

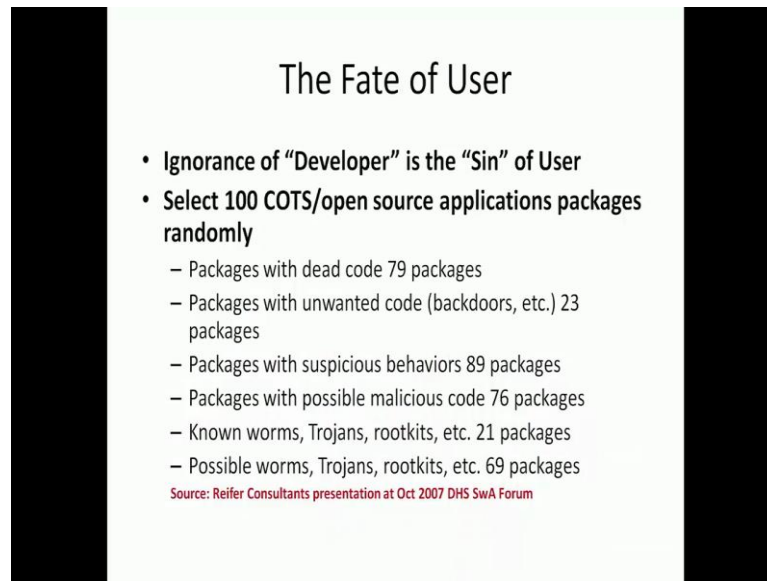
**Information Security - II**  
**Prof. V. Kamakoti**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture - 03**  
**Architectural Aid to secure systems Engineering**  
**Session – 2: Ignorance of “A” is Sin of “B”.**

So, we start with session-2 of the course. I titled this as Ignorance of A is Sin of B, A naught equal to B, right. So, you make application software like a compiler or you make see in normal life, right. Your sin, you are going to take the effect of your sin, but in Information Security, in the computing infrastructure if you generate a stupid operating system with lot of vulnerability, you are not the guy who is suffering. There will be somebody else who will suffer. So, if you are ignorant about say, security and you make systems and for some reason that system becomes popular as you see many systems today, then the sin that you are committing will be actually that the ignorance that you have will be realized like a sin for somebody else. They will start repeating the bad effects of that, right?

I am going to give you some case studies here, where some stupid decisions in the past and stupid, I am using that word voluntarily stupid software that as cover would in the market has now made life miserable for somebody else for which we are not responsible and we are paying to get those software. So, let us go and look at this.

(Refer Slide Time: 02:03).



## The Fate of User

- Ignorance of “Developer” is the “Sin” of User
- Select 100 COTS/open source applications packages randomly
  - Packages with dead code 79 packages
  - Packages with unwanted code (backdoors, etc.) 23 packages
  - Packages with suspicious behaviors 89 packages
  - Packages with possible malicious code 76 packages
  - Known worms, Trojans, rootkits, etc. 21 packages
  - Possible worms, Trojans, rootkits, etc. 69 packages

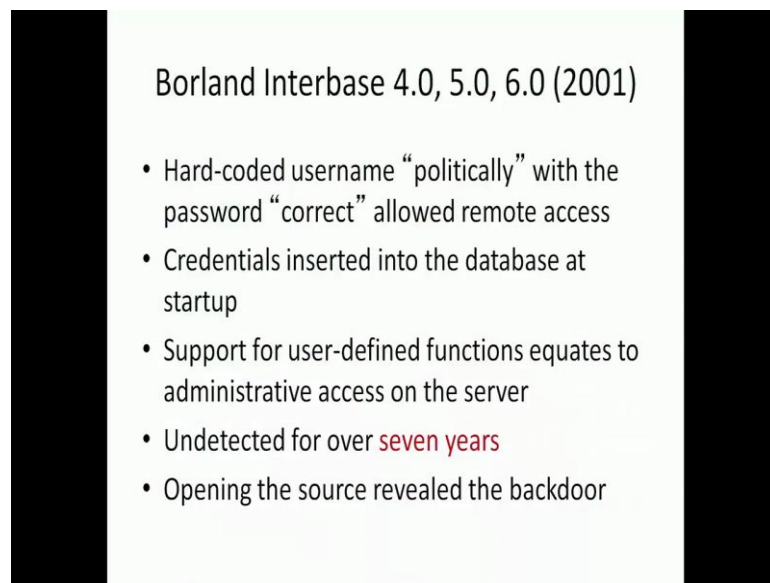
Source: Reifer Consultants presentation at Oct 2007 DHS SwA Forum

