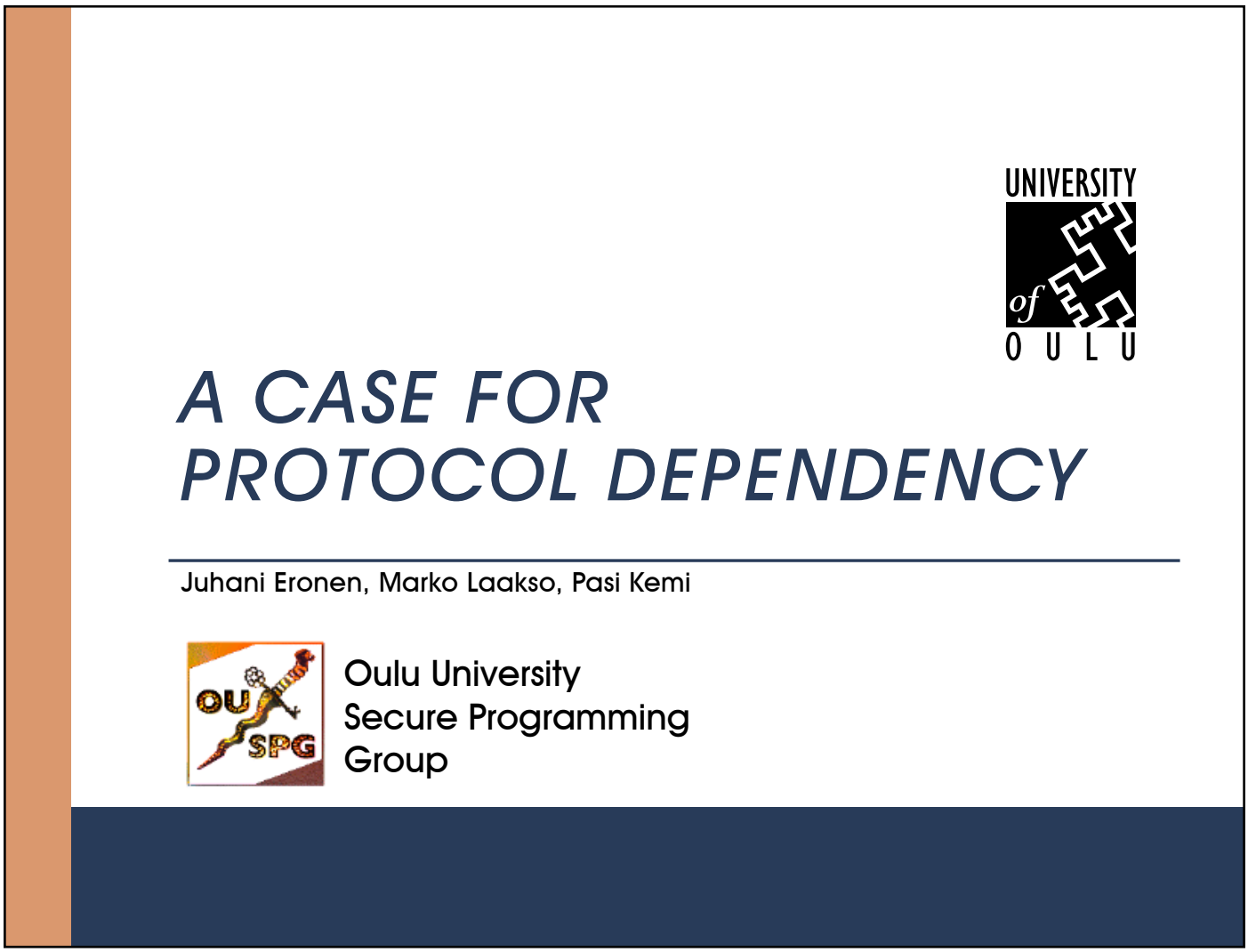
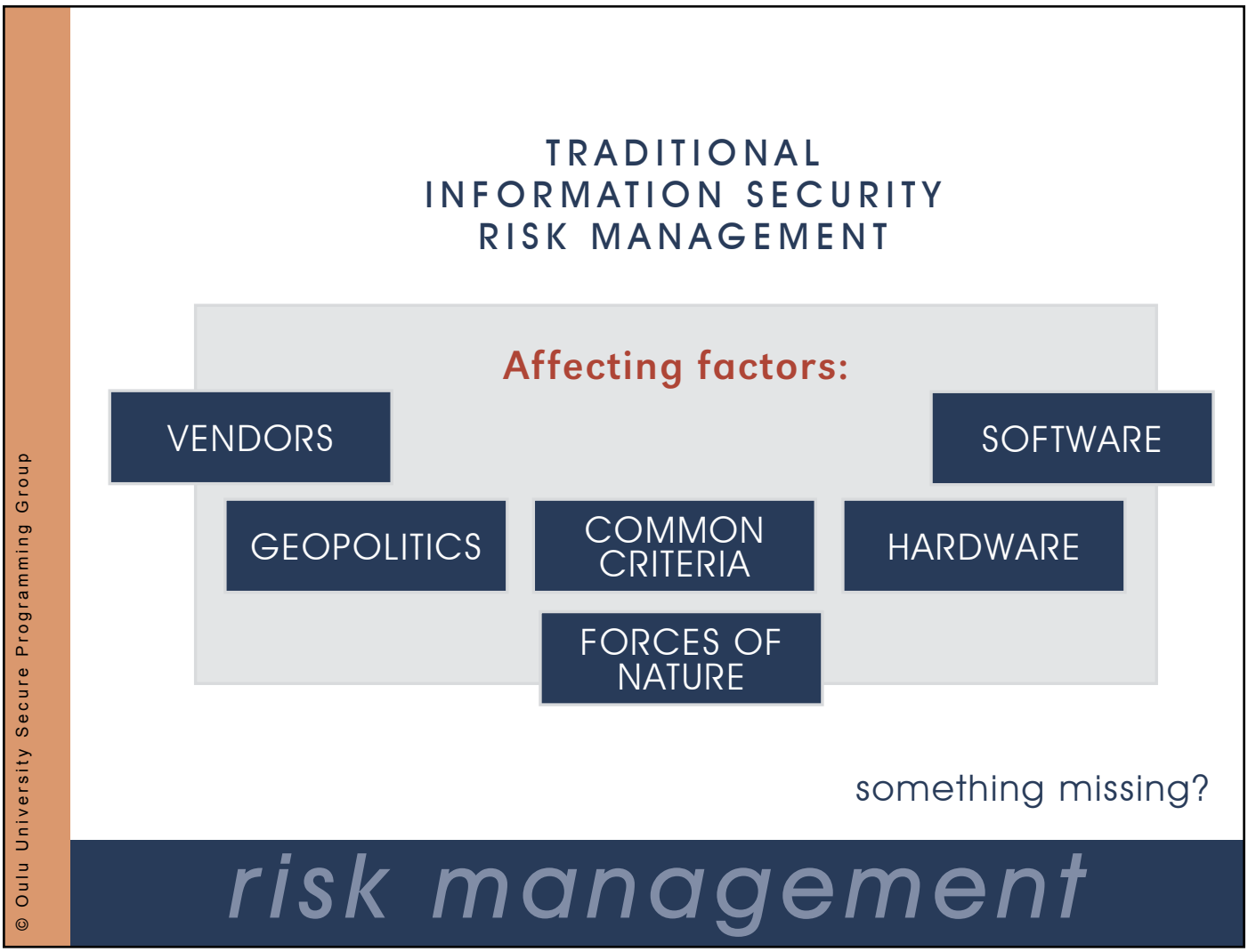


Slide 1: A Case for Protocol Dependency

Vulnerabilities infest information technology. There is a lack of tools in risk assessment for understanding the impact that the disclosed vulnerabilities have on the critical information infrastructures.

To address this need, this work derives a new dimension of dependency from practical vulnerability work, namely that of protocol dependency. Classic technology dependency views were reviewed, a chain of systematic vulnerability disclosures was followed as a case study and analysis revealed evidence of protocol dependency.

