Network Security

AA 2015/2016

Foundations of Computer and Network Security

Dr. Luca Allodi

Computer security

The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, the availability and confidentiality of information systems resources.

(NIST Computer Security Handbook)

First concept: preserving

- Wait, wasn't computer security about
  - "creating security"
  - "cracking security"
  - "hacking security"?
- No. Rather, it is about preserving security
- What does this mean?
  - You can only *preserve* something that you already have
  - Security technologies do not build security properties
    - They "merely" make sure that the security properties that are already there are maintained

What to preserve

- We need to understand two concepts
  - What is it there to be preserved
  - What is that "built-in" security that need be preserved
- Question time:
  - What is it that computers do?
    - They efficiently operate over information
- Everything that operates and is operated by a computer is information
  - A computer program (e.g. an OS, a videogame, a web service)
  - Temporary functions, variables
  - An excel file
  - A picture file
  - …
- All a computer system is about is *information*

Preserving what? Properties of information.

- At the bare minimum, any "piece of information" is only useful if
  - It can be reached
  - It can be read
  - It is correct
- These can be seen as "properties" of information
- Computer security is about preserving these properties
  - **Availability** → Assure that a piece of information can be reached when needed
  - **Confidentiality** → Assure that a piece of info can be read by those who can read it
  - **Integrity** → Assure that a piece of info communicates the right thing/concept (i.e. it is correct)
- **This is known as the CIA triad**

CIA as properties

- Confidentiality, integrity and availability are the "core" security properties of a piece of information
- On top of this, one can build additional properties
  - Accountability
    - The ability to know with certainty who/what operated on a piece of information
  - Non-repudiation
    - The entity that acted on the information can not "repudiate" his/her/its action
  - Authenticity
    - The piece of information has not been tampered with by anybody/anything
  - …

Who can act upon information?

- The previous slides used terms such as
  - Who/what
  - Entity
  - His/her/its
  - Anybody/anything
- Humans are not the only "users" of information
  - The human user of the system **avertedly** or **un-avertedly** modifies information
  - The automated user can modify the information
    - Avertedly?
    - Un-avertedly?

Personification

- A system/software/module/thread can act on the whole system (or another system) as if it were a human user
- "Personification" is the mechanism by which, for example, software threads are spawned
  - E.g. with the privileges of the entity that spawned it
- While a human user may or may not know what he/she does..
- ..A "personified" thread does not
  - There is no notion of "avertedly" or "un-avertedly"

The core problem of computer security

- Computer systems **do not know** what they are doing
  - They can only execute instructions (i.e. information) to operate over some other information
- Systems can only be **instructed** to protect the security properties of that information by means of some mechanism
  - E.g. Confidentiality → authentication
  - E.g. Integrity → crypto
  - E.g. Availability → redundancy

Trust in information security

- "The core idea of **computer security** is to instruct and trust a computer system not to compromise the security properties of the information it itself manages."

Allodi, Luca

- "The core idea of **home security** is to instruct and trust burglars to supervision and prevent themselves from stealing from a house they have complete access to."

Nobody, ever

Trust in information security (more formally)

- Ken Thompson
  - 1983 ACM Turing Award Lecture
  - **"Reflections on Trusting Trust" → mandatory reading**
    - http://dl.acm.org/citation.cfm?id=358210
- What software/system can one trust?
- Imagine an authentication mechanism
  - User inputs username
  - User inputs password
  - User presses enter
  - User has access to Desktop

Trusting trust (1)

- The user trusts the authentication mechanism
- But what happened **really?**
  - Did the authentication sw do the actual match?
    - Will it only grant access if it matches **your** credentials?
  - Did it send your credentials to a third party?
  - Did it use your credentials to read and copy your data (e.g. as stored in an encrypted volume)?
- Question: How do you increase your level of trust with the software?
  - You look at the source code, but is this enough?

