WEB SECURITY 3

Web Security and Privacy
A.A. 2020/2021

Leonardo Querzoni

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

CROSS SITE REQUEST FORGERY

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

CSRF

- Cross Site Request Forgery or CSRF/XSRF
  - Pronounced “c-surf”
- Works by exploiting the trust website has for a user
  - Victim: web application
  - Attack vector: user session
- Basic requirement: the user has a valid authenticated session with the target website

Some of this content is from EncriptionIT

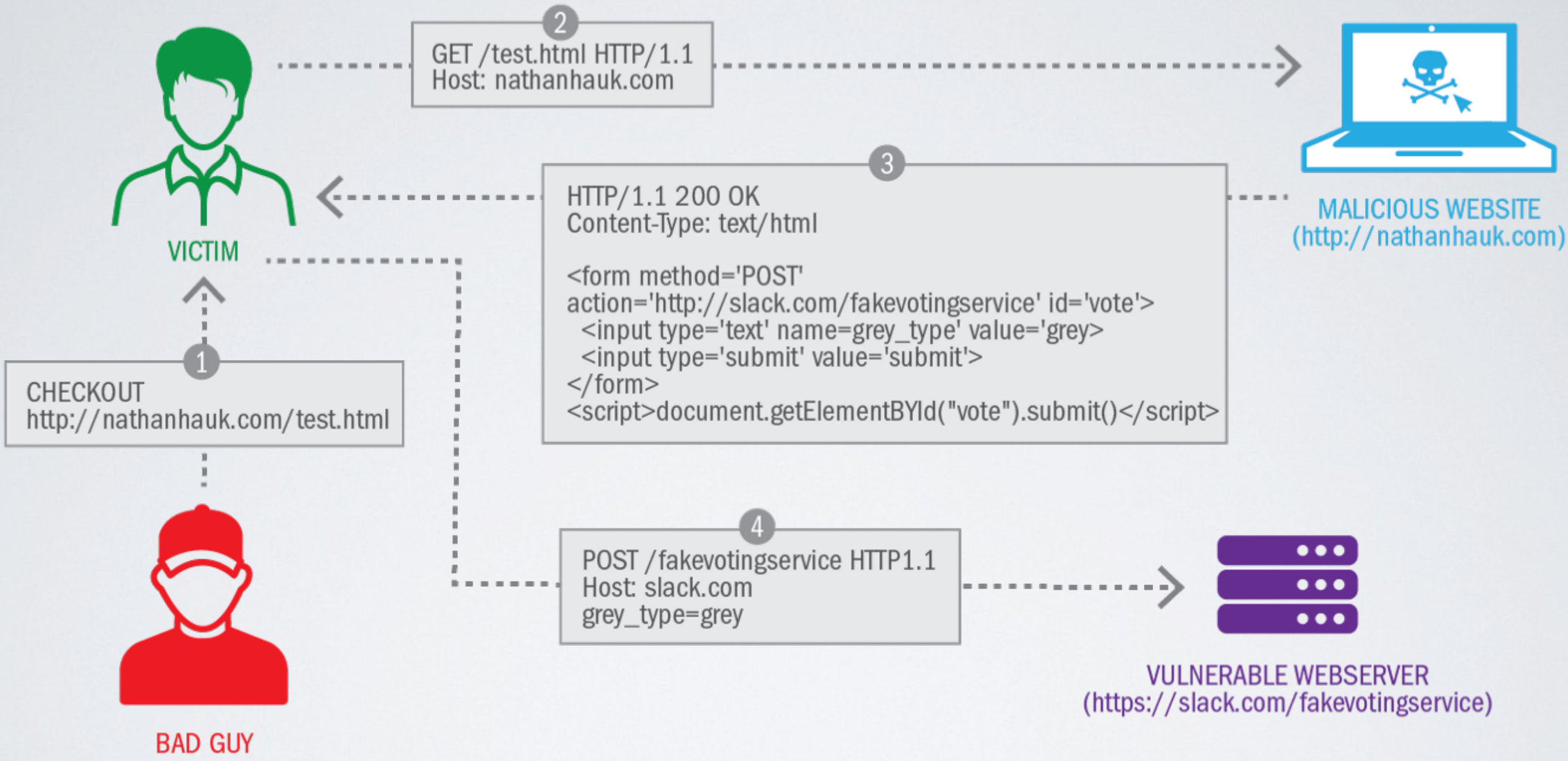
CSRF

How does it work?