Extrapolating from the experiences of a complex case, this new dependency dimension can be modelled. The model will benefit from going beyond a narrow technical view.



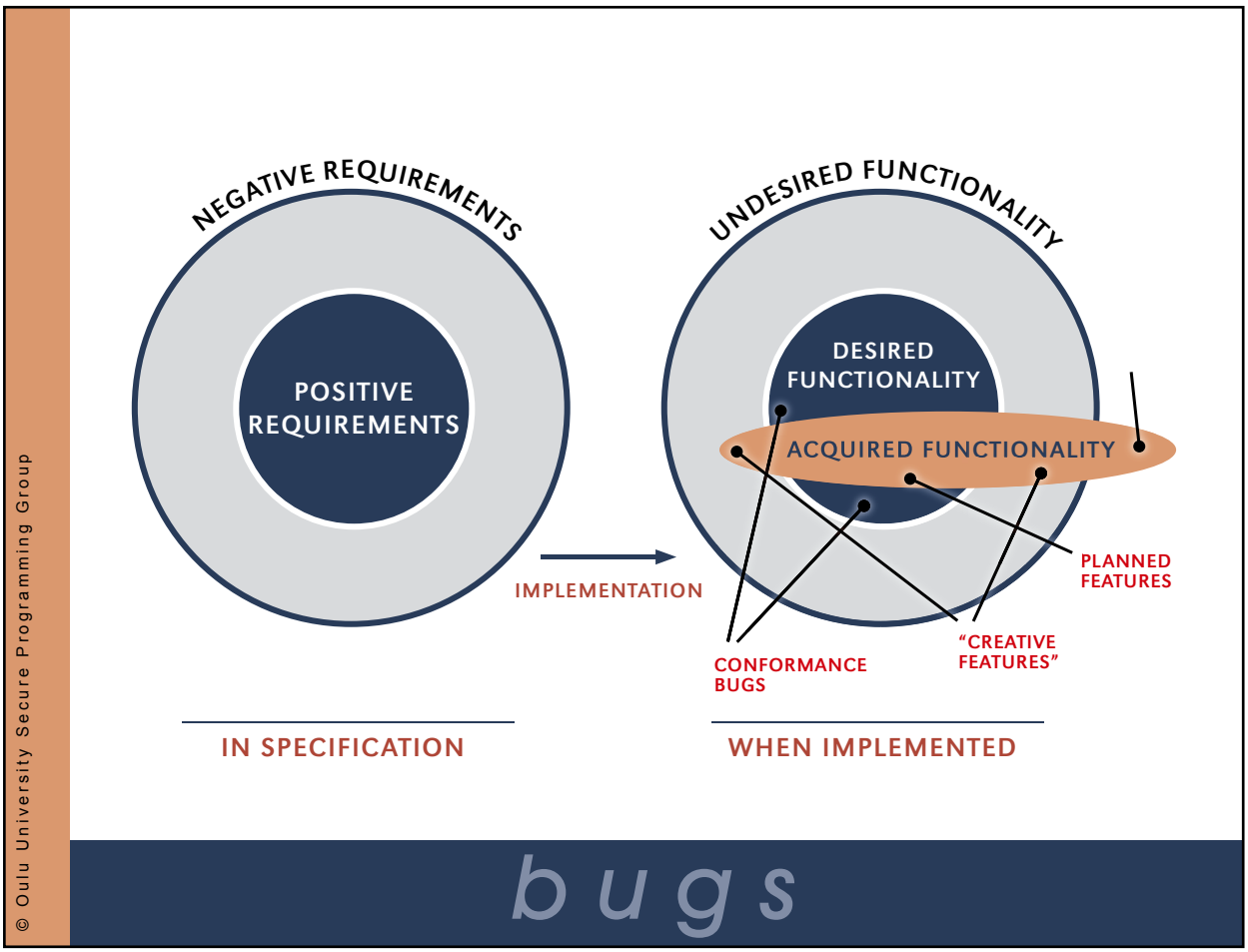
Slide 2: Risk Management

Risk management is used in practically all current organisations and enterprises to protect their assets and ensure their continued operation. Traditional risk management of information security focuses on many concrete aspects such as technological redundancy, availability of hardware and spare parts, geopolitical factors, the certification of software and hardware, assuring the energy feed of hardware and so on.

However, as current information infrastructures consist of several interconnected infrastructures, the study of dependencies within and between these infrastructures is essential for mitigating risks caused by the vulnerabilities of an infrastructure.

There has been some research on the interdependencies of different critical infrastructure sectors and their technical and managerial layers. Some analyses have gone as far as identifying critical information technology components and their interdependencies with other components, or using historical data to model the dynamic behaviour of an interconnected system.

Understanding protocol dependency is a more subtle view on the technologies that form the information infrastructure. It complements the known methods and opens deeper views into the technological dependency of the critical infrastructure.



Slide 3: Bug

When a a system is implemented, requirements are gathered for it, focusing on the positive requirements i.e. what the system should do.

Some security is specified with positive requirements (such as use of crypto etc.), but usually they represent design choices instead of actual needs.

Most of the security features are with negative requirements (i.e. what the system should not do). This is a problem, because negative requirements are not testable - you can only tell when they're not attained.

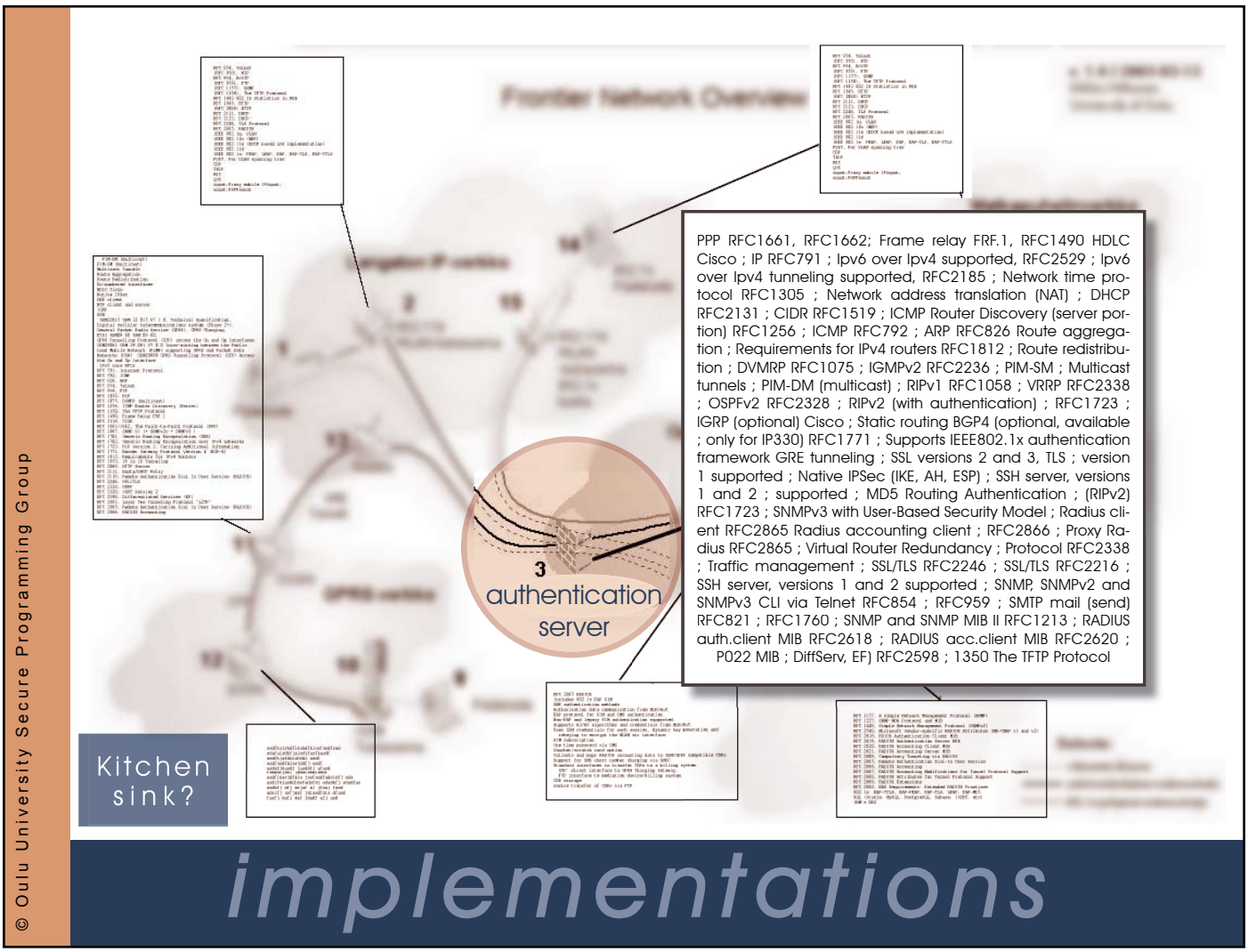
Implementation gives us more or less than we asked for, because the translation process from specification to implementation is not straightforward, including translations between natural and machine language representations, interpretations, technical decisions: choices of languages, tools etc.