See this is the ignorance of the “Developer” which is the “Sin” of the user. So, this was one something happened in 2007, and I do not see something happening in the same way on recent times. May be it was I missed it, but this itself is very scary and I am sure more recent surveys would be much scarier if we could called that word, right? So, hundred commercially of the cells open source applications were picked up randomly and they were analyzed and the analysis is put down as those 6 points. There are packages with dead code 79 packages. Dead code is a code which you do not execute normally, but the dead code can be a potential atom bomb because, that can come into existence in some contingency situation, where it can go and kill or it can go and be a major vulnerability, right. Then, there were unwanted codes which can be interpreted as back doors. There are 23 packages; there are packages with suspicious behavior around 89 packages. 89 percent of your software has suspicious behaviors. There were malicious code, known malicious code in 76 packages known worms, Trojans rootkits in 21 packages and possible worms, Trojans and rootkits in around 69 packages. So, 90 percent of your software were not trustable and this from some randomly picked hundred commercially of the shelf slash open source applications. This is where you are now heading and all the software you are executing and all the development environments you are having or taken from the set. So, this is the faith of the user with respect to security and there you

are now; going to expose your heart, your kidney, your blood pressure, everything on systems running with this type of software. That is this scare, right.

I think this is the best open problem for any researcher to handle, right and that is why we say this course is very important.

(Refer Slide Time: 04:23)



Borland Interbase 4.0, 5.0, 6.0 (2001)

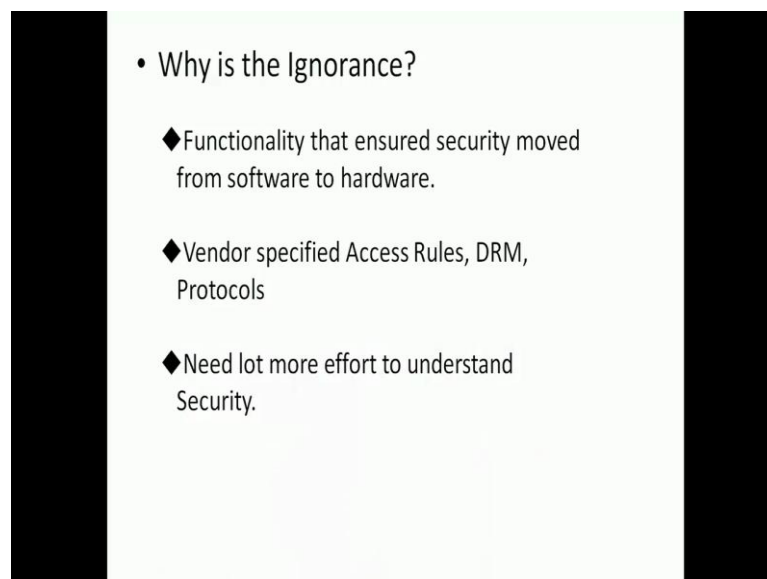
- Hard-coded username “politically” with the password “correct” allowed remote access
- Credentials inserted into the database at startup
- Support for user-defined functions equates to administrative access on the server
- Undetected for over **seven years**
- Opening the source revealed the backdoor

Now, this is very interesting case study that we get from the web. There was a popular database, there were 3 releases post 2001 and after 7 years, somebody casually went through the code and they found certain words like politically and they went and started analyzing this code. They found that if you enter the name as user name as politically and you give the password as correct, you get super user permission into the database. You mean people who worked on database would understand what it means to have a super user permission can entirely go and kill. Suppose your bag in which you are put your money uses this database, some politically correct guy can actually come and completely transfer the money from your account to his account. You can also make loans in your name and transfer that money also there. So, it is not just lose your money that you become accountable for money which you did not possess or you did not take. So, the amount of (Refer Time: 05:30) that could happen if something like existing in this database. Today you are using some end databases in for different applications at

different sectors, right, financial sector, you know strategic sectors. So, many things are there and lot of things use commercial of the shelf. Are you sure that such a thing does not exist there?

