

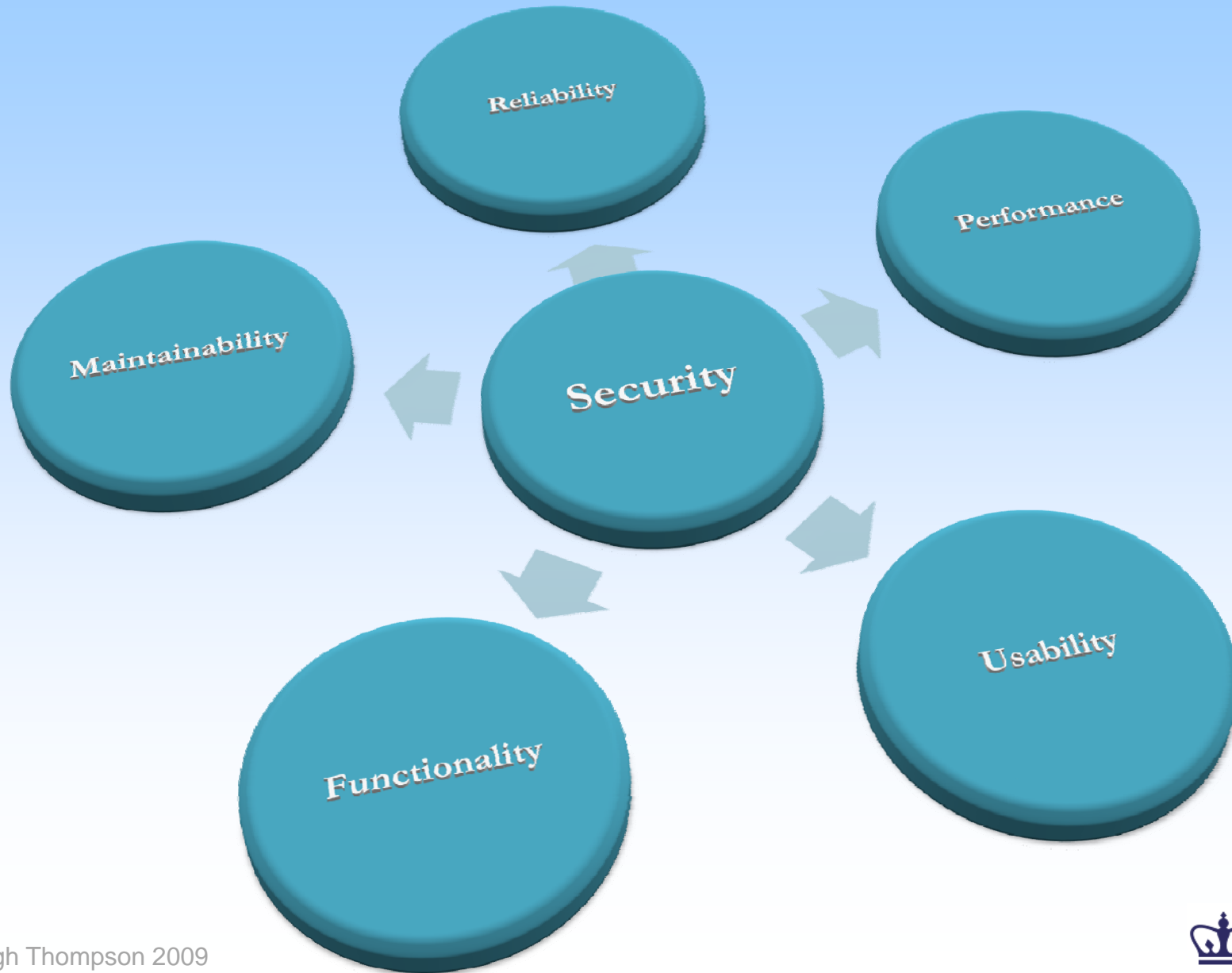
COMS E6998-9:
Software Security and
Exploitation
Lecture 3: Design

Hugh Thompson, Ph.D.
hthompson@cs.columbia.edu

Design Principles

5. **Beware of the Economics of Security** – Need to consider the economics of attackers during design.
6. **Fail secure** – When software fails it need to consider security.
7. **Patchability** – Software need to be designed with update in mind.
8. **Determine security tradeoffs: Usability, Performance, Maintenance** – Security often conflicts directly with usability, performance, and maintainability. We need to strike the right balance.