Results: desired features - conformance bugs
creative features (not anticipated, even by the programmer)
undesired features -> bugs (with security implications)

Different implementations represent different ellipses on this map, interoperability is based on the narrow common ground shared between them.

Risk analysis of a single IT component is challenging due to the very nature of information technology. This way IT suffers from its continuous evolution and modification.

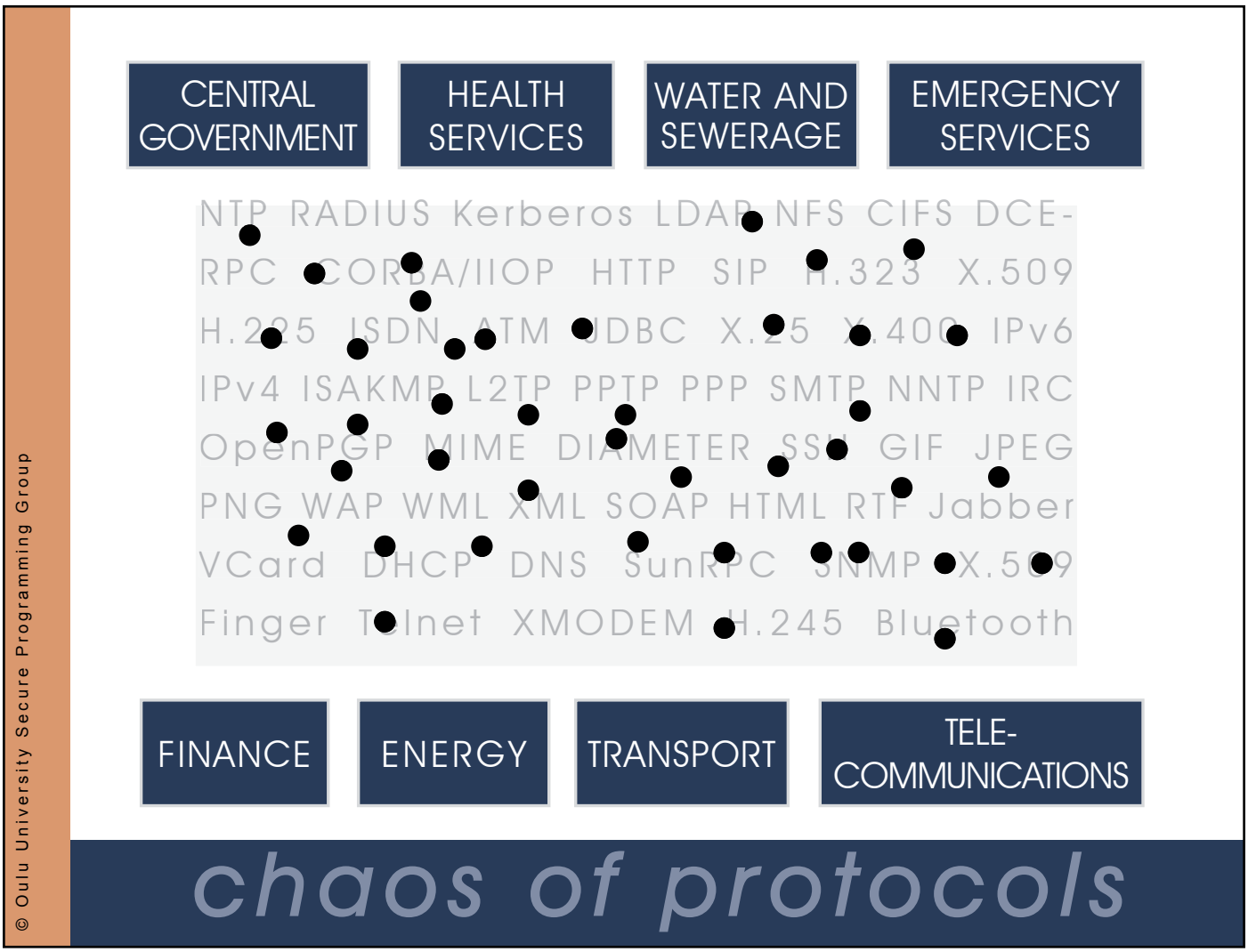
The need for differentiating between desired <-> actual functionality has been acknowledged e.g. by the US President's Information Technology Advisory Committee (PITAC)



Slide 4: Implementations

Risk analysis of larger IT systems is even more challenging than that of a single component. Currently, even the simplest of systems are typically connected to other systems and dependent on the services they provide, breeding complexity. Changes in the environment, components, architecture, and procedures related to connected systems create new risks and demand continuous reassessment of the prevalent risk assessment. Moreover, the current trend of ubiquitous interconnectivity between IT systems has resulted in widespread vulnerability, where continuously increasing levels and varieties of attacks have emerged.

A single failure in one part of the infrastructure can cascade throughout the network over a varying timespan and finally cause catastrophic failures in the infrastructure. The origin of the failure might not be evident due to the complexity of the interconnections. Thus, the study of dependencies is essential for mitigating risks caused by the vulnerabilities of an infrastructure.



Slide 5: Chaos of Protocols

Protocol dependency is one of the dependency types that plague IT systems. It creates a difficult practical problem: How to determine the impact of a disclosure of a vulnerability in a product, in certain types of products or in a more abstract concept such as a protocol or implementations of a certain protocol? There is no easy way to gather answers to the following questions:

1. If a product is affected, are similar products affected?

2. If a product is affected, are seemingly unrelated or different products affected?

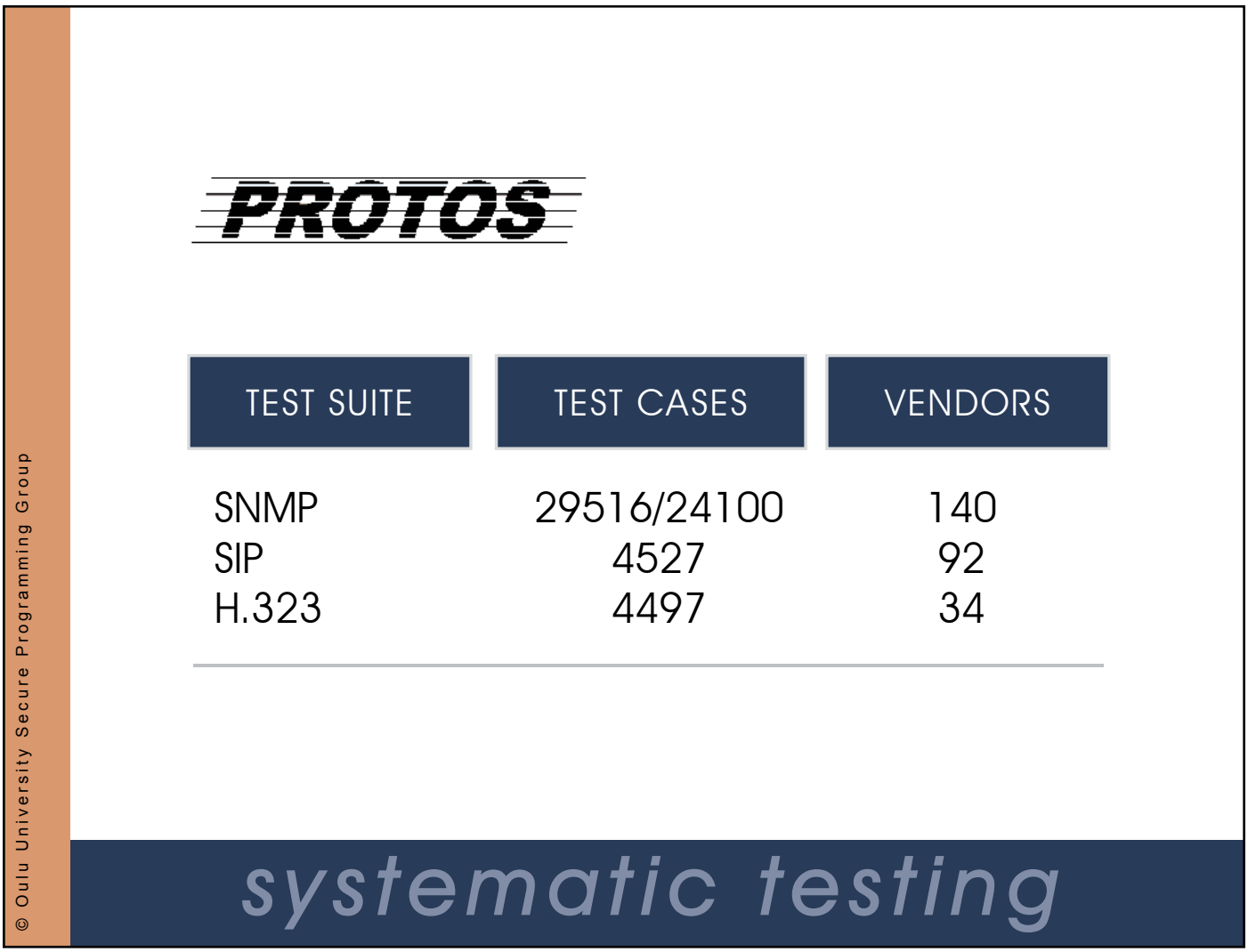
3. If a vulnerability is found in one networking context, e.g. the Internet, does it affect a different context, such as the telephone networks?

4. If a vulnerability affects desktop computing, are appliances with embedded software in danger?

In order to answer these questions we took the approach of vulnerability research, seeking common ground between seemingly different vulnerabilities.

Traditional vulnerability research is reactive, relying on a ‘penetrate & patch’ paradigm. Knowledge about common vulnerability types has been accumulated through years, and the remedies for them are known. These common vulnerabilities, that are based on simple programming errors, are still made daily.

OUSPG initiated the PROTOS project, claiming that common vulnerabilities are systematic and many of them could be eliminated with systematic testing.



Slide 6: Systematic Testing

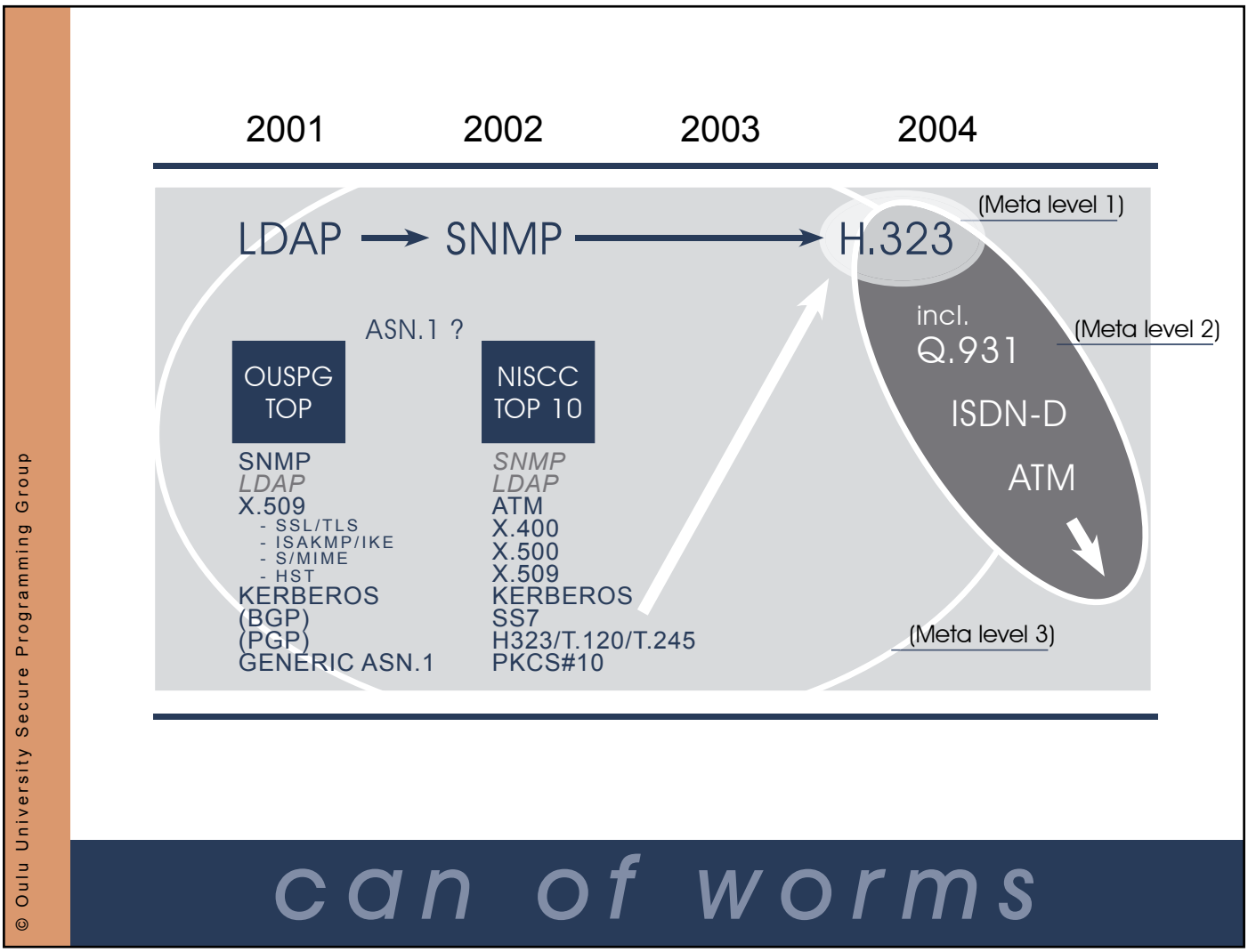
The method of PROTOS project was systematic black-box testing of protocols.

Protocols are interface languages (between software, functions, modules, soft-user). Black-box testing assumes we know nothing about the tested implementation, as is the case with most COTS software.

A mini-simulation of a protocol was created. Basically, it is just a syntax tree of the protocol augmented with rules that govern the handling of protocol data - length/checksum calculators, UDP/TCP injectors etc. Syntactical errors were introduced into protocols, and the protocol messages were used to test implementations. (Some call this kind of an approach fuzz testing.) A great number of vulnerabilities were found.