So, every day morning; pray God, that your money should be safe. You understand what I mean, right. So, this is very important and this is something disgruntled. User would have put something there and disgruntled developer and when he checked out of the company, he can come out and create ravage about that company and you would also concealed may say in such type of logins do not even log, put it in the log. If I do this type of access, do not even log these accesses, you can write that and if you are looking at a typical database scored up several millions of lines, this 10 or 20 lines or 50 lines that you need to do this and that you compile it, right, it can go as a stealth. Nobody can even see this, right. So, this is the state of affairs.

(Refer Slide Time: 06:42).



Now, why is this ignorance? I have already explained in the previous thing, but I am going to go in a depth. The ignorance was that there were several layers. There is operating system layer, micro architecture compiler application software and I said all these layers were isolated and this isolation was one of the reasons for this ignorance. Now, I can go much deeper into this and now, we say that today what has happened is

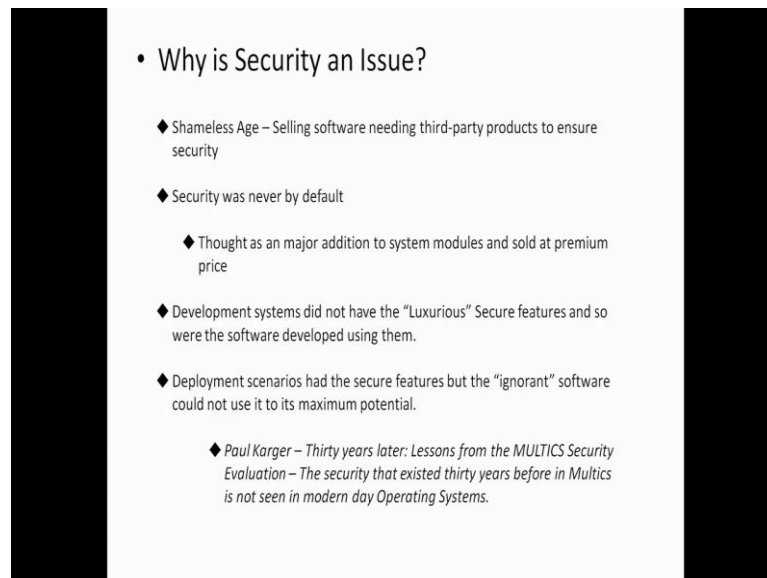
functionality as actually moved into hardware. So, yesterday, day before yesterday software become yesterday's co-processor and becomes today's hardware, a typical example. Today is the graphics, in olden days the CPU was doing the graphics and that is why when I want to draw a line, it goes like this, right. Then, this was shifted into, it was basically software. What did they do? They got graphics co-processor which was doing this functionality. So, I want to do graphics as CPU. I will tell the co-processor does it and today what happens is that graphics co-processor actually has become part of your chip. That is why you call it as a system on chip, right? So, day before yesterday's software became yesterday's co-processor and it has become today's hardware.

Now, when I had that as software, I knew what it was trying to do today. It has become hardware. It is not even a code. There is no way by which I can go and find out what it is trying to do, correct. So, if there is going to be a Trojan or a malware inside that hardware, there is no way by which I could go and find it out today. So, functionality that ensures security actually moved from software to hardware. We will see some of them as we proceed here and then, there are lot of vendor specified access rules and digital management protocols. We really do not know what it means. Today somewhere you get on the screen. Now, there is some problem with a software, should I go and report it. You say yes report, right. Visit going to put just what you have done, what has happened with the software or it is also going to take some very confidential information. One directly can zip and also send it. Are you going to monitor it, right? There is some problem, should I report? You go and click that report key. It is just sending that report or it is also sending some other confidential information along with it, you do not know. It can send your shadow file of your password, right. It can. What is the guarantee it is not doing it? We do not have a guarantee, correct.

So, lots of times you say no save the password. You are saving the password in your systems or every time you go and say www dot some mail dot com and then, you actually get logged in. So, this password is stored in some place within your system. In which form is it stored? May be stored with something, but is there a guarantee that somebody is not going to take away that password because your system is always there on the internet. So, there are certain access rules and certain vendor specifies this access rules, and he makes an assumption because the vendor also works in one of those layers.