Warning: Tradeoffs



Design Principles

9. **Auditing and logging** – Logging helps to battle insider threats and is key for compliance.
10. **Choose components based on functionality *and* security** – Applications are collections of individual components. Choices of components need to factor in security.

Design Principles

- 11. Don't make security assumptions about other people's code** – Operating system libraries, 3rd party code, and modules developed internally don't often do what you assume.
- 12. Identify behavioral impact of security controls** – Adding security controls has to be tempered by how users will react to them.

Design Principles

- 13. Compliance: Attackers may attack you, auditors will show up** – Designing “secure” software is as much about compliance requirements as it is about actual security.
- 14. Threat Analysis and Modeling** – Threat modeling is the process of thinking like attackers and can help identify risks and prioritize application security efforts.
- 15. Tunable security levels** – Software needs to accommodate a range of users with varying security needs.
- 16. Don't assume data integrity** – We need to ensure that when an application trusts data or code that the trust is warranted.

Design Principles

17. **Least exposure** – Where possible, designers need to minimize the attack surface of their software.
18. **Secure by default** – Most users accept defaults; we need to ensure that software behaves securely out of the box.
19. **Don't reinvent the wheel** – Creating your own encryption algorithm or even your own implementation of complicated security code is often a bad idea.
20. **Beware legacy code and backward compatibility** – We need to ensure that backward compatibility balances with security and doesn't weaken the system.

Design Principles

21. **Secure the weakest link** – Software is only as secure as its weakest link.
22. **Secure all access routes** – We need to consider all access routes to sensitive data or functionality.
23. **Security through obscurity doesn't work** – We need to assume that attackers have full knowledge of the code and design.
24. **Beware of shared resources** – Memory, disk space, and network bandwidth are usually shared resources. We need to make sure an attacker can't control them to compromise security.

Design Principles

25. Learn from mistakes – There is no better security lesson than to thoughtfully go through the vulnerabilities reported in similar projects or designs.

Input Validation



News Focuses/statments_full.asp?statID=105

12 August 2007

[Statements home](#) | [Full text](#)

Spokesperson for Secretary-General
[Latest Statements](#)
[Briefing Highlights](#)
[Briefing Transcripts](#)

Press Releases and Meetings Coverage

News Conferences

• [Press encounters](#) by Secretary-General
• [Other Press Briefings](#)
[Search](#) | [Video](#)

What, When at UN

[New York](#)
[Geneva](#)
[Calendar of Events](#)

E-mail News Alerts

[Subscribe here](#) **Free**

Multimedia

[UN Radio](#)
[Webcast](#)
[TV/Video](#)
• [UN in Action](#)
• [21st Century](#)
[Photos](#)
[Media Accreditation](#)

Resources

[RSS version](#) **XML**
[UN Daily News](#)
[News Resources](#)
[News Focus](#)
[Documents](#)
[UN System Links](#)
[UN Envoys](#)
[Fact Sheets](#)
[Maps](#)
[Databases](#)

Secretary-General Ban Ki-moon
UN Headquarters
31 July 2007

Address to General Assembly thematic debate: "Climate Change as a Global Challenge"

Madam President, Excellencies, Dear Delegates,

Let me thank the President of the General Assembly, Her Excellency Haya Rashed Al Khalifa, for convening this timely and topical debate.

We meet at a time when climate change – long on the international agenda – is finally receiving the very highest attention that it merits. We have all heard a lot about the findings of the Intergovernmental Panel on Climate Change. They have unequivocally affirmed the warming of our climate system, and linked it directly to human activity.

The effects of these changes are already grave, and they are growing. The Arctic is warming twice as fast as the global average. The resultant melting threatens the region's people and ecosystems, but it also imperils low-lying islands and coastal cities half a world away. On the other hand, as glaciers retreat, water supplies are being put at risk. And for one-third of the world's population living in dry lands, especially those in Africa, changing weather patterns threaten to exacerbate desertification, drought and food insecurity.

We cannot go on this way for long. We cannot continue with business as usual. The time has come for decisive action on a global scale.

Excellencies,

I am convinced that this challenge, and what we do about it, will define us, our era, and ultimately, our global legacy. It is time for new thinking. We all need to shoulder this responsibility, not just for ourselves, but for our children and their children. Will succeeding generations have to ask why we failed to do the right thing, and left them to suffer the consequences?

