0202:

Malicious Software:

* + Email Spam
  + Spyware
  + Sending web requests
  + Scanning for vulnerabilities
  + Data crunching
  + Back door for future access

Botnets

* Sizes are measured in hundred thousand or millions
* Wait quality for instructions
* Rented out for profit
* Can be used for any proposes

Hacker:

* White hacker
* Black hacker
  + Curious users
  + Thrill users
  + Trophy hunters
  + Script kiddies
  + Professionals

0204:Total security

Zero day exploits

* Security vulnerabilities known only to hackers
* Kept security and traced
* Developers has had “zero days” of non-awareness

0205: Weakest link

0206: Security policy

* Communicates how information assets are protected
* Rules and guidelines
* Keep it simple, clear, and easy to follow
* Involve all stakeholders
* Review
* periodically security concerns changes

Security policy:

* Define the scope of the policy
* Identify and classify data to be protected or controlled.
* Map the integration of people and systems(Who need access and what kind of access)
* Define the handling procedures for each type of data

E.g: Backup credit card data make sure not misuse the data

* Designate user or department responsibilities

0301: General security principals

Least privilege “Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job”

Least privilege benefits:

* Code stability
  + Controlled data access
  + Easier to test action and integration
* System Security
  + Vulnerabilities are limited and localized

0302: Simple and more secure

* Complexity invited bugs
* Use clearly named function and var’s
* Write code comments
* Break up long sections of code into small functions
* Don’t repeat your self
* Legacy code is security concern
* Build in functions are often better than your own versions
* Disable or remove unused features when possible
  + Ex: Delete un used packages and var’s

0303: Never trust Users

0304: Except the unexpected

* Security is not reactive
* Prevent the crime before it happens
* What are all the things a user could try on the page?
* Consider Edgecase

0305: Defense in Depth

* Layered defenses
* Redundant security
* People: Educate the people
* Technology : firewalls, Hardware, Servers, Database, Encryption
* Operations: data banding procedures, procedures

0306: Security through Obscurity

* More information benefits hacker
* Limits exposed information
* Limit feeback
  + Ex: in login failed error do not give more details in error message

0307: Black listing and white listing

* Black listing
  + No access list
  + Reference list for what forbidden
* White listing
  + Opposite to black listing but not equal
  + Reference list for what is permitted
    - EX: for http request blockes

Black list {div, span, image, input, form……}

White list {p, br, strong, em}

0308: Map exposure points and data passages ways

* Incoming exposure points
  + URLS
  + Forms
  + Your public API’s
* Outputting Exposure points
  + HTML – Java scripts/Json/XML/ RSS
  + Cookie/ sessions – Database writes
  + Third party API (ex: Google maps)
* Mapping Data Passage ways
  + What path does data take
  + Understand site topography
  + Awareness + protection = Secuity

04. Filtering Input, Controlling output

0401: Regulating requests methods

* Get requests: URL’s ,links
* Post requests: forms
* Connect, delete, Head, Options, Put, Trace

Make sure application accepts only need request instead accessing all kind of requests

* Regulate Request/Response format
* Request content type is format of sent data[what type data need to send as req]
  + Request “Accept” in format for returned data [in what format need to accept the data] ex:{HTML, JSON, XML, TEXT}

0402: Validate Input

* First step in security
  + Is the input accessible?
  + Determine the data expectations
  + Preventing the bugs, as well as hacks
  + Consider application and database requirement

Ex: In DB you need to access a field like length 255 char

* Validate Input
  + Only allow expected data in submission

Ex: Form has first name, Last name etc. then accepts only those fields

Fields {firstname, last name}

Hacker will send all kind of date do not put birthdates, passwords, ss-numbers

* + - Set Default values
* Common Validation
  + Presence/length

Ex: Mandatory fields

Min length Max length of fields

* + Type:
    - Type of data not accepting special characters
    - File upload types
  + Format:
    - Ex: Email address
    - Within a set of values
    - White list: Values with 1-10 like that
    - Black list: not include values
  + Uniqueness:
    - Check is the values used or not like NaN

0403: Sanitizing Data

* Is the data potentially dangerous
* Neutralize
* render harmless
* use type casting
* not type juggling

Type Casting V/s Type juggling

1==”1” 1+”1”

Better to use 1==(int) “1”

Sanitizing Data:

* Sanitize SQL, HTML, Java Script, Json, XML etc
* Make sure language powerful characters not slip into String
* Encoding characters: replace powerful characters with harmless equivalents

0404: Labeling Variables

Use names to identify condition of data

PHP code: Raw\_Email = post{Email}

After sanitize safe\_Email = post{Email}

0405: Keeping code private

* Public directory –(On Stage)
* Accessible by the web browser
* Point of entry to your web site or application
* Library directory
  + Not accessible by the web browser
  + Accessible by your code via the file system

Need to configure the web browser to connect private& public code

* Private Code
  + Web server configuration
    - Set document root( this is public directory in this dir all the information in public)
    - Allow/deny access
  + .ht access file: set the access configuration information in server

Like Authorization, authentication, blocking

0406: Keeping credentials private

* Plain text credentials are dangerous
* Give them highest level of care
* Keep them separate form code
* Keep them out of version control system
* Have a few copies as necessary
* Do not reuse passwords
* Unique for every computer, environment, database
* Private hashed passwords whenever possible
* Public key cryptography (public key+ private key)
* SSH key, SSH agent(UNIX), Key chain(Mac), SSH agent Forwarding

0407: Keeping error Message Vague

* Turn off detained error reporting for production server
* Return generic 404 and 500 error pages
* Developers can lookup errors in log file
* Configure web server to use same error page

0408: Smart logging

Protection & detection (logging)

* Smart logging
* Error
* Sensitive action
* Possible attacks
* Data worth logging
* Date and time
* Source(User or IP)
* Action
* Target
* Cookie/session
* URL and all parameters
* Back trace
* Activity logging
  + Keep an activity history or audit trail
  + Add database table called “logs”
  + Write a function to add entries with a time stamp
  + Call the function whenever an admin make changes
  + Review logs routinely
* Smart logging
  + Don’t log sensitive DATA
  + Especially if application has built-in logging
  + Beware post parameters and database queries (filter out passwords, keys, tokens)
* Keep old content
  + Versions
  + Paranoid delete (means actually don’t delete set the flag as to act deleted. You can retrieve in case of chaos)

0501: Cross site scripting (XSS)

* XSS in short
* Hacker can inject java script into web page.
* Used to trick users into running java script code
* Used to steal cookie
* “Cross-site” because scripting in done via another website
* Successful because browser trust the javascript
* Browser willing lots the java script access cookie data