He is probably working in that operating system layer or the application software development layer. He does not know what the hardware is. There is some level of ignorance about the hardware and the execution environment. So, he assumes certain environments and say in this environment, your confidential information is really going to be stood confidential, but when it goes to some other environment where some of his assumptions you know violated, he is in no position to assure you that confidentiality. Are you able to understand what I am saying? So, there are certain vendors specified access rules, but when they are imposed into some other scenario, they make certain assumptions about hardware and those hardware things are violated. You basically start clicking information. So, there is lot more effort needed to understand security, right. So, this is where we start.

(Refer Slide Time: 11:04).



- Why is Security an Issue?
  - ◆ Shameless Age – Selling software needing third-party products to ensure security
  - ◆ Security was never by default
    - ◆ Thought as an major addition to system modules and sold at premium price
  - ◆ Development systems did not have the “Luxurious” Secure features and so were the software developed using them.
  - ◆ Deployment scenarios had the secure features but the “ignorant” software could not use it to its maximum potential.
    - ◆ Paul Karger – *Thirty years later: Lessons from the MULTICS Security Evaluation – The security that existed thirty years before in Multics is not seen in modern day Operating Systems.*

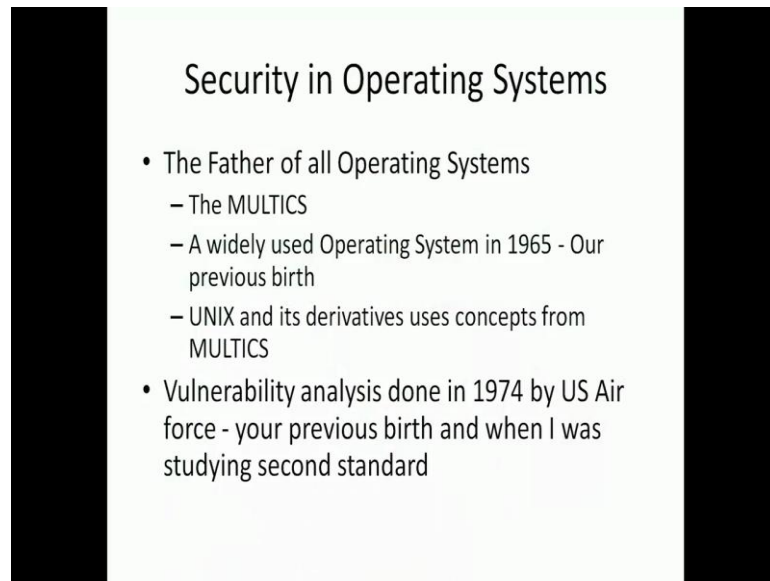
Now, why is this has become an issue? We are in a shameless age. There are people selling software and they say you need to buy third party products to ensure security. I buy some operating system. Along with an operating system order, I have to send one anti-virus order also, correct. This is a shameless age, so security was not ever in my agenda when I was developing software. Security was never by default. People talk the security is a major addition to system modules and it was actually sold at premium price, right if I want to. So, that is much going to be much costlier than an unsecure system.

People did not think that security is an inherent blood of your body. They said some extraneous thing which you need to add if you want to make your body glow much better.

So, when I start developing software, I do not give a secure environment there. So, you start developing the software in whatever in a non-secure environment. You have the basic things to develop software, so obviously you are not even. So, when you compile the software, it is not a secure compilation. It is an ordinary recompilation because secure compiler causes much more than an ordinary compiler. So, you start compiling it and when you go to a customer site, there every security is there. You put software that is developed in a non-secure environment that has been compiled in a non-secure environment. You go and put it in a secured environment. It never uses those security features. Those are all some you know in Tamilnadu, we have this Navaratri dolls, right. Just keep it in that thing and then, keep it and do some Arathi, right. So, they are never used, right. They are just dolls, right. So, because you have developed everything in non-secure environment, your compilation has been in secured. Now, you can go and put it in a secured environment, nothing comes. So, although securities features are just mute watched, the point is deployment scenarios had the secure features, but the ignorance software which was developed in a non-secured environment could not use it. It has maximum potential because when you develop software, you do not want to give an environment which is costly, which has the security features, so that every time I compile, I subject it to a secured compilation. I subject it to unsecured execution that does not happen and this was one major mistake.