- Print this article
- Email this article
- News story
- Video
- Radio report

http://www.un.org/apps/news/infocus/speeches/statments_full.asp?statID=105' - Microsoft Internet Explorer

File Edit View Favorites Tools Help



Address http://www.un.org/apps/news/infocus/speeches/statments_full.asp?statID=105'

ADODB.Recordset.1 error '80004005'
SQLState: 37000
Native Error Code: 8180
SQLState: 37000
Native Error Code: 105
[MERANT][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark before the character string ".
[MERANT][ODBC SQL Server Driver][SQL Server]Statement(s) could not be prepared.

/apps/news/infocus/speeches/statments_full.asp, line 26



Done

Internet



News Focus

Spokesperson for Secretary-General

[Latest Statements](#)
[Briefing Highlights](#)

Press Releases and Meetings Coverage

News Conferences
• [Press encounters](#) by Secretary-General
• [Other Press Briefings](#)
[Search](#) | [Video](#)

What, When at UN

[New York](#)
[Geneva](#)
[Calendar of Events](#)

E-mail News Alerts

[Subscribe here](#) **Free**

Multimedia

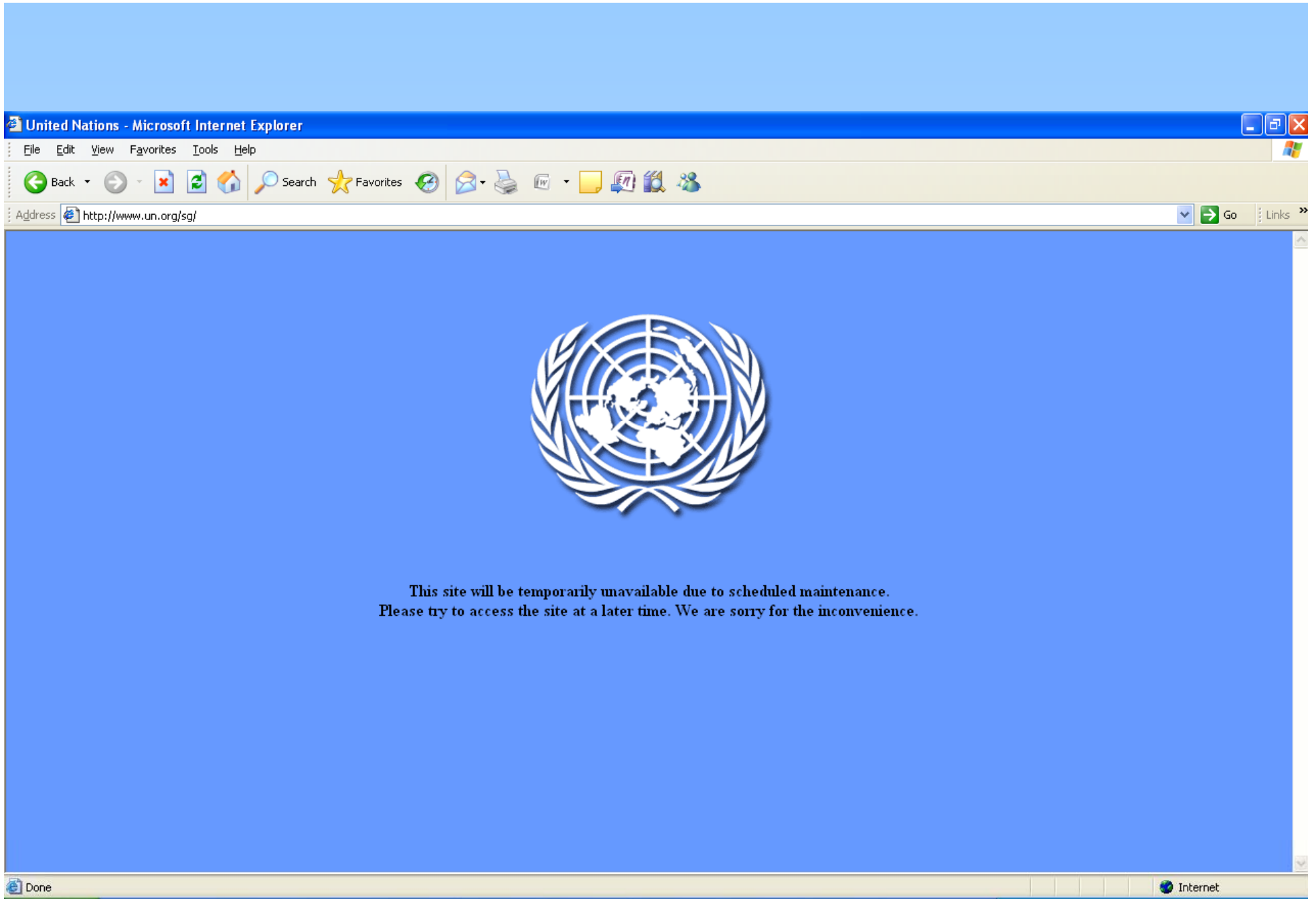
[Statements home](#) | [Full text](#)