PROTOS tested a large number of protocols and protocol implementations, the handling of found vulnerability was co-ordinated by AusCERT, CERT/CC and NISCC (UK National Infrastructure Security Co-ordination Centre). Test suites included:

-Simple Network Management Protocol (SNMP)
-Session Initiation Protocol (SIP)
-H.323 (the video conferencing suite, tests concerned the H.225.0
protocol, responsible Call signalling, Registration, Admission and Status)



Slide 7: Can of Worms

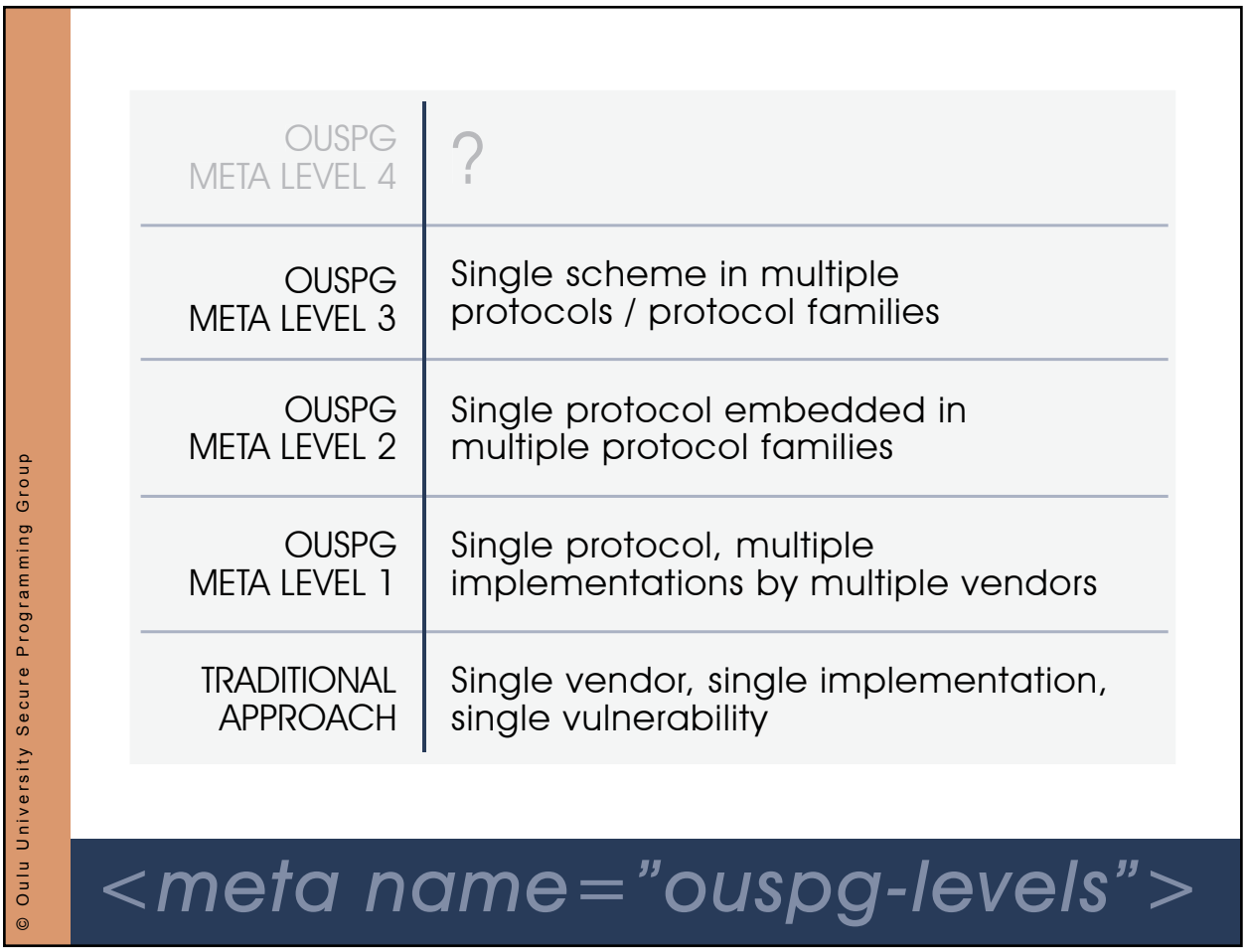
After the test material for LDAP it became suspect, for the PROTOS team, that there are implementation level vulnerabilities in various ASN.1 parsers, which are prevalent in protocol implementations. Initial analysis and observations supported this reasoning.

The PROTOS team compiled an internal list of core protocols to select the target protocol for the next test suite. Creation of an ASN.1 Basic Encoding Rules (BER) test suite was considered, but after some consideration an SNMP test suite with extensive ASN.1 BER encoding tests was selected.

The SNMPv1 test material attracted much attention, and other actors became aware of ASN.1 vulnerabilities and began to work on the subject. NISCC compiled a list of top ten ASN.1 protocols relevant in the CNI perspective. This served as guidance for further research, and prompted the PROTOS team to develop H.323 test material.

During the creation of the test suite a new vulnerability domain was discovered. Initially the PROTOS team studied Q.931 as a part of the H.323 research, and not until some studies on ISDN it was discovered that the protocols shared a common connection control protocol. Later it was discovered that ATM also shares a related control protocol - User to Network Interface.

Thus, the potential impact of the H.323 test suite containing Q.931 messages would be vastly larger than intended. This raised questions on the linkage of protocol specifications and prompted research on protocol dependency.



Slide 8: <meta name="ouspg-levels">

Looking back to the test materials, the concept of vulnerability meta levels emerges. Meta levels present a multilevel vulnerability taxonomy designed to categorise vulnerabilities based on their impacts. The basic idea of the taxonomy is that a vulnerability affecting a low-level concept should have a high meta level, and vice versa.

Traditional Infosec research often focuses on a single vulnerability in a single implementation. Faults, usually leading to vulnerabilities are found from one or more versions of a single product. These constitute meta level zero vulnerabilities.

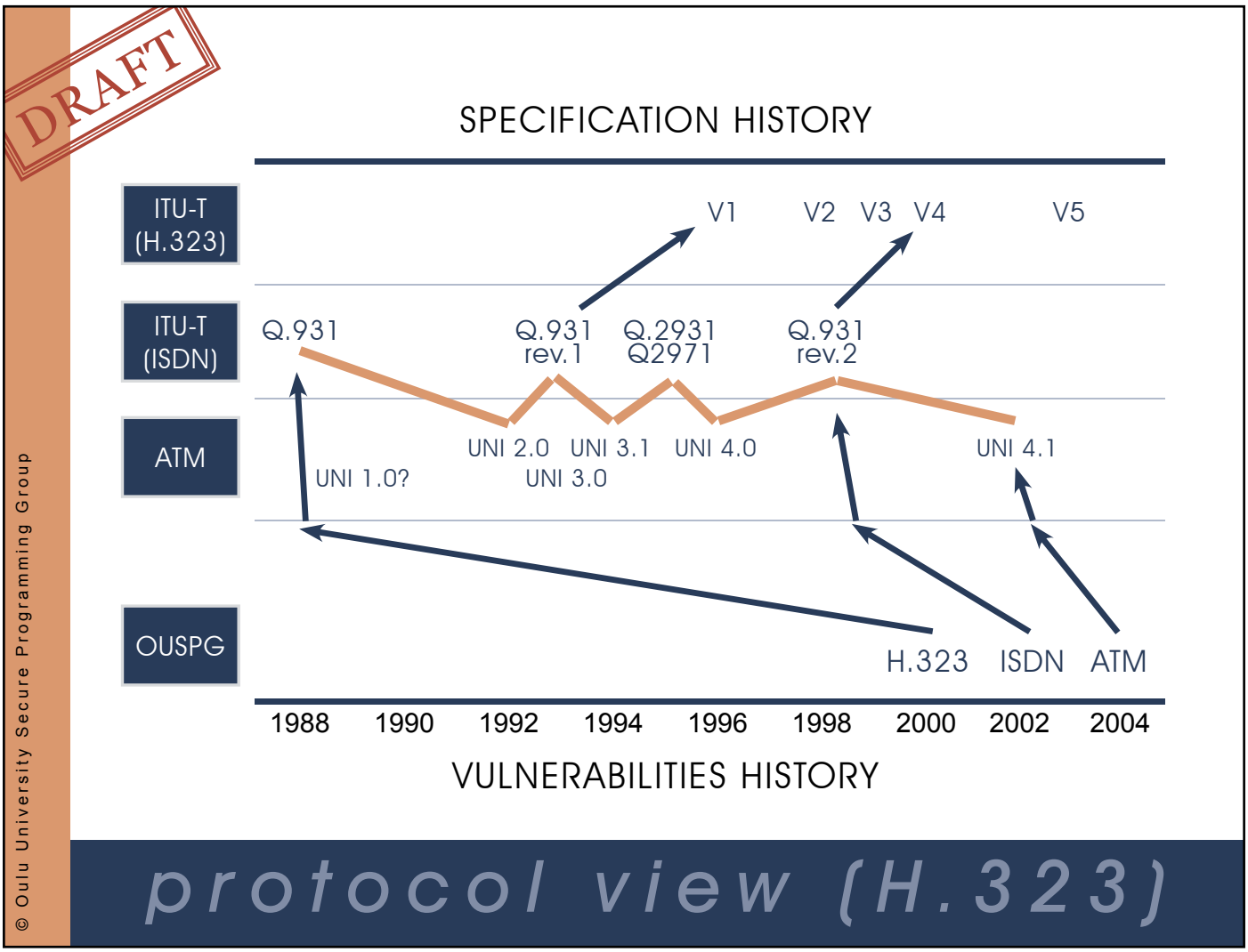
The PROTOS test suites have found common vulnerabilities between all implementations of a protocol. As these vulnerabilities have a greater impact, they are considered meta level one vulnerabilities.

