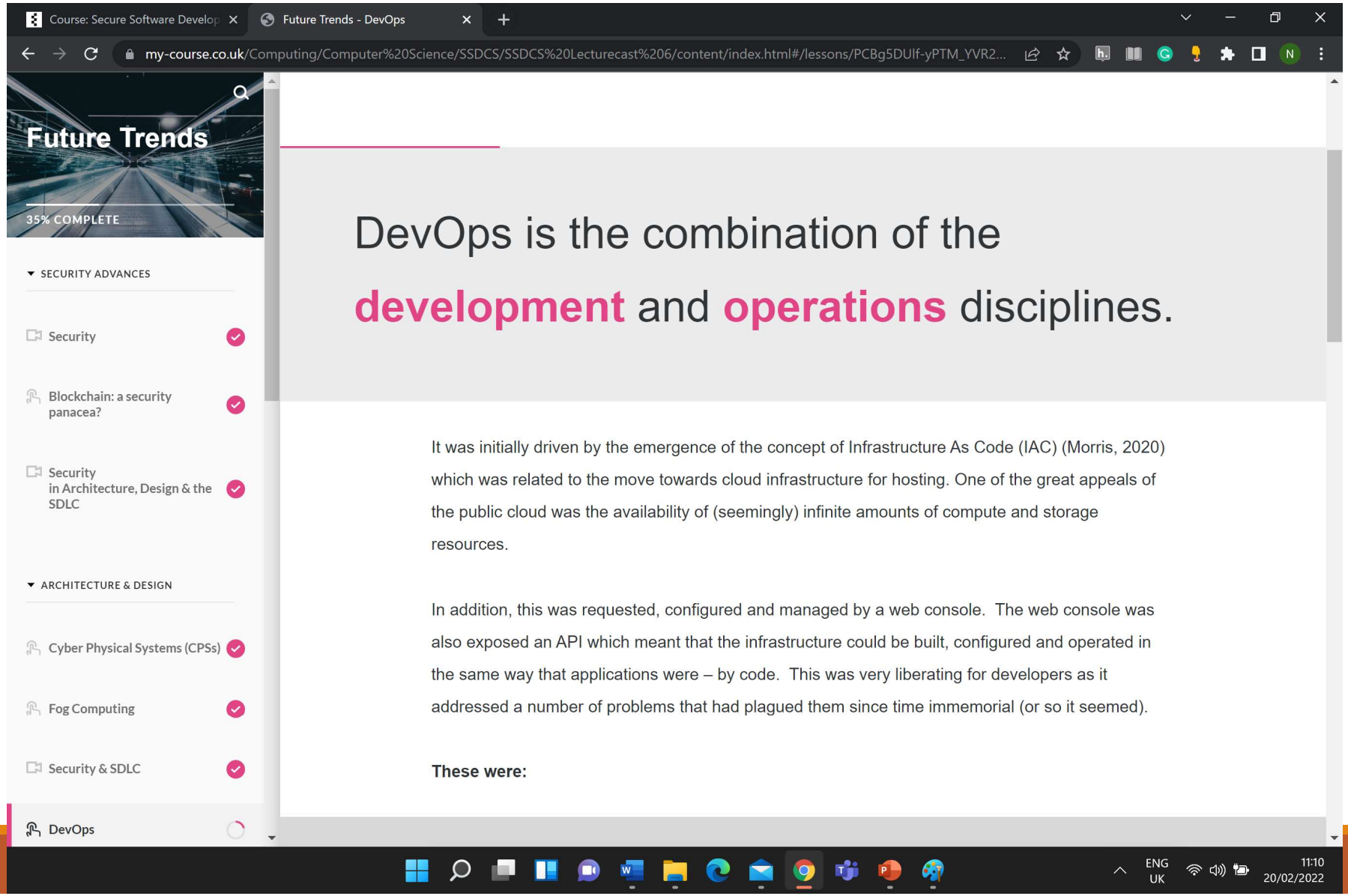
Categorize resources into abstractions.

STEP 3

Identify resource interactions through the lifetime of the system.