Trusting trust (2)

- What or who did really generate that software?
  - The human being that wrote the sw source code?
  - The compiler that compiled the sw source code?
  - The human being that wrote the compiler that compiles the sw source code?
  - The compiler that compiled the compiler that compiled the sw source code?
  - Etc..
- "Chicken or egg" problem
- **Who do you trust?**

Trusting trust (3) - Thompson's view

- A compiler is written in C and is compiled by a previous version of itself → it takes one generation to add a "backdoor" that will automatically be included in sw compiled with the next compiler

Compiler 1.0

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\ n ');
```

Compiler 1.1

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\ n ');
if(c == 'v')
        return(11 );
```

Compiler 1.2

```
c = next( );
if(c != '\\')
        return(c);
c =next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
```

Modify compiler such that "\v" is interpreted as "|"

11 is ASCII code for "|"

Now when new compiler version finds \v it inserts "|"

Trusting trust (3) - Thompson's view

Compiler 1.0

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\ n ');
```

Compiler 1.1

```
c = next( );
if(c != '\\')
        return(c);
c = next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\ n ');
if(c == 'v')
        compile(backdoor);
```

Compiler 1.2

```
c = next( );
if(c != '\\')
        return(c);
c =next( );
if(c == '\\')
        return('\\');
if(c == 'n')
        return('\n');
if(c == 'v')
        return('\v');
```

Modify compiler such that "\v" introduces a backdoor in the software

Trusting Trust (4) - Thompson's view

- The compiler can be modified in any way to include code that **never appears** in the sw source code
  - And depending on how many generations passed, it won't appear in the previous compiler versions source code either.
- Trusting trust

"You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code."

Thompson, Ken

Computer Security vs Network Security

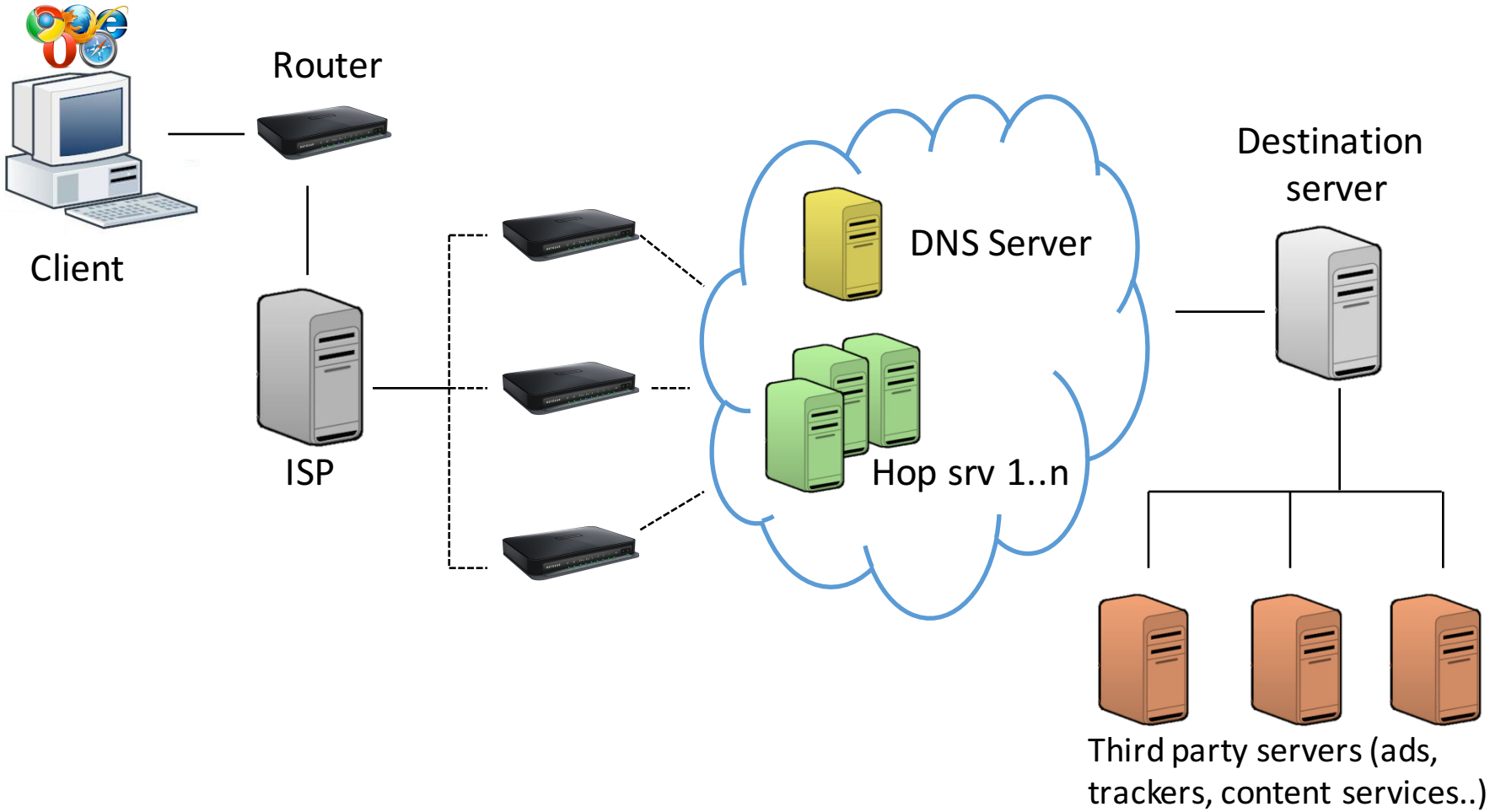
- A computer network is a general architecture that allows computer systems to share information remotely
- Network security is based on the same exact idea of preserving the CIA properties of information
- It makes for an especially interesting case
- Who can be trusted over the network?
  - Can you trust your own system?
  - Can you trust the communication channel?
  - Can you trust the destination system?

Confidentiality, Integrity, Availability over the network

- Question time:
  - An example of confidentiality threat that is created by a network communication?
  - An example of integrity threat that is created by a network communication?
  - An example of availability threat that is created by a network communication?

Client-server, for dummies edition

Routing client-server



Traceroute example for google.com