Multiple protocols or protocol families can share a common subprotocol. If the shared protocol exhibits vulnerabilities, all the protocol families involved may have vulnerabilities. These shared vulnerabilities are what we perceive as meta level two vulnerabilities.

Protocols also share a number of (en)coding schemes, encryption schemes, notations and the like. A vulnerability in e.g. a coding scheme or in it's common implementation can have unforeseeable scope and consequences. These we call meta level three vulnerabilities.

H.323 is a meta level three vulnerability, which makes it's impact extremely hard to analyse. The vulnerability might be more widespread than expected at least because of:

- Q.931, a shared protocol with at least ISDN-D and ATM (Lvl 2)
- ASN.1, a shared scheme with numerous protocol families (Lvl 3)

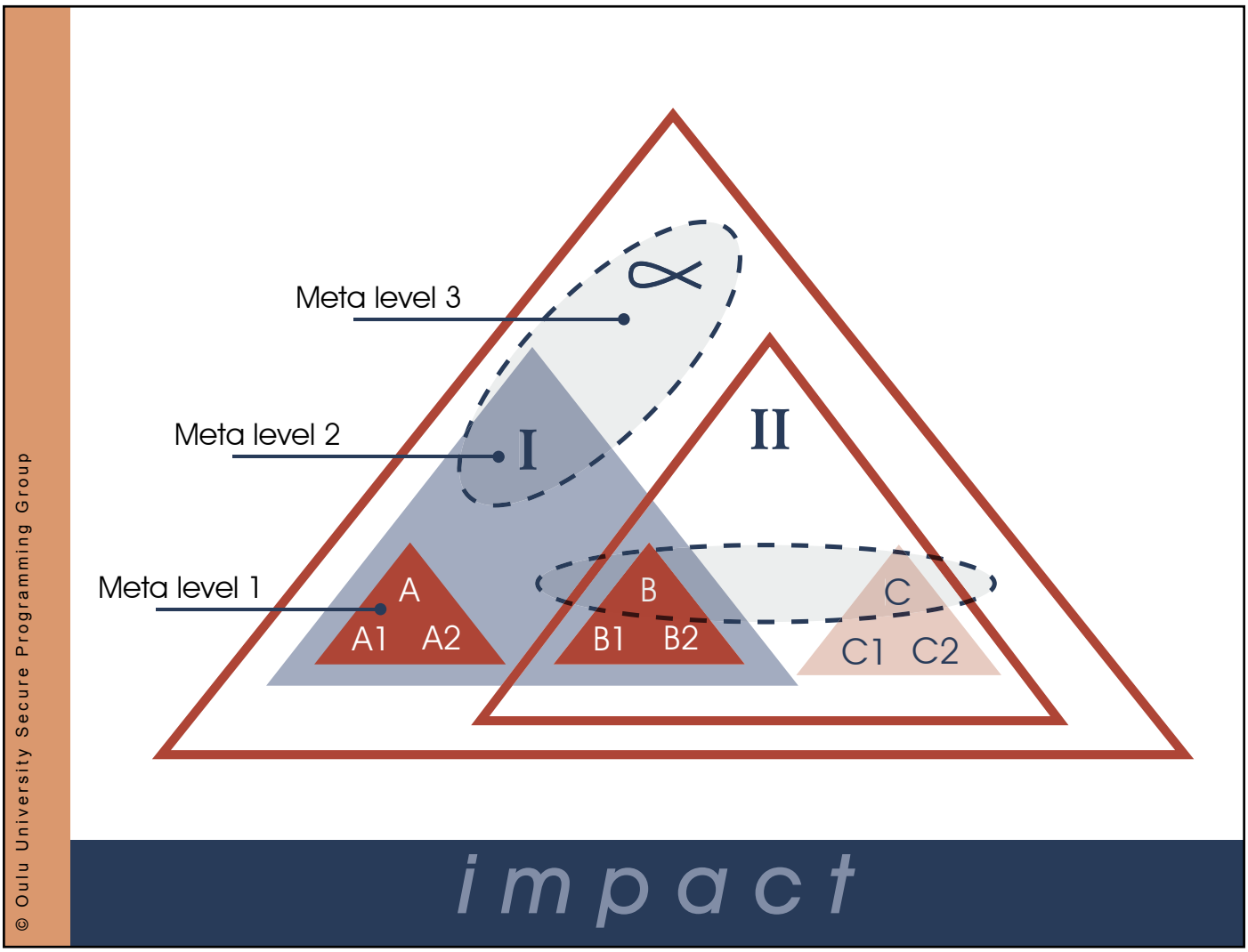


Slide 9: Protocol View (H.323)

While meta level three vulnerabilities seem hard to grasp by any analytic means, it was debated whether meta level two vulnerabilities could be apprehended by charting protocol dependency.

Post-mortem analysis on the vulnerabilities revealed by the H.323 test suite included a study on the history of Q.931 specifications. The results, shown on this slide, clearly show the forks taken in the development of the specification, and the resulting dependencies to other protocol families. Included is also the order of OUSPG findings leading that shed light on the dependencies.

This history study serves as a proof of concept for meta level two vulnerability research - had it been performed in the early stages of test material development, it would have been an invaluable asset in impact assessment and test subject selection.