STEP 4

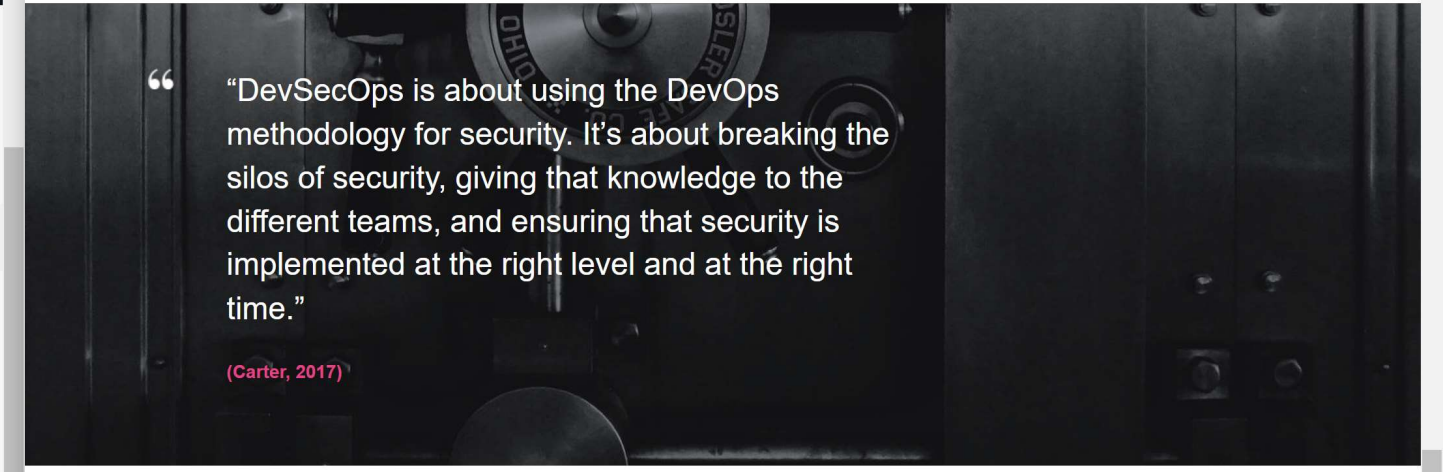
For each category, specify mechanisms for addressing each core security services.


Course: Secure Software Develop
Future Trends - DevOps
my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/PCBg5DUlf-yPTM_YVR2...
Future Trends
35% COMPLETE
SECURITY ADVANCES
Security
Blockchain: a security panacea?
Security in Architecture, Design & the SDLC
ARCHITECTURE & DESIGN
Cyber Physical Systems (CPSs)
Fog Computing
Security & SDLC
DevOps
DevOps is the combination of the development and operations disciplines.
It was initially driven by the emergence of the concept of Infrastructure As Code (IAC) (Morris, 2020) which was related to the move towards cloud infrastructure for hosting. One of the great appeals of the public cloud was the availability of (seemingly) infinite amounts of compute and storage resources.
In addition, this was requested, configured and managed by a web console. The web console was also exposed an API which meant that the infrastructure could be built, configured and operated in the same way that applications were – by code. This was very liberating for developers as it addressed a number of problems that had plagued them since time immemorial (or so it seemed).
These were:
ENG
UK
11:10
20/02/2022

Course: Secure Software Develop | Future Trends - DevOps

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/PCBg5DUlf-yPTM_YVR2...