- The attacker observes how users interact with the target app
  - He may study how data can be sent to the application through form POST
- Attacker builds a fake request that mimics a valid user request
  - Can't issue the request directly as he still lacks a valid user session!
- Lures a user with a valid authenticated session to visit a website that silently submit data to the target application using the user's session
  - The data submission must be triggered silently

Some of this content is from EncriptionIT

CSRF


2
GET /test.html HTTP/1.1
Host: nathanhauk.com
MALICIOUS WEBSITE
(http://nathanhauk.com)
3
HTTP/1.1 200 OK
Content-Type: text/html
<form method='POST'
action='http://slack.com/fakevotingservice' id='vote'>
<input type='text' name=grey_type' value='grey>
<input type='submit' value='submit'>
</form>
<script>document.getElementBYId("vote").submit()</script>
VICTIM
1
CHECKOUT
http://nathanhauk.com/test.html
BAD GUY
4
POST /fakevotingservice HTTP1.1
Host: slack.com
grey_type=grey
VULNERABLE WEBSERVER
(https://slack.com/fakevotingservice)

CSRF

Steps

- Use a proxy to intercept a non malicious request
  - Take note of the request parameters
  - You need to carefully recreate the request!
- Setup a malicious webpage containing the the CSRF attack
  - e.g. setup a form that recreates the request
  - make sure the form is autosubmitted silently when the webpage is hit
- Send the user a phishing email luring him in visiting your attack webpage.

Some of this content is from EncriptionIT

CSRF

Where is the problem?

- The target application blindly trusts every request that comes from authenticated users
- This happens also if the user request does not follow a corresponding challenge from the server
- i.e. a POST submitting a form should happen only if the server previously served that form!

Some of this content is from EncriptionIT

CSRF

Solution

- Make sure all forms require some sort of random token or secret on every submission
  - Define a CSRF token for each user session
    - Unique per user session
    - Secret
    - Unpredictable
  - Include the token in each single form
  - Upon submit check the submitted token is equal to the current session token
- Most web development frameworks include functionalities to automatically implement CSRF protection through tokens.

CSRF

Further info

- https://owasp.org/www-community/attacks/csrf

CONTENT SECURITY POLICY

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

CONTENT SECURITY POLICY (CSP)

What is it?

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including XSS. HTTP response header that provides tighter control over which scripts and content will be loaded (and run) on your site.

Why use it?

Mitigates XSS (cross-site scripting) and other injection attacks.

Some of this content is from UWashington slides

CONTENT SECURITY POLICY (CSP)

XSS attacks are based on the trust the browser has on content provided by the server.

CSP makes it possible for server administrators to reduce or eliminate the vectors by which XSS can occur by specifying the domains that the browser should consider to be valid sources of executable scripts.

ENABLING CSP

Set an HTTP response header:

`Content-Security-Policy: policy`

Alternatively, add a <meta> element to the page:

```
<meta http-equiv="Content-Security-Policy"
content="…">
```

Some of this content is from UWashington slides

CSP EXAMPLE

A policy is set by specifying a set of directives.

```
Content-Security-Policy:
     default-src 'self';
     img-src *;
     media-src *.media1.com *.media2.com;
     script-src userscripts.example.com
```

Some of this content is from UWashington slides

CSP EXAMPLE

```
Content-Security-Policy: default-src ‘self’; img-src
*; media-src *.media1.com *.media2.com; script-src
userscripts.example.com
```

By default, content is only permitted from the document’s origin, with the exceptions outlined by subsequent directives.

- Images may be loaded from anywhere.
- Media (audio/video) is allowed from media1.com and media2.com, as well as any subdomains of those sites.
- Executable scripts are allowed only from userscripts.example.com.

Some of this content is from UWashington slides

CSP

```
Content-Security-Policy: default-src ‘self’; img-src
*; media-src *.media1.com *.media2.com; script-src
userscripts.example.com
```

This strongly reduces the attacker’s options to include in a target web page external content to exploit the vulnerable parameters.