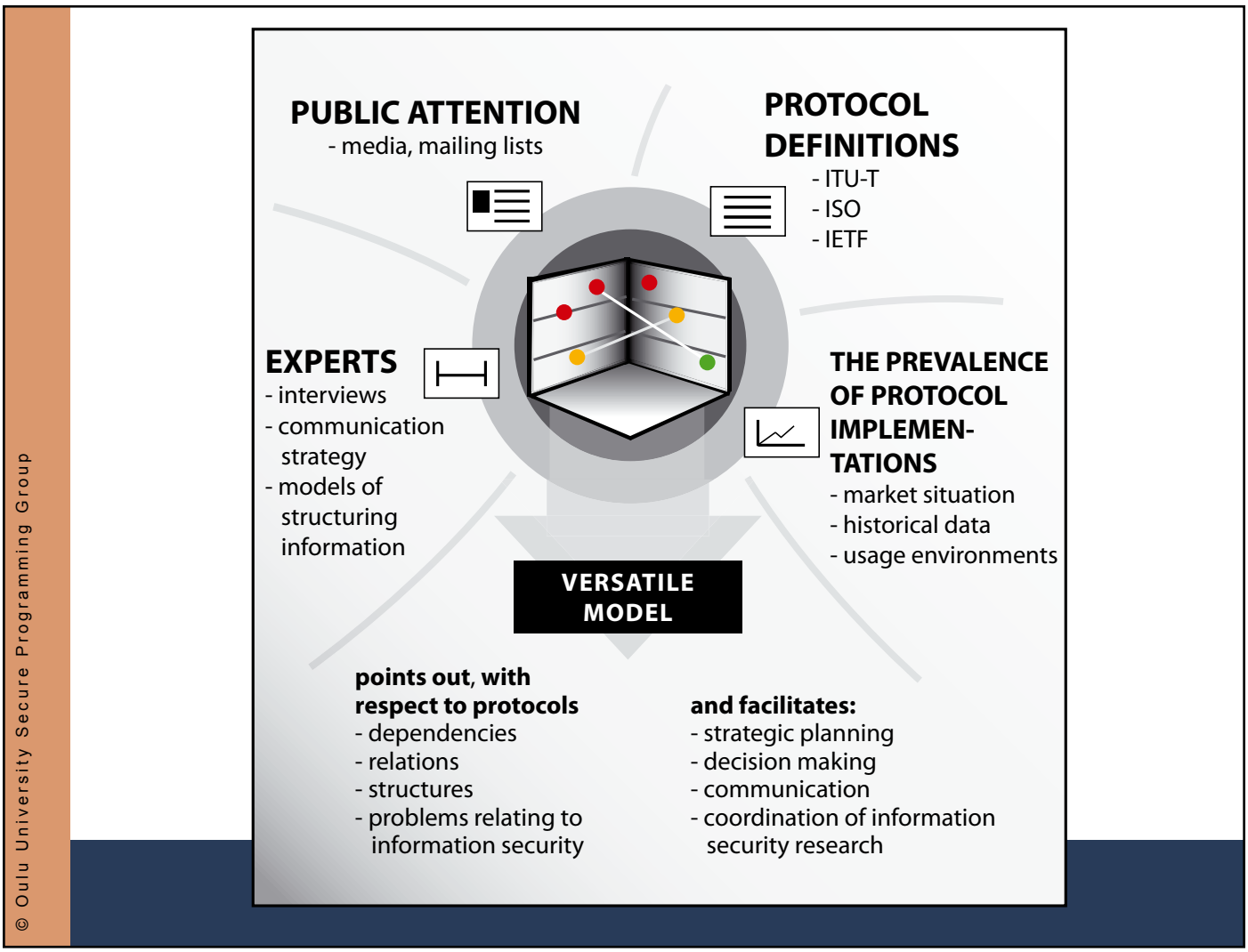
Slide 10: Impact

This is a theoretical protocol dependency example case involving a meta level three schema (alpha), two meta level two constructs (protocol families I and II), three meta level one concepts (protocols A, B and C) and six meta level zero structures (implementations A1...).

In most cases, the impact area expands geometrically by meta levels. However, there can be dependencies that are only shared between some entities of different meta levels. Represented by ellipses on the slide are two cases: a portion of the schema alpha that is only implemented in protocol family I and an add-on feature to protocols B and C that does not appear in other members of the protocol family II.

In conclusion, the impacts of protocol dependencies may be varied. Impact assessment calls for detailed analysis of the entities involved.

Common reasons for dependency include
- Standardisation process
- Code re-use, libraries, legacy code and other re-use
- Re-inventing the wheel, often results in making the same mistakes
- Making infinitesimal changes to an existing specification and publishing the result under a different name (more common that you'd think)



Slide 11: The Proposed Model

The scope of protocol dependency is wide, and its study would require advanced data gathering methods along with visualisation methods to present the results in easily grasped and compressed forms. Thus, the proposed methods for researching protocol dependency include:

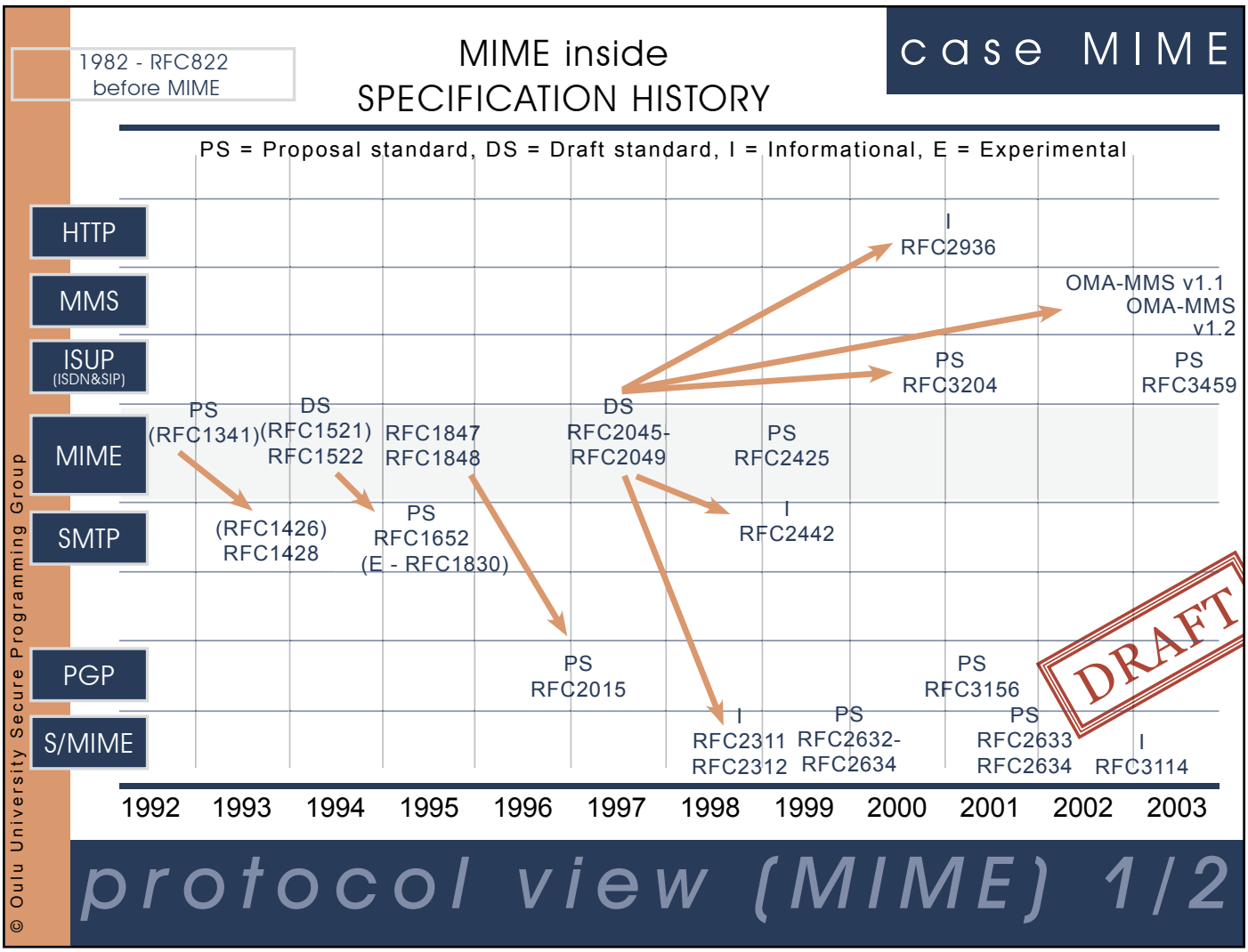
1. Summaries of relevant technical specifications, their relations to other specifications and historical dependencies