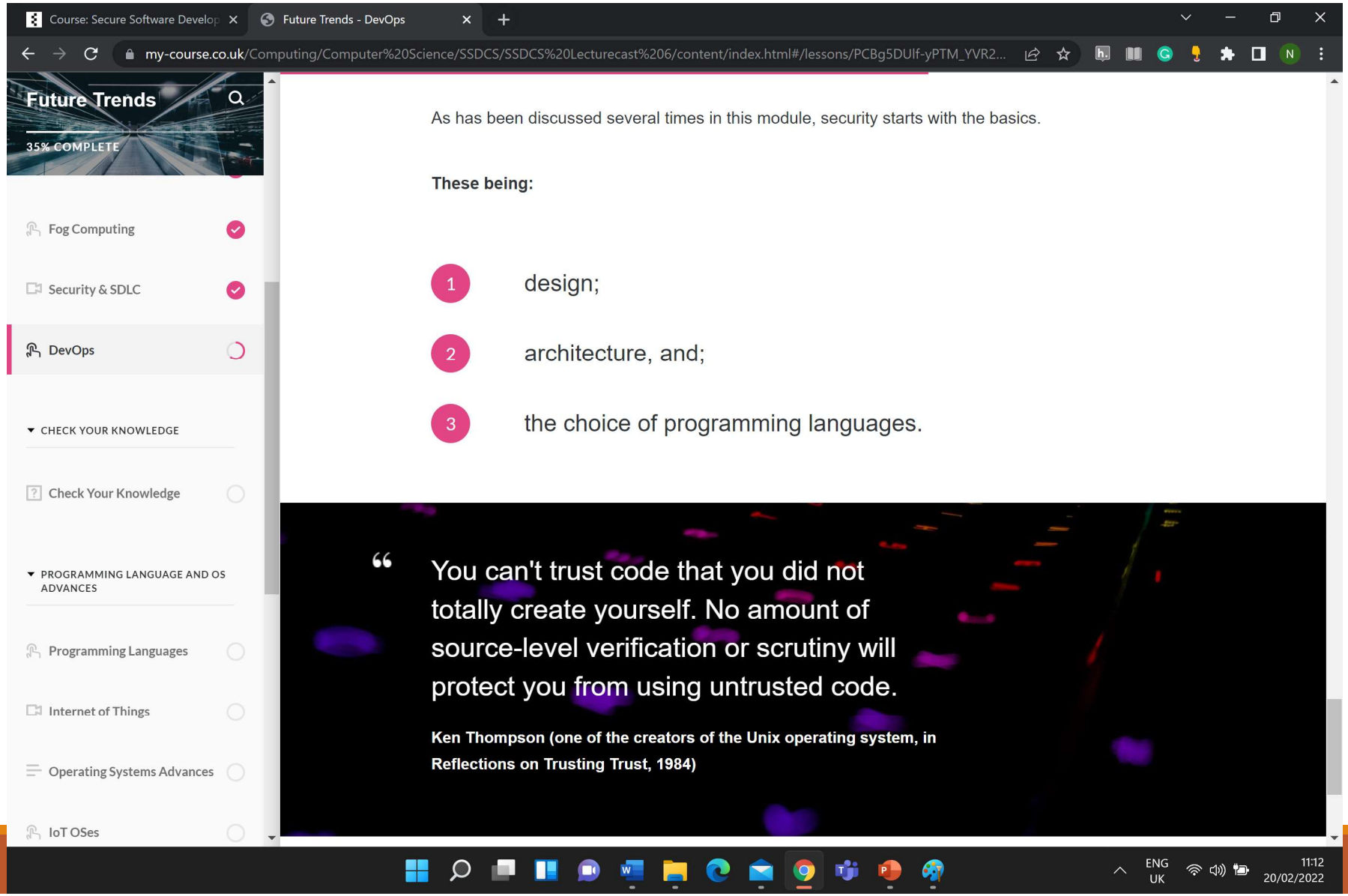
Raynaud, an Enterprise Security Architect, in an interview with Carter (2017) said:

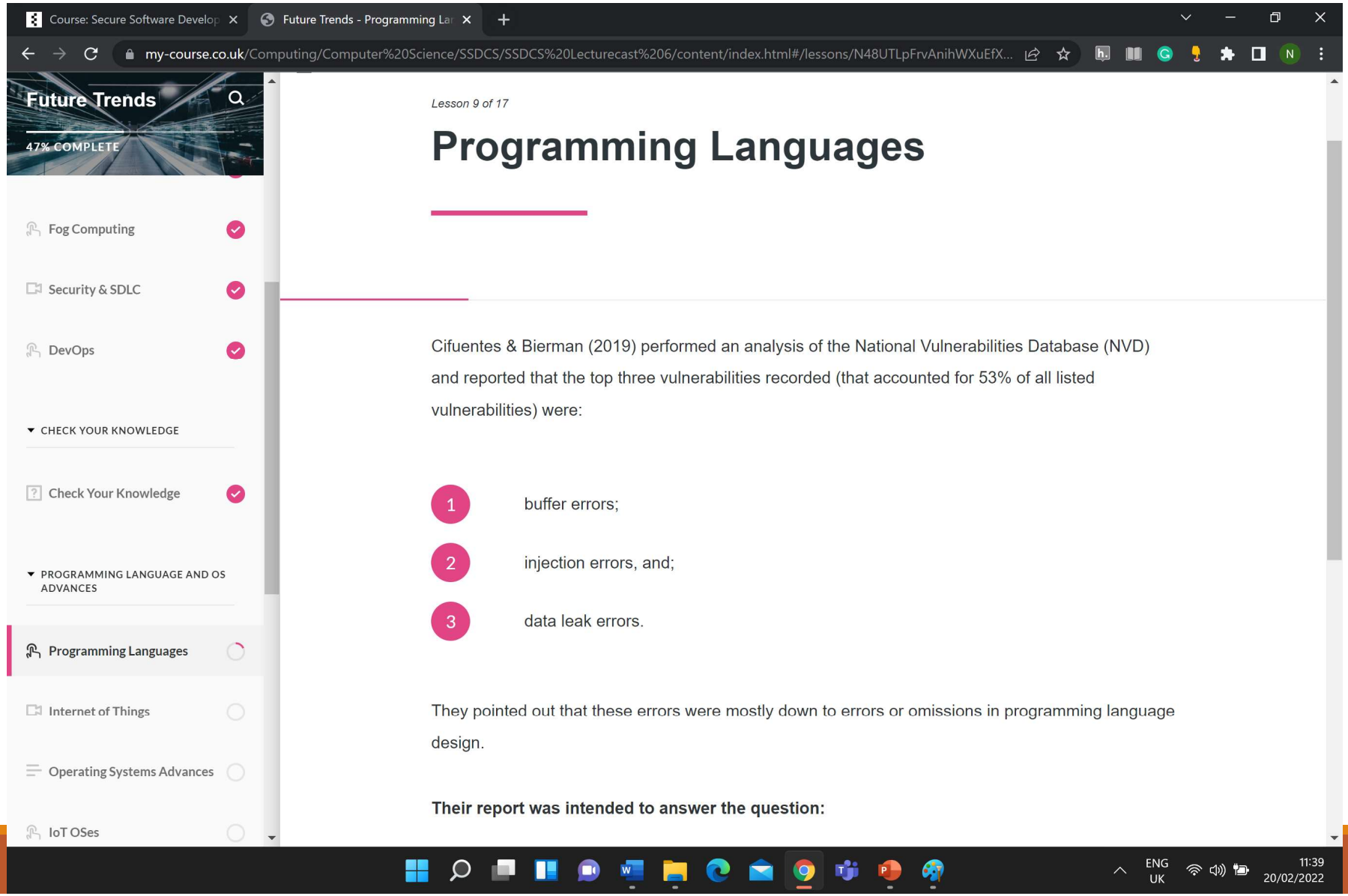


DevSecOps is about integrating security, both through automation as well as human checkpoints into the pipeline described above. The nature of a DevOps pipeline means that automated testing tools, such as:

- linters,
- static code checkers,
- and dynamic code checkers.

ENG UK 11:10 20/02/2022


Course: Secure Software Develop
Future Trends - DevOps
my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/PCBg5DUlf-yPTM_YVR2...
Future Trends
35% COMPLETE
Fog Computing
Security & SDLC
DevOps
CHECK YOUR KNOWLEDGE
Check Your Knowledge
PROGRAMMING LANGUAGE AND OS ADVANCES
Programming Languages
Internet of Things
Operating Systems Advances
IoT OSes
As has been discussed several times in this module, security starts with the basics.
These being:
1 design;
2 architecture, and;
3 the choice of programming languages.
“ You can't trust code that you did not totally create yourself. No amount of source-level verification or scrutiny will protect you from using untrusted code.
Ken Thompson (one of the creators of the Unix operating system, in Reflections on Trusting Trust, 1984)
ENG
UK
11:12
20/02/2022