Some of this content is from UWashington slides

CSP

In addition to whitelisting specific domains, content security policy also provides two other ways of specifying trusted resources: nonces and hashes:

- The CSP directive can specify a **nonce** (a random value) and the same value must be used in the tag that loads a script. If the values do not match, then the script will not execute. To be effective as a control, the nonce must be securely generated on each page load and not be guessable by an attacker.
- The CSP directive can specify a **hash of the contents of the trusted script**. If the hash of the actual script does not match the value specified in the directive, then the script will not execute. If the content of the script ever changes, then you will of course need to update the hash value that is specified in the directive.

Some of this content is from UWashington slides

CSP LEVEL 1 DIRECTIVES

`default-src`	All assets
`script-src`	JavaScript
`style-src`	Stylesheets
`img-src`	Images
`connect-src`	Ajax, WebSocket, and EventSource
`font-src`	Fonts
`object-src`	Plugins e.g.`<object>`,`<embed>`,`<applet>`
`media-src`	Audio & Video
`sandbox`	Similar to iframe sandbox attribute

Some of this content is from UWashington slides

CSP BROWSER SUPPORT (DEC 2020)

Content Security Policy Level 2 - REC

Mitigate cross-site scripting attacks by only allowing certain sources of script, style, and other resources. CSP 2 adds hash-source, nonce-source, and five new directives

Usage: % of all users ?

Global: 90.2% + 4.25% = 94.44%

Current aligned | Usage relative | Date relative | Filtered | All

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
		2-30														
		1 31-34	4-35		10-22											
	12-14	2 35	4 36-38		4 23-25											
	9 15-18	3 36-44	5 39	3.1-9.1	5 26	3.2-9.3										
6-10	79-86	7 45-82	40-86	10-13.1	27-71	10-13.7		2.1-4.4.4	12-12.1				4-12.0			
11	87	7 83	87	14	72	14.2	all	81	59	87	6 83	12.12	13.0	10.4	7.12	7 2.5
		7 84-85	88-90	TP												

Source: https://caniuse.com/#feat=contentsecuritypolicy

CSP: QUIZ!

```
Content-Security-Policy: default-src ‘self’; img-src *; media-src *.media1.com *.media2.com; script-src userscripts.example.com
```

If this site’s origin is `https://www.testsite.com`, will external stylesheets be loaded with this policy?

Some of this content is from UWashington slides

CSP: QUIZ!

```
Content-Security-Policy: default-src ‘self’; img-src *; media-src *.media1.com *.media2.com; script-src userscripts.example.com
```

If this site’s origin is `https://www.testsite.com`, will external stylesheets be loaded with this policy?

No, since there is no explicit `style-src` directive, `default-src` is used as a fallback.

Some of this content is from UWashington slides

CSP: QUIZ!

```
Content-Security-Policy: default-src ‘self’; img-src
*; media-src *.media1.com *.media2.com; script-src
userscripts.example.com
```

If this site’s origin is `https://www.testsite.com`, will scripts from the its own origin be loaded and executed?

Some of this content is from UWashington slides

CSP: QUIZ!

```
Content-Security-Policy: default-src ‘self’; img-src *; media-src *.media1.com *.media2.com; script-src userscripts.example.com
```

If this site’s origin is https://www.testsite.com, will scripts from the its own origin be loaded and executed?

No, since ‘`self`’ is not included in the `script-src` directive, only those from `userscripts.example.com` would be loaded.

Some of this content is from UWashington slides

ACCESS CONTROL ATTACKS

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

ACCESS CONTROLS

- A system which enables an **authority to control access to areas and resources** in a given physical facility or computer-based information system.
- Access controls are a critical defense mechanism within the application because they are responsible for making the decision of whether it should permit a given request to perform its attempted action of access the resources that it is requesting.
- When they are defective, an attacker can often compromise the entire application, taking control of administrative functionality and accessing sensitive data belonging to every other user.
- Are among the most commonly encountered categories of web application vulnerability

source: "Attacking Access Control" J. van Malsen

VERTICAL VS HORIZONTAL ACCESS CONTROLS

- **Vertical Access Controls**: allow different types of users to access different parts of the application's functionality
  - Different types of users have access to different application functions
  - Mechanisms that restrict access to sensitive functionality that is not available to other types of users
    - used to enforce business policies such as separation of duties and least privilege

source: "Attacking Access Control" J. van Malsen

VERTICAL VS HORIZONTAL ACCESS CONTROLS

- **Horizontal Access Controls**: allow users to access a certain subset of a wider range of resources of the same type
  - Restrict access to resources to the users who are specifically allowed to access those resources
  - Web mail application may allow you to read your email but no one else's; you can only see your own details
  - A banking application will allow a user to view transactions and make payments from their own accounts, but not the accounts of any other user.