2. Surveys of the public attention on the security of different protocols and protocol implementations

3. Surveys on the prevalence of protocol implementations and their usage environments

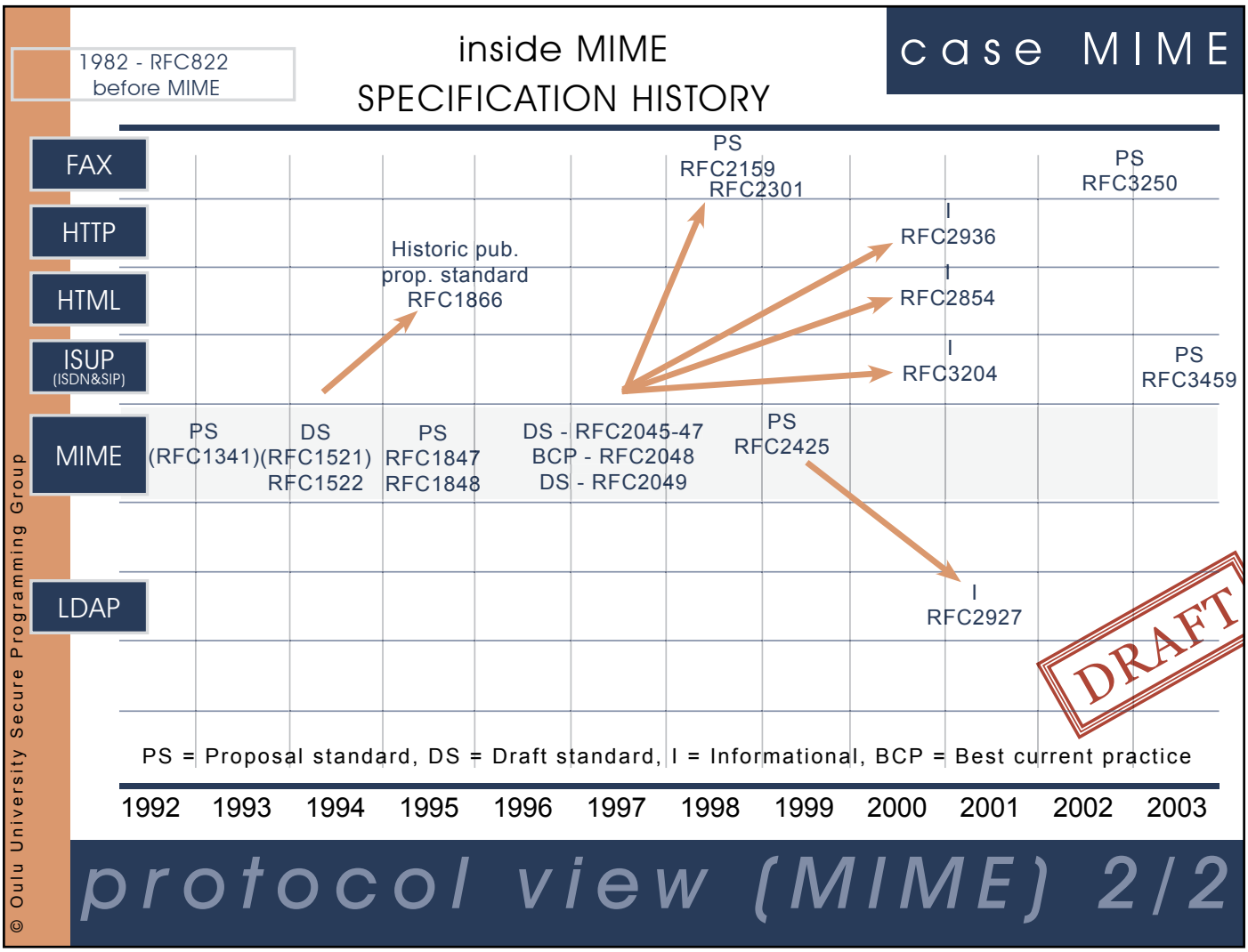
4. Expert interviews augmenting the plain data mining approach

The accumulated data is presented with visual models that bring up different aspects from the data related to protocol dependency and security. As the model is highly visual, it can additionally be used as a communication method between researchers and other actors.



Slide 12: Protocol View (MIME) 1/2

Charting what protocols incorporate MIME could be useful for understanding what should be tested with a MIME test-suite. How do we depend on MIME in sense that if vulnerabilities are found in one instance, then what else might be affected.

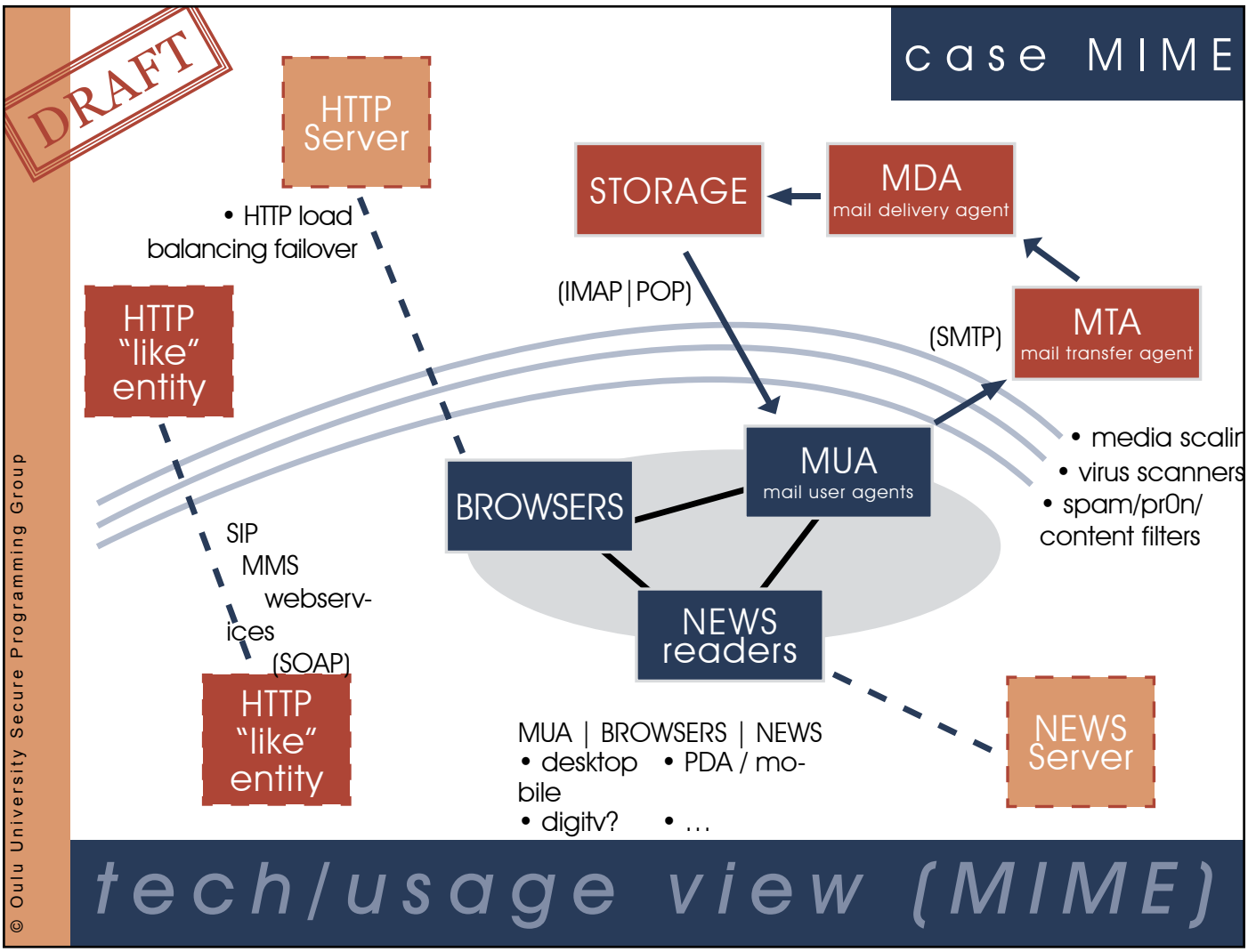


Slide 13: Protocol View (MIME) 2/2

Charting what MIME is used to carry could be useful for understanding where and for what MIME is used. Actual content types on top of MIME bearer may give organisations attempting to understand scope of potential MIME problems hints about the usage scenarios and services affected.

1st example:
LDAP's RFC 2927 MIME Directory Profile for LDAP Schema. This document defines how a MIME type may be used to transfer a single LDAPv3 schema definition. A schema for use with LDAPv3 consists of any number of object class, attribute type, matching rule and syntax definitions.

2nd example:
RFC 2159 A MIME Body Part for FAX. This document contains the definitions, originally contained in RFC 1494, on how to carry CCITT G3Fax in MIME, and how to translate it to its X.400 representation.



Slide 14: Tech/Usage View (MIME)

What types of products or equipment either receive, originate or intercept MIME messages? In other words, what parses mime? This could be expanded to specific lists of products and vendors by organisations whom it concerns.

The draft MIME slides presented here have been used in the co-ordination of actual MIME vulnerabilities.

Slide 15: The End

http://www.ee.oulu.fi/research/ouspg/