```
calvin:~ stewie$ traceroute -e www.google.com
traceroute: Warning: www.google.com has multiple addresses; using 74.125.136.104
traceroute to www.google.com (74.125.136.104), 64 hops max, 52 byte packets
 1  alicegate (192.168.1.1)  84.628 ms  98.506 ms  99.929 ms
 2  192.168.100.1 (192.168.100.1)  48.961 ms  45.948 ms  45.503 ms
 3  172.17.121.141 (172.17.121.141)  45.594 ms  45.811 ms  46.191 ms
 4  172.17.120.5 (172.17.120.5)  51.270 ms  46.060 ms  47.581 ms
 5  172.19.242.29 (172.19.242.29)  53.809 ms  47.827 ms  47.942 ms
 6  195.22.192.54 (195.22.192.54)  50.915 ms  52.674 ms  48.391 ms
 7  74.125.51.12 (74.125.51.12)  111.389 ms  113.022 ms  100.623 ms
 8  209.85.241.94 (209.85.241.94)  47.376 ms  46.401 ms
    209.85.241.92 (209.85.241.92)  48.155 ms
 9  72.14.232.76 (72.14.232.76)  56.724 ms  57.176 ms  63.901 ms
10  216.239.40.178 (216.239.40.178)  70.799 ms
    216.239.40.212 (216.239.40.212)  59.381 ms
    216.239.41.137 (216.239.41.137)  70.951 ms
11  216.239.41.130 (216.239.41.130)  67.473 ms  68.692 ms  69.244 ms
12  74.125.37.117 (74.125.37.117)  71.689 ms
    209.85.255.85 (209.85.255.85)  92.163 ms
    209.85.254.213 (209.85.254.213)  78.909 ms
13  72.14.233.109 (72.14.233.109)  77.700 ms  71.758 ms
    216.239.49.30 (216.239.49.30)  73.835 ms
14  * * *
15  ea-in-f104.1e100.net (74.125.136.104)  72.907 ms  71.416 ms  70.586 ms
calvin:~ stewie$
```

Possible attacker actions

- A general attacker might:
  - Infiltrate the communication in between hops
    - Impersonificate the client
    - Modify connections/routing/..
  - Be/infiltrate one of the hops
    - Act “legally” until end of service (after which it may act maliciously)
- How do you know if any of this happened?