source: "Attacking Access Control" J. van Malsen

ACCESS CONTROL VULNERABILITIES

- Access controls are broken if any user is able to access functionality or resources for which he is not authorized
- Two main types of attack against access controls
  - Vertical privilege escalation: when a user can perform functions that their assigned role does not permit them to do
  - Horizontal privilege escalation: when a user can view or modify resources to which he is not entitled

source: "Attacking Access Control" J. van Malsen

ACCESS CONTROL WEAKNESSES

- Completely Unprotected Functionality
- Identifier-Based Functions
- Multistage Functions
- Static Files

source: “Attacking Access Control” J. van Malsen

COMPLETELY UNPROTECTED FUNCTIONALITY

- In many cases of broken access controls, sensitive functionality and resources can be accessed by knowing the relevant URL
  - https://wahh-app.com/admin/ allows user to enter certain user interface
- Weaknesses:
  - URL can be guessed (especially by insider)
  - Link appears in browser histories and the logs of web/proxy servers
  - Users may write them down, bookmark them or email them around
  - They are not normally changed periodically, as passwords should be
  - When users change job roles, there is no way to delete their knowledge of a particular URL.

source: "Attacking Access Control" J. van Malsen

IDENTIFIER-BASED FUNCTIONS

- When a function of an application is used to gain access to a specific resource, it is very common to see an identifier for the requested resource being passed to the server in the request.
- In order to be able to open the link/application an attacker needs to know the name of the application page and the identifier of the document he wishes to view.
- Weaknesses:
  - Passwords often easy to guess
  - Lots of people write down resources identifiers or save them on their computer, so easy to find

source: "Attacking Access Control" J. van Malsen

MULTISTAGE FUNCTIONS

- Involves capturing different items of data from the user at each stage. This data is strictly checked when first submitted and then is usually passed on to each subsequent stage, using hidden fields in an HTML form.
- Main Weaknesses:
  - Often assumed by the developers is that any user who reaches the later stages of the process must have the relevant privileges because this was verified at the earlier stages
  - Also often assumed is that people will access application pages in the intended sequence; by taking “other path” people could avoid user identification

source: “Attacking Access Control” J. van Malsen

STATIC FILES

- In some cases, requests for protected resources are made directly to the static resources themselves, which are located within the web root of the server.
  - e.g. an online publisher may allow users to browse its book catalog and purchase ebooks for download. Once the payment has been made, the user is directed to a download URL.
    - As this is a completely static resource, it does not execute on the server, and its contents are simply returned directly by the web server.
- When static resources are accessed in this way, it is highly likely that there are no effective access controls protecting them and that anyone who knows the URL naming scheme can exploit this to access ay resources they desire.

source: “Attacking Access Control” J. van Malsen

SECURING ACCESS CONTROLS: PITFALLS

- Access controls are one of the easiest areas of web application security, though, there are several obvious pitfalls to avoid:
  - Usually arise from ignorance about the essential requirements of effective access control or flawed assumptions about possible user requests.
  - Web application developers often implement access control functions on a piecemeal basis, adding code to individual pages.
  - Approach:
    - Do not trust any user-submitted parameters to signify access rights (such as admin = true)
    - Do not assume that users will access application pages in the intended sequence (make sure people will also not be able to avoid access controls by taking a different “path”)
    - Do not trust the user not to tamper with any data that is transmitted via the client. ➔ If some user-submitted data has been validated and is then transmitted via the client, do not rely upon the retransmitted value without revalidation.

source: “Attacking Access Control” J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (1)

- Explicitly evaluate and document the access control requirements for every unit of application functionality.
- This needs to include both who can legitimately use the function and what resources individual users may access via the function.
- Development without planning is a common source of vulnerabilities.
- Design for:
  - Least privilege
  - separation of duties

source: “Attacking Access Control” J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (2)