So, this is not, what I am not taking off. Please go and read the paper that you see at the end of this slide, Paul Karger, unfortunately he is not alive, but he wrote a very interesting paper 30 years later, Lessons from the MULTICS Security Evaluation, so the output that take away from this paper is that the security that existed that 30 years before in MULTICS is not seen in modern day operating systems, and that was the take away from this paper. Now, I will go and talk about this paper in some detailed during this course.

(Refer Slide Time: 14:16).



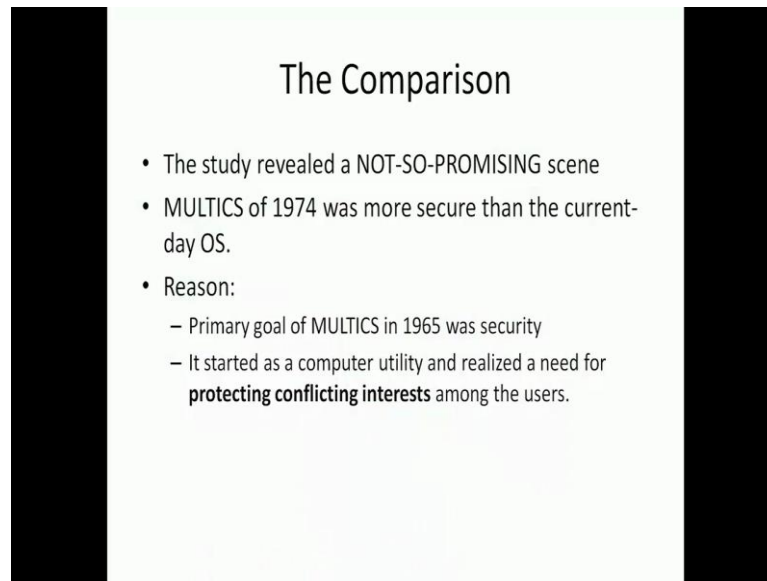
## Security in Operating Systems

- The Father of all Operating Systems
  - The MULTICS
  - A widely used Operating System in 1965 - Our previous birth
  - UNIX and its derivatives uses concepts from MULTICS
- Vulnerability analysis done in 1974 by US Air force - your previous birth and when I was studying second standard

Let us talk about this. I will just briefly tell about that paper that paper is very beautifully written. It's a 10 page paper. I suggest all of you can go and search the Google and you can get it. Google it and you will get it, but I will talk about some important points in that paper. MULTICS was one of; we could call it as father of all operating systems. It was a very widely used operating system in 1965 or something. None of us were born. At least I was not born. UNIX and its derivatives use concepts from MULTICS. That is why we say it is a father of many operating systems. In 1974 probably many of you would not have born and I was studying 2nd standard, right. At that point the US Air Force wanted to do a vulnerability analysis on this software environment, especially many of them were running MULTICS. Please note that there was no internet. There was some minimalistic communication between devices, right. At that point of time, they even thought that there is a need for vulnerability analysis. This is a very big you know vision I should say.



(Refer Slide Time: 15:36).

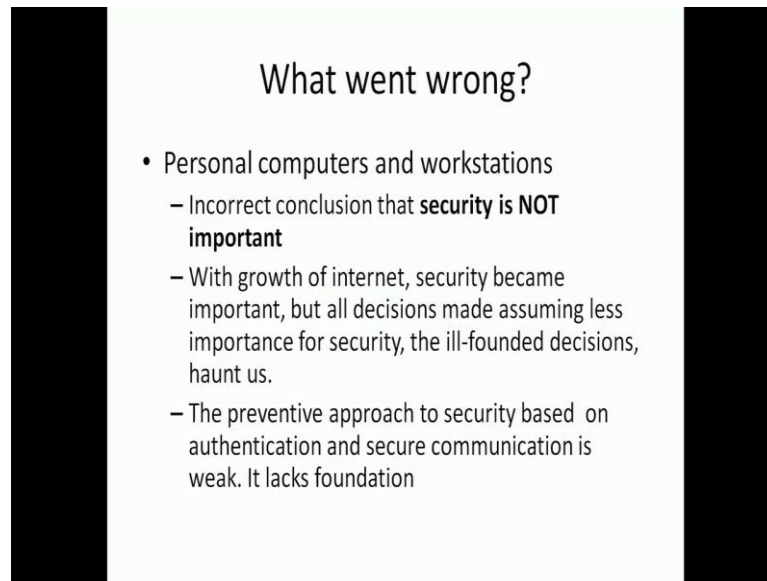


### The Comparison

- The study revealed a NOT-SO-PROMISING scene
- MULTICS of 1974 was more secure than the current-day OS.
- Reason:
  - Primary goal of MULTICS in 1965 was security
  - It started as a computer utility and realized a need for **protecting conflicting interests** among the users.