Course: Secure Software Develop
Future Trends - Programming Lar
my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/N48UTLpFrvAnihWXuEfX...
Future Trends
47% COMPLETE
Fog Computing
Security & SDLC
DevOps
CHECK YOUR KNOWLEDGE
Check Your Knowledge
PROGRAMMING LANGUAGE AND OS ADVANCES
Programming Languages
Internet of Things
Operating Systems Advances
IoT OSes
Lesson 9 of 17
Programming Languages
Cifuentes & Bierman (2019) performed an analysis of the National Vulnerabilities Database (NVD) and reported that the top three vulnerabilities recorded (that accounted for 53% of all listed vulnerabilities) were:
1 buffer errors;
2 injection errors, and;
3 data leak errors.
They pointed out that these errors were mostly down to errors or omissions in programming language design.
Their report was intended to answer the question:
ENG
UK
11:39
20/02/2022

Course: Secure Software Develop | Future Trends - Programming Lar

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/N48UTLpFrvAnihWXuEfX...

Future Trends

53% COMPLETE

- Fog Computing
- Security & SDLC
- DevOps

CHECK YOUR KNOWLEDGE

- Check Your Knowledge

PROGRAMMING LANGUAGE AND OS ADVANCES

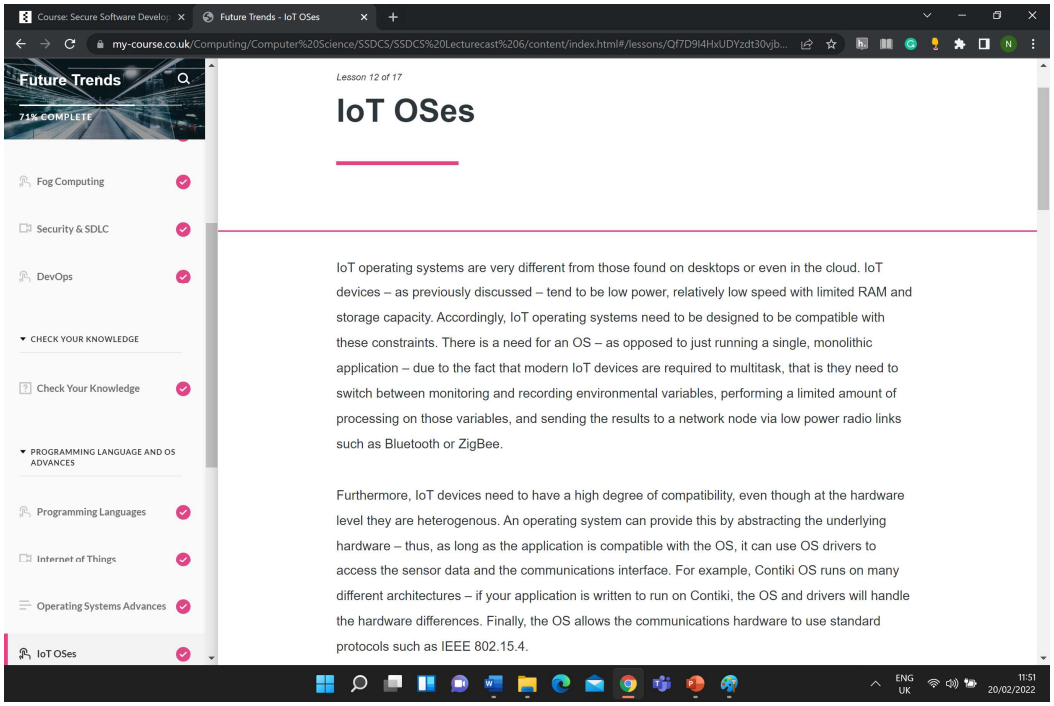
- Programming Languages
- Internet of Things
- Operating Systems Advances
- IoT OSes

Unfortunately, none of the modern languages reviewed support taint tracking or faceted data (although as Cifuentes & Bierman (2019) noted, some languages such as Ruby and Perl support taint tracking, and there are packages for Python that support faceted data).