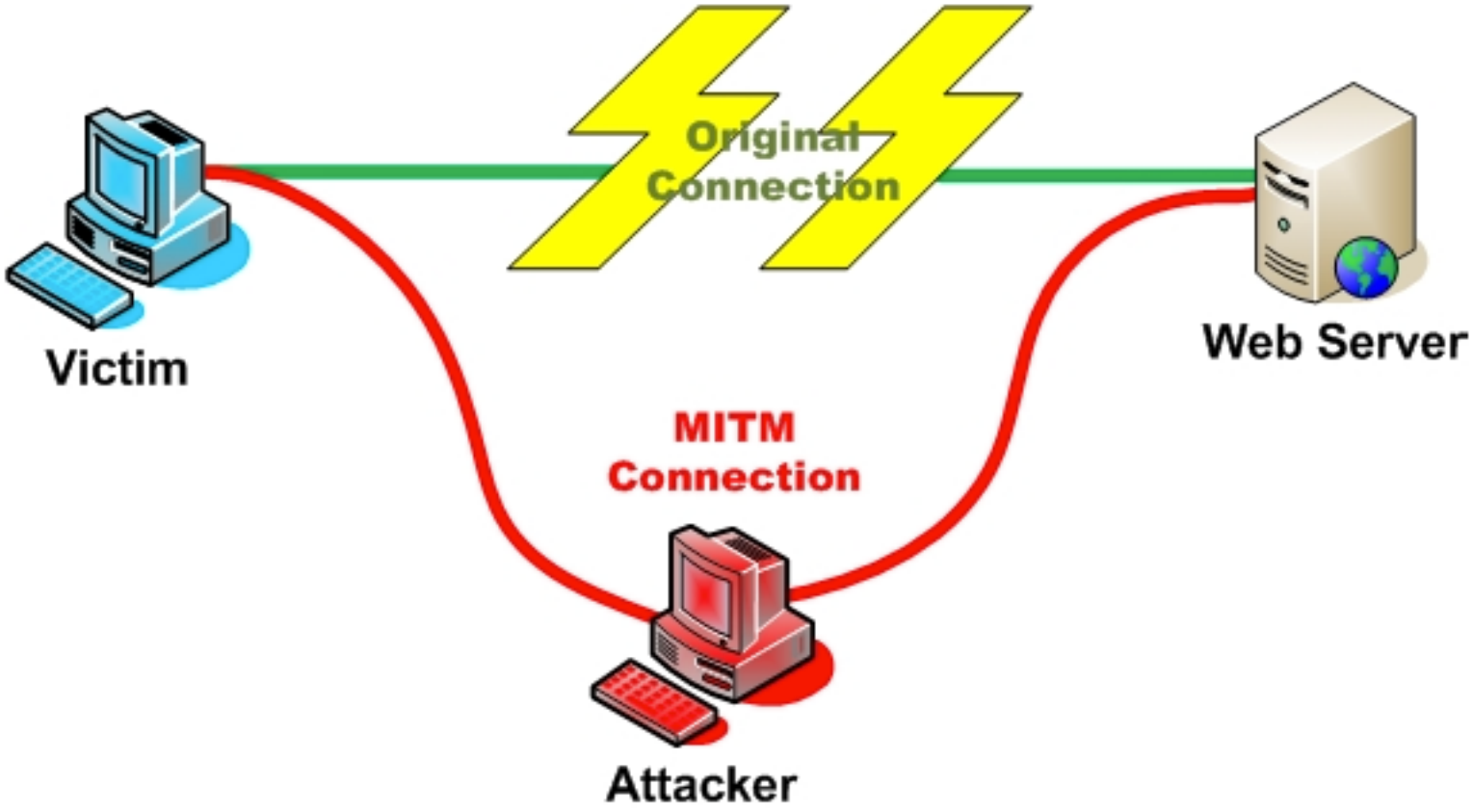
Attack models in network setting: outright malicious attacker

- Typically the malicious attacker aims at reading or modifying the communication (in part or fully)
  - That's a confidentiality, integrity, availability problem
- In this contest, this attacker is typically called "man in the middle"
  - Or "man in the browser"
- Attacker can intercept and act upon a communication between client and server
  - Channel redirection
  - Block communication entirely
  - Spoofing the user's identity

Outright malicious attacker

- Example: injection of malicious content
  - Manipulation of server response
    - Client's answer can also be modified by the attacker
  - Connection Hijack
    - Attacker injects him/herself in the communication and spoofs the victim's identity

Attack example: malicious attacker Man in the middle attack



Attack models in network setting: Honest-but-curious attacker

- The goal of this attacker is to use the client's information after correctly handling the service
  - Typically resides at the service level
    - E.g. ISP
  - Typically implies confidentiality and possibly integrity threats
- Example
  - DB Server is the attacker. Provides agreed service correctly.
    - E.g. answers queries with correct data
  - After the query is delivered to the client, the server uses the query's information to perform user profiling

Wrap-up

- Today we have introduced Computer Security
  - Confidentiality, Integrity, Availability
- Can we trust computer systems?
- What happens when we add networks in the scenario?
- Suggested reading:
  - **Ken Thompson. "Reflections on Trusting Trust"**
    - http://dl.acm.org/citation.cfm?id=358210