Even that time where even tens of or maybe hundreds of computers are connected together, it is not that internet that. The moment I add my cell phone on, switch on my cell phone, I am on internet with one billion users or whatever. So, you are talking of such type of things, but even that time they did study and they found it was not so promising, The MULTICS of 1974. Today was more secure than the current day or even that day they said it was not so promising, but what you see now, the MULTICS of 1974 was much more secured than the current day operating system. This is what the paper says. I am not saying that and the reason for that was the primary goal of MULTICS. In 1965 was security, it started as actually a computer utility and realized a need for protecting conflicting interest among the users. For example, users even they were trying to get more CPU time, right. So, security started as a habit. When a baby was born from day one, it was tried to put into some secured habit there, right.

(Refer Slide Time: 16:52)



The slide has a light green background and is flanked by two vertical black bars. The title 'What went wrong?' is centered at the top. Below it is a bulleted list with three main items, each followed by a sub-point.

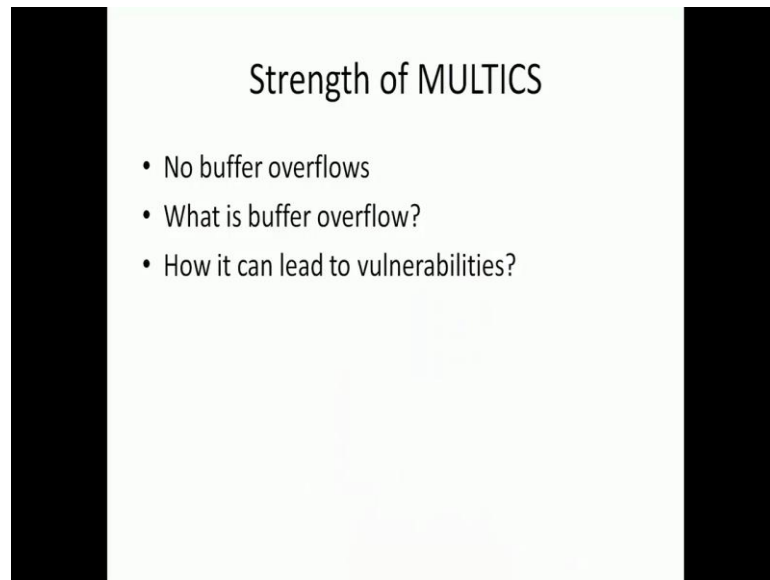
- Personal computers and workstations
  - Incorrect conclusion that **security is NOT important**
  - With growth of internet, security became important, but all decisions made assuming less importance for security, the ill-founded decisions, haunt us.
  - The preventive approach to security based on authentication and secure communication is weak. It lacks foundation

So, what happened later when personal computers and work stations started coming up, people never thought they were going to be networked in such a massive manner. So, people thought there was incorrect conclusion. The security is not important. I am only going to use, so why should I bother. Why should I buy all these things, but with growth of internet, security became very important, but all decisions made assuming less importance for security. Those ill-founded decisions are haunting us. You are getting a point. So, there has been an era of development, era of development of complex systems with security put in the back burner and those ill decisions.

Now, those same systems are now put into the internet and we are trying to retro fit security, the preventive approach to security based on authentication and secured communication is very weak, right. Today I say if you want to login, you use the password. You use this, but there is something much beyond that we are feeling just by introducing one wrapper around it and say if you want to come here and you should use this wrapper is not working out because the wrapper itself is executing on a secured platform, insecure platform where security was not a habit. So, this ignorance penetrates into that wrapper, right. So, just by saying I will have good authentication and secured communication alone is not going to solve much of the problem. I could have all those

things at the wrapper in, but the inherent weakness within the system can still expose your system to vulnerabilities and that is the big take away from this paper.

(Refer Slide Time: 18:34).



So, the strength of MULTICS is, there are no something called buffer overflows. We will talk about what is buffer overflow in the next session. What is buffer overflow, how it can lead to vulnerabilities, I will give it as one case study, right which we will deal in the next session 3.s