Secretary-General Ban Ki-moon
UN Headquarters
31 July 2007

HACKED BY KEREM125 M0STED AND GSY THAT IS CYBERPROTEST HEY ÝSRAIL AND USA DONT KILL CHILDREN AND OTHER PEOPLE PEACE FOR EVER NO WAR

HACKED BY KEREM125 M0STED AND GSY THAT IS CYBERPROTEST HEY ÝSRAIL AND USA DONT KILL CHILDREN AND OTHER PEOPLE PEACE FOR EVER NO WAR

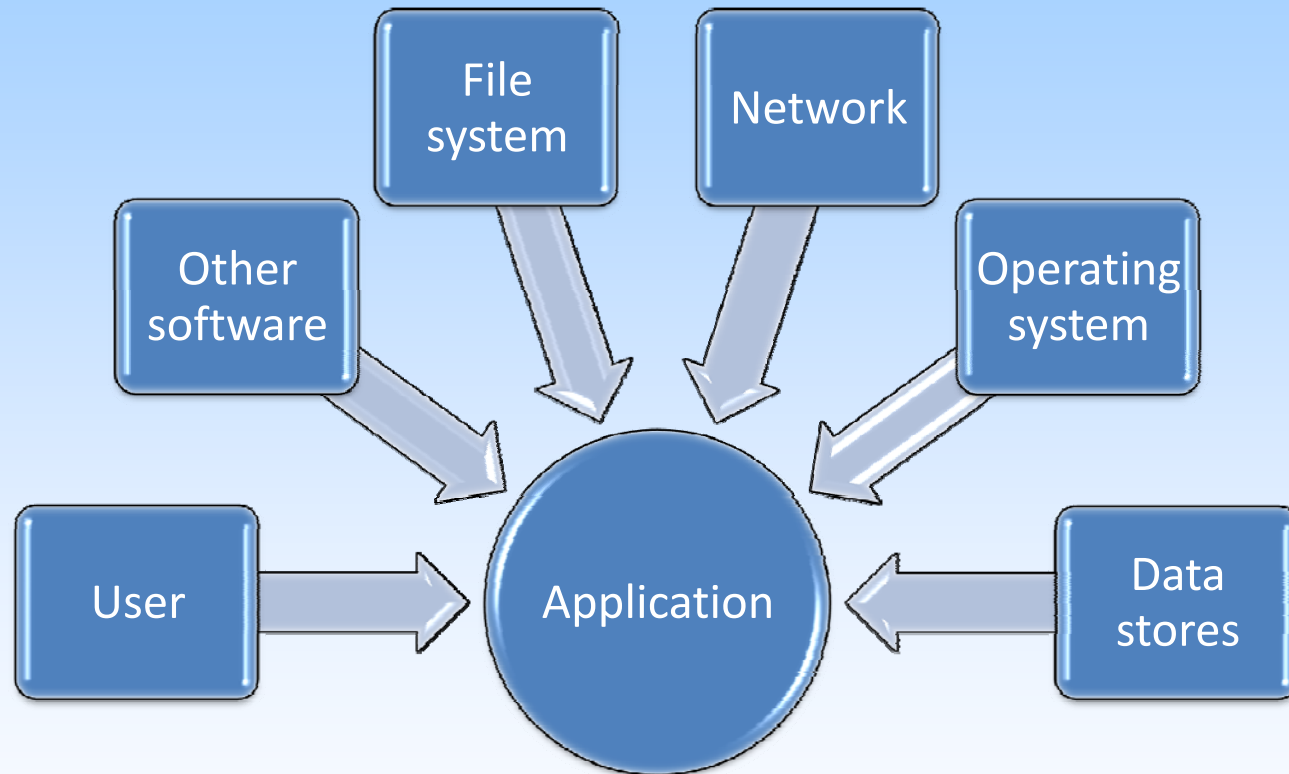
[Print this article](#)
 [Email this article](#)
 [Hacked By kerem125 M0sted and Gsy That is CyberProtest Hey Ýsrail and Usa dont kill children and other people Peace for ever No war](#) > [News story](#)



Beware the dreaded INPUT

- Many of the most severe vulnerabilities in software come from a failure to properly constrain input:
 - Buffer overflows, SQL Injection, format string, vulnerabilities, cross site scripting...
- As developers, we need to:
 - Identify assumptions we are making about input
 - Understand what input can be harmful
 - Constrain input appropriately

Think broadly about input...