- Drive all access control decisions from the user's session

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (3)

- Use a central application component to check access controls
- Advantages:
  - Increases the clarity of access controls within the application, enabling different developers to quickly understand the controls implemented by others
  - Maintenance is more efficient and reliable
  - Improves adaptability. Where new access control requirements arise, these can be easily reflected within an existing API implemented by each application page
  - Results in fewer mistakes and omissions than if access control code is implemented piecemeal throughout the application

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (4)

- Process every single client request via this component to validate that the user making the request is permitted to access the functionality and resources being requested

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (5)

- Use programmatic techniques to ensure that there are no exceptions to the previous point.
  - An effective approach is to mandate that every application page must implement an interface that is queried by the central access control mechanism.
- By forcing developers to explicitly code access control logic into every page, there can be no excuse for omissions

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (6)

- For particularly sensitive functionality, such as administrative pages, you can further restrict access by IP address to ensure that only users from a specific network range are able to access the functionality, regardless of their login status.

source: “Attacking Access Control” J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (7)

- If static content needs to be protected, there are two methods of providing access control:
  - Static files can be accessed indirectly by passing a file name to a dynamic server-side page which implements relevant access control logic
  - Direct access to static files can be controlled using HTTP authentication or other features of the application server to wrap the incoming request and check the permissions for the resource before granting access.

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (8)

- Identifiers specifying which resource a user wishes to access are vulnerable to tampering whenever they are transmitted via the client.
  - The server should trust only the integrity of server-side data. Any time these identifiers are transmitted via the client, they need to be revalidated to ensure the user is authorized to access the requested resource

source: “Attacking Access Control” J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (9)

- For security-critical application functions consider implementing per-transaction re-authentication and dual authorization to provide additional assurance that the function is not being used by an unauthorized party.
  - This will also mitigate the consequences of other possible attacks, such as session hijacking

source: "Attacking Access Control" J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (10)

- Log every event where sensitive data is accessed or a sensitive action is performed.
  - These logs will enable potential access control breaches to be detected and investigated

source: “Attacking Access Control” J. van Malsen

IMPLEMENTING EFFECTIVE ACCESS CONTROLS WITHIN WEB APPLICATIONS (11)

- Use layered access control
  - each application layer has its own access control implementation
    - From the UI, down to the backend DB
  - mitigates possible pitfalls arising from the design of a system-wide security model
- Enables the implementation of defense in depth policy

source: "Attacking Access Control" J. van Malsen

A MULTI-LAYERED PRIVILEGE MODEL

- Issues relating to access apply not only to the web application itself but also to the other infrastructure ties which lie beneath it
- Thinks about adopting a general purpose authorization schema:
  - Programmatic Control
  - Discretionary Access Control (DAC)
  - Role-Based Access Control (RBAC)
  - Declarative Control

source: "Attacking Access Control" J. van Malsen

ATTACKING ACCESS CONTROLS

- Finding a break in access controls is sometimes trivial
  - Request a common administrative URL and gain direct access to the functionality.
    - web server logs are full of directory brute-forcing attacks
  - In other cases, it may be very hard, and subtle defects may lurk deep within application logic, particularly in complex, high-security applications.
    - e.g. defect in the implementation of key selection in a proprietary secure transmission protocol
- The most important lesson when attacking access controls is to look everywhere. If you are struggling to make progress, be patient and test every single step of every application function. A bug that allows you to own the entire application may be just around the corner.

source: "Attacking Access Control" J. van Malsen

PROTECT DATA IN TRANSIT

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

SECURE CHANNELS

- When using an ordinary HTTP connection, users are exposed to many risks arising from the fact data is transmitted in plaintext
  - Subject to Main-in-the-middle attacks
- You should always consider the Internet as an adversarial environment
  - no control on its security
  - adversaries may control any link/router
- The standard solution is to use an end-to-end secure channel based on TLS

SECURE CHANNELS

- HTTPS is HTTP on top of TLS
- Guarantees
  - Confidentiality
  - Data Integrity
  - Authentication (if needed)
- Was initially developed to secure critical transaction
- Nowadays, its usage is suggested for ANY web-based interaction