The conclusion is that, although modern languages support several functions that makes secure programming easier and takes the developer one step closer to truly safe applications, there is no silver bullet yet. So, language development still needs to address even the limited list produced by Cifuentes and Bierman in 2019.

Lesson 10 - Internet of Things

ENG UK 11:44 20/02/2022


Course: Secure Software Develop
Future Trends - IoT OSes
my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/Qf7D9l4HxUDYzdt30vjb...
Future Trends
71% COMPLETE
Fog Computing
Security & SDLC
DevOps
CHECK YOUR KNOWLEDGE
Check Your Knowledge
PROGRAMMING LANGUAGE AND OS ADVANCES
Programming Languages
Internet of Things
Operating Systems Advances
IoT OSes
Lesson 12 of 17
IoT OSes
IoT operating systems are very different from those found on desktops or even in the cloud. IoT devices – as previously discussed – tend to be low power, relatively low speed with limited RAM and storage capacity. Accordingly, IoT operating systems need to be designed to be compatible with these constraints. There is a need for an OS – as opposed to just running a single, monolithic application – due to the fact that modern IoT devices are required to multitask, that is they need to switch between monitoring and recording environmental variables, performing a limited amount of processing on those variables, and sending the results to a network node via low power radio links such as Bluetooth or ZigBee.
Furthermore, IoT devices need to have a high degree of compatibility, even though at the hardware level they are heterogenous. An operating system can provide this by abstracting the underlying hardware – thus, as long as the application is compatible with the OS, it can use OS drivers to access the sensor data and the communications interface. For example, Contiki OS runs on many different architectures – if your application is written to run on Contiki, the OS and drivers will handle the hardware differences. Finally, the OS allows the communications hardware to use standard protocols such as IEEE 802.15.4.
ENG
UK
11:51
20/02/2022

Course: Secure Software Develop | Future Trends - Cloud OS Advanc

my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/x7GUeO6TSCtAZpE8Ves...

Future Trends

71% COMPLETE

- Fog Computing
- Security & SDLC
- DevOps

CHECK YOUR KNOWLEDGE

- Check Your Knowledge

PROGRAMMING LANGUAGE AND OS ADVANCES

- Programming Languages
- Internet of Things
- Operating Systems Advances
- IoT OSes