Solution Strategies

- Regular expressions
 - Allow you to define *what is correct* as opposed to *what is incorrect* (especially important given canonicalization)
- Set input “gates” to filter/constrain input and then define a trust boundary
 - This helps you to maintain performance by assuming that data has been sanitized
 - Ensure that gates consider special characters, canonicalization, and 2nd order attacks

Public enemy number 1 (...sort of)

Vulnerability	% of all vulnerabilities
Cross-site scripting (aka XSS)	13.80%
Buffer overflows	12.60%
SQL injection	9.30%
PHP remote file inclusion	5.70%
Directory traversal	4.70%
Information leakage	3.40%
Denial of Service by malformed input	2.80%
Symbolic link following	1.80%
Format string vulnerability	1.70%
Cryptographic error	1.50%

Buffer Overflows: Dangerous Functions

- Buffer overflows are the number one vulnerability reported in C/C++ software (by a significant margin!)
- They occur when we make assumptions about input length but don't enforce them
- Top offenders (these don't consider destination buffer size):
 - strcpy(), strcat(), sprintf(), gets(), scanf(), sscanf(), fscanf()

```

int main(int argc, char* argv[]){
//some code
FILE* fp = fopen(fileName, "rb");
int len;
//file checking code
HexIt(fp);
//other code
}

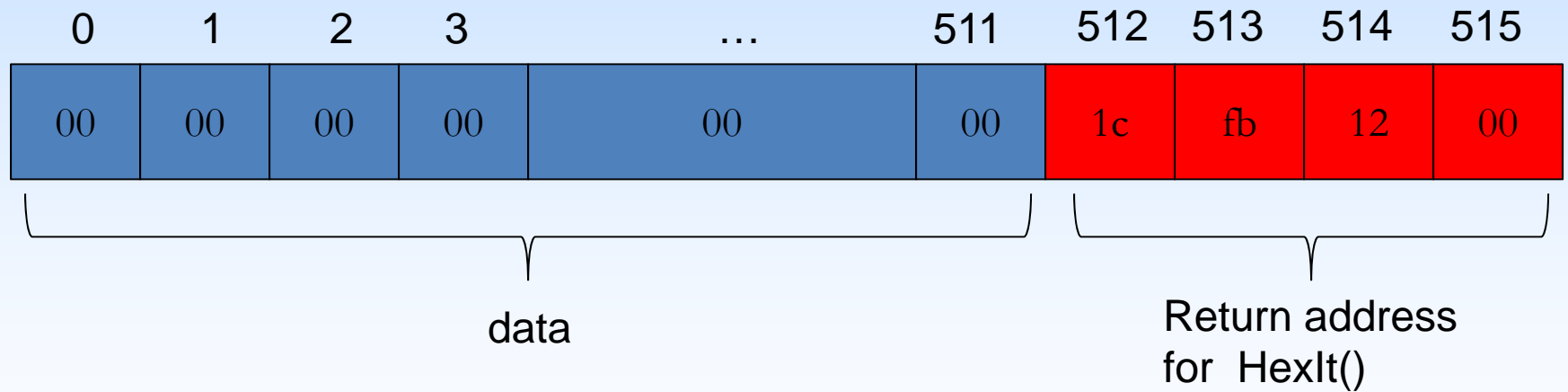
Void HexIt(FILE* file)
{
    int len;
    unsigned char data[512]; //Buffer to hold data
    FILE* fp = file;
    fseek(fp, 0, SEEK_END);
    len = ftell(fp);
    fseek(fp, 0, SEEK_SET);
    fread(data, 1, len, fp); //reads the file data
    fclose(fp);
    PrintHex(data, len); //function to output the data
}

```

The Stack

```
void HexIt(FILE* file)
{
    unsigned char data[512];
}
```

data[n]



BUG OF ZEN