TLS ANTI-PATTERNS

- Use HTTPS to protect critical paths only
  - Protect login procedures
  - Protect sensitive transactions
- Issues
  - Unprotected functionalities may leak session id
  - MitM attacks for session side jacking

TLS ANTI-PATTERNS

- Mix HTTP resources on HTTPS content
  - Common example: a https web page loads images through http
- Issues
  - Unprotected functionalities may leak session id
    - MitM attacks for session side jacking
- Solution:
  - use TLS everywhere
  - All cookies should be marked with the "Secure" attribute

TLS ANTI-PATTERNS

- Redirect users from HTTP to HTTPS
  - Browsers usually perform an https connection if you don't specify the protocol
  - Server responds with a 301/302 response
  - This "upgrades" the connection from HTTP to HTTPS
- Issues
  - Still vulnerable to "SSL strip attack"

SSL STRIP ATTACK

- IDEA: downgrade a HTTPS connection through HTTP with MitM attack
  - Attacker setup a device to intercept connections from his victim
    - Create a rogue wireless network
  - Proxies all connections
    - The first (HTTP) request is intercepted and passed to the server that answer with 301
    - Attacker does not forward the response to the client but rather setup the https connection with the server
    - Protected content from the server is forwarded on the unprotected channel with the victim
    - All subsequent requests are mediated from the proxy

SSL STRIP ATTACK

- Solution: enable HTTP Strict Transport Security (HSTS)
  - A web security policy mechanism that helps to protect websites against SSL stripping attacks and cookie hijacking
  - Allows web servers to declare that web browsers should interact with them using only secure HTTPS connections, and never via the insecure HTTP protocol.
  - Conformant user browsers will automatically redirect any insecure HTTP requests to HTTPS for the target website.
  - The policy includes a duration policy
  - It can be installed on a first HTTP request (still risky)
  - It can be added on a preloaded list of HSTS policies

THE COST OF VULNERABILITIES

SAPIENZA
UNIVERSITÀ DI ROMA

CIS SAPIENZA
CYBER INTELLIGENCE AND INFORMATION SECURITY

DEFINITIONS

- A vulnerability is a weakness which allows an attacker to reduce system's information assurance.
- A vulnerability is the intersection of three elements:
  - a system susceptibility or flaw
  - attacker access to the flaw
  - attacker capability to exploit the flaw

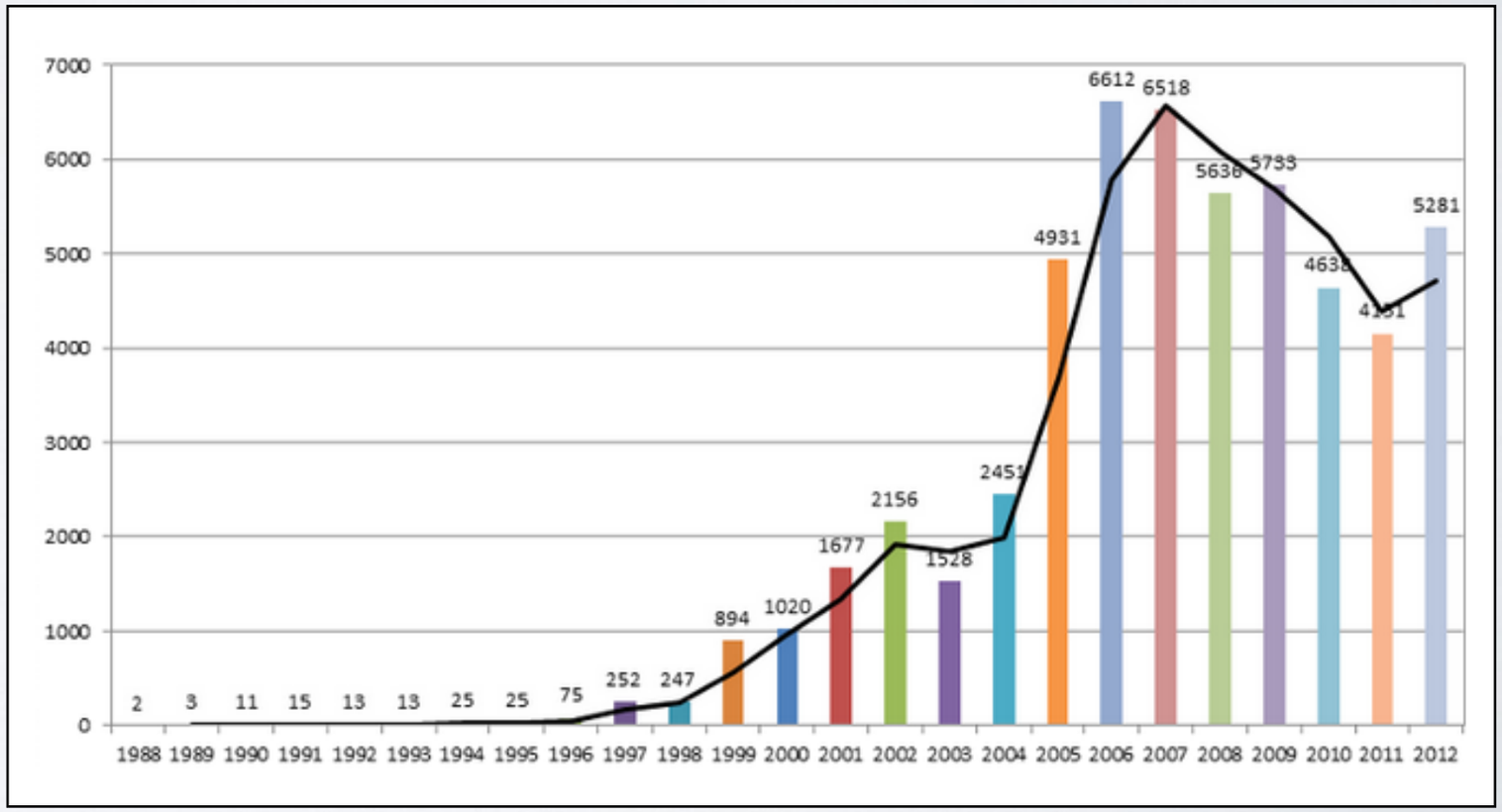
DEFINITIONS

- Causes:
  - Bugs
  - Design defects
  - Misconfigurations
  - Aging
- Fostering factors
  - System complexity
  - Connectivity
  - Incompetence

MOTIVATIONS

- Vulnerability in software is one of the major reasons for insecurity
  - 20 flaws per thousand lines of code (Dacey 2003)
  - Steady increase in vulnerability exploitations
- Fully secure software is unlikely
- 95% of breaches could be prevented by keeping systems up-to-date with patches (Dacey 2003)

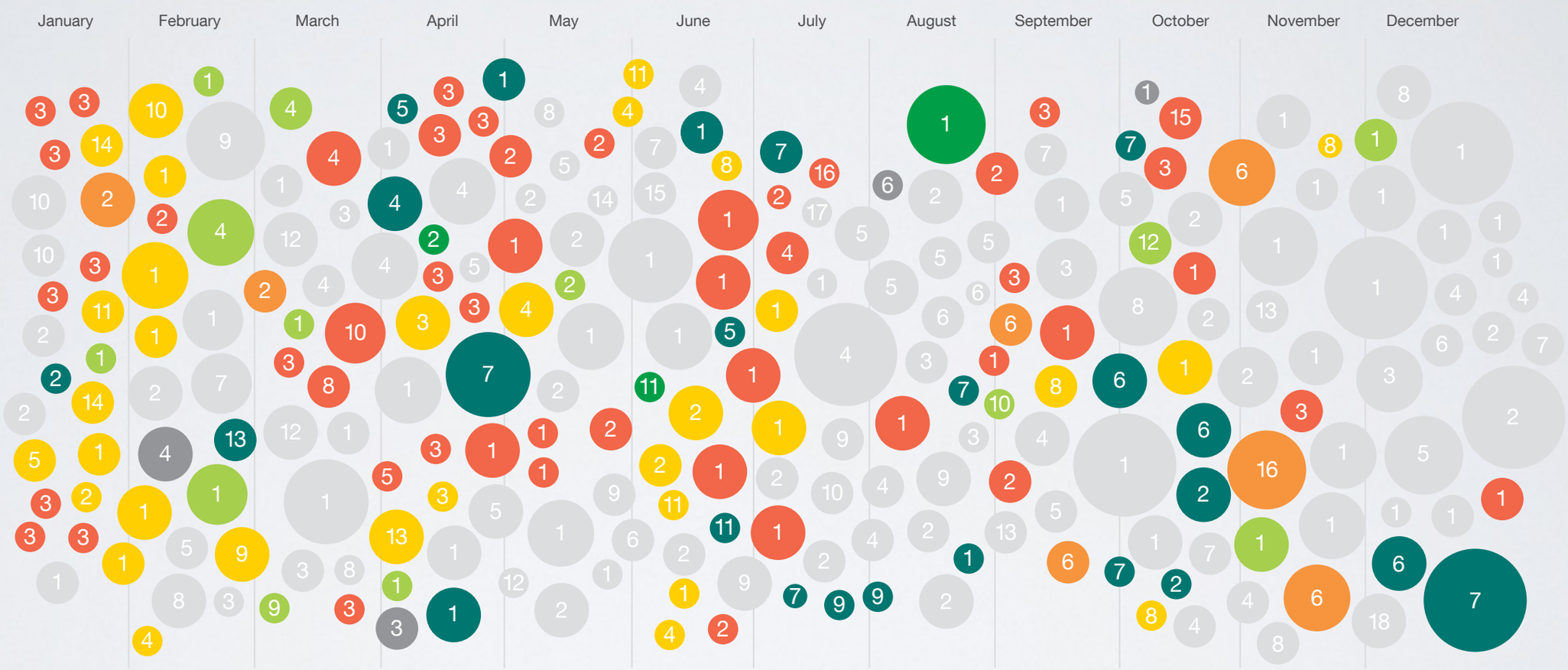
Source: 25 Years of Vulnerabilities: 1988-2012, Sourcefire



MOTIVATIONS

Sampling of 2013 security incidents by attack type, time and impact

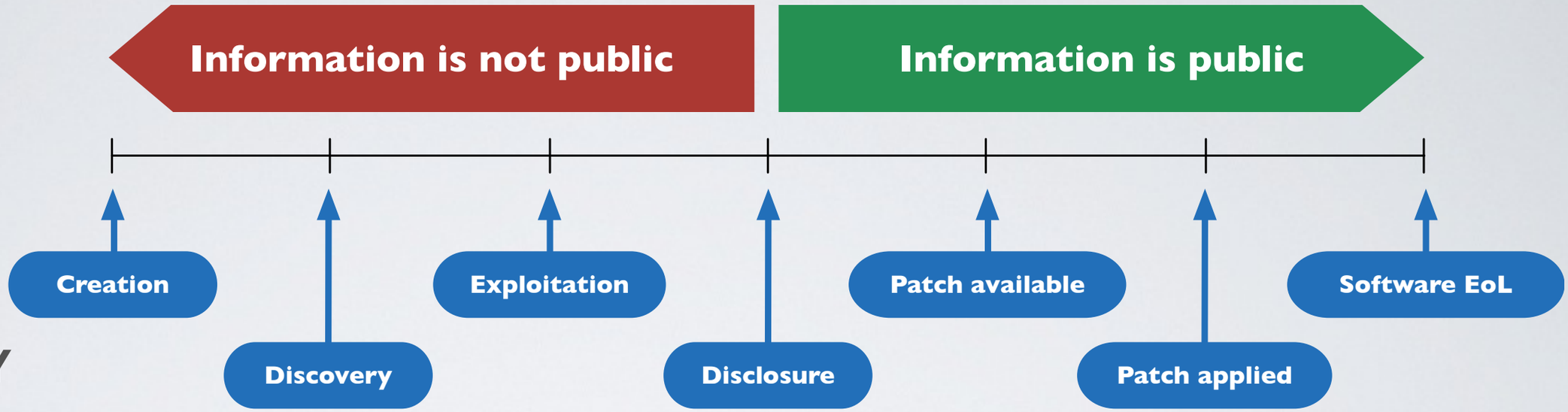
conjecture of relative breach impact is based on publicly disclosed information regarding leaked records and financial losses



Size of circle estimates relative impact of incident in terms of cost to business.

Source: IBM X-Force Threat Intelligence Quarterly - Q1 2014