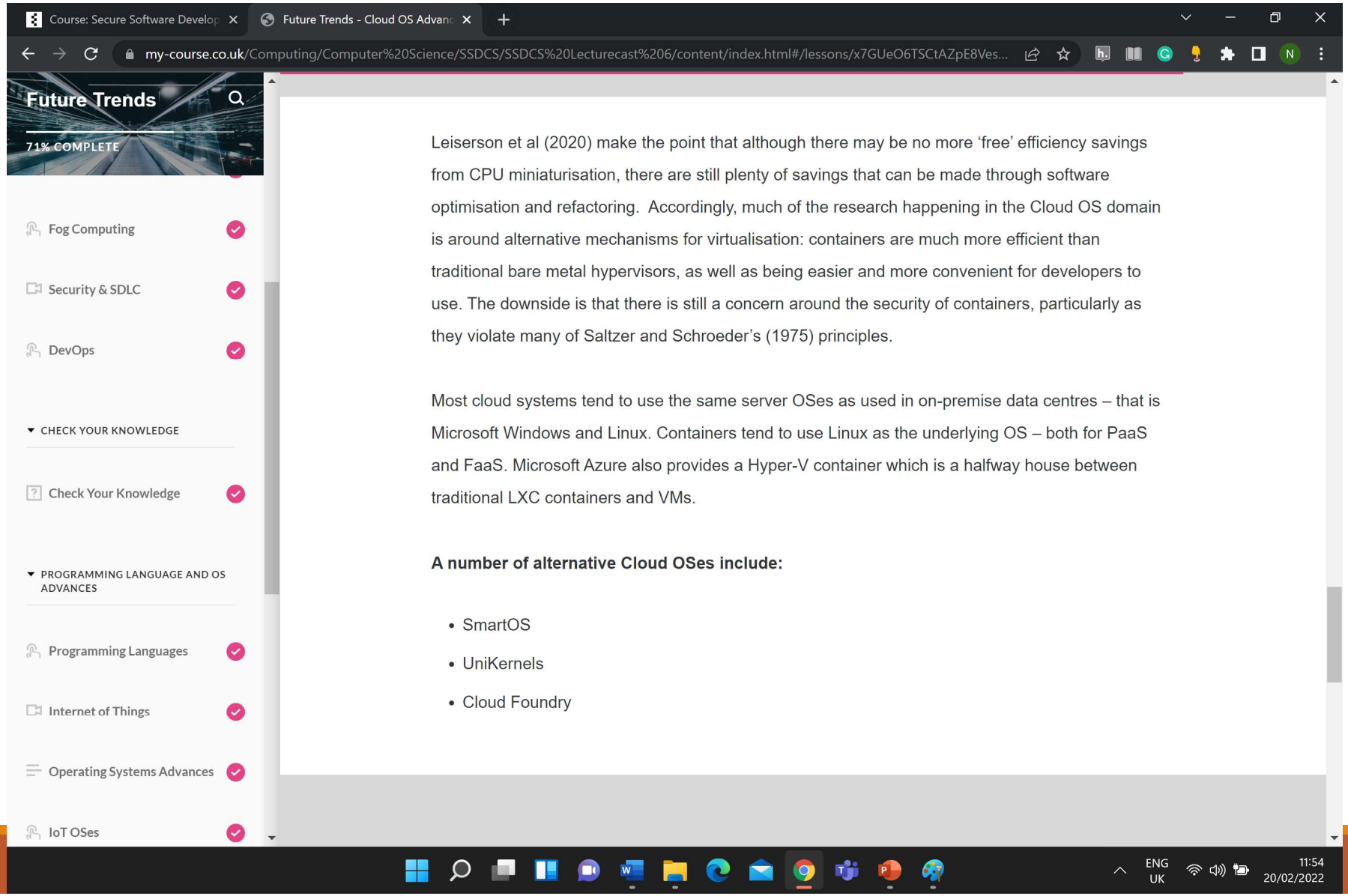
Lesson 13 of 17

Cloud OS Advances & Alternatives

As discussed in a previous section, the Cloud is big business.

Hostingtribunal.com (Galov, 2020) reports that 83% of enterprise workloads will be hosted in the cloud by the end of 2020 – split between public (41%), private (20%) and hybrid (22%) clouds. Cloud adoption still focuses on three models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) although the use of so called 'serverless computing' (a.k.a Function as a Service FaaS) is on the rise (Slater, 2017).

ENG UK 11:51 20/02/2022


Course: Secure Software Develop
Future Trends - Cloud OS Advanc
my-course.co.uk/Computing/Computer%20Science/SSDCS/SSDCS%20Lecturecast%206/content/index.html#/lessons/x7GUeO6TSCtAZpE8Ves...
Future Trends
71% COMPLETE
Fog Computing
Security & SDLC
DevOps
CHECK YOUR KNOWLEDGE
Check Your Knowledge
PROGRAMMING LANGUAGE AND OS ADVANCES
Programming Languages
Internet of Things
Operating Systems Advances
IoT OSes
Leiserson et al (2020) make the point that although there may be no more 'free' efficiency savings from CPU miniaturisation, there are still plenty of savings that can be made through software optimisation and refactoring. Accordingly, much of the research happening in the Cloud OS domain is around alternative mechanisms for virtualisation: containers are much more efficient than traditional bare metal hypervisors, as well as being easier and more convenient for developers to use. The downside is that there is still a concern around the security of containers, particularly as they violate many of Saltzer and Schroeder's (1975) principles.
Most cloud systems tend to use the same server OSes as used in on-premise data centres – that is Microsoft Windows and Linux. Containers tend to use Linux as the underlying OS – both for PaaS and FaaS. Microsoft Azure also provides a Hyper-V container which is a halfway house between traditional LXC containers and VMs.
A number of alternative Cloud OSes include:
SmartOS
UniKernels
Cloud Foundry
ENG
UK
11:54
20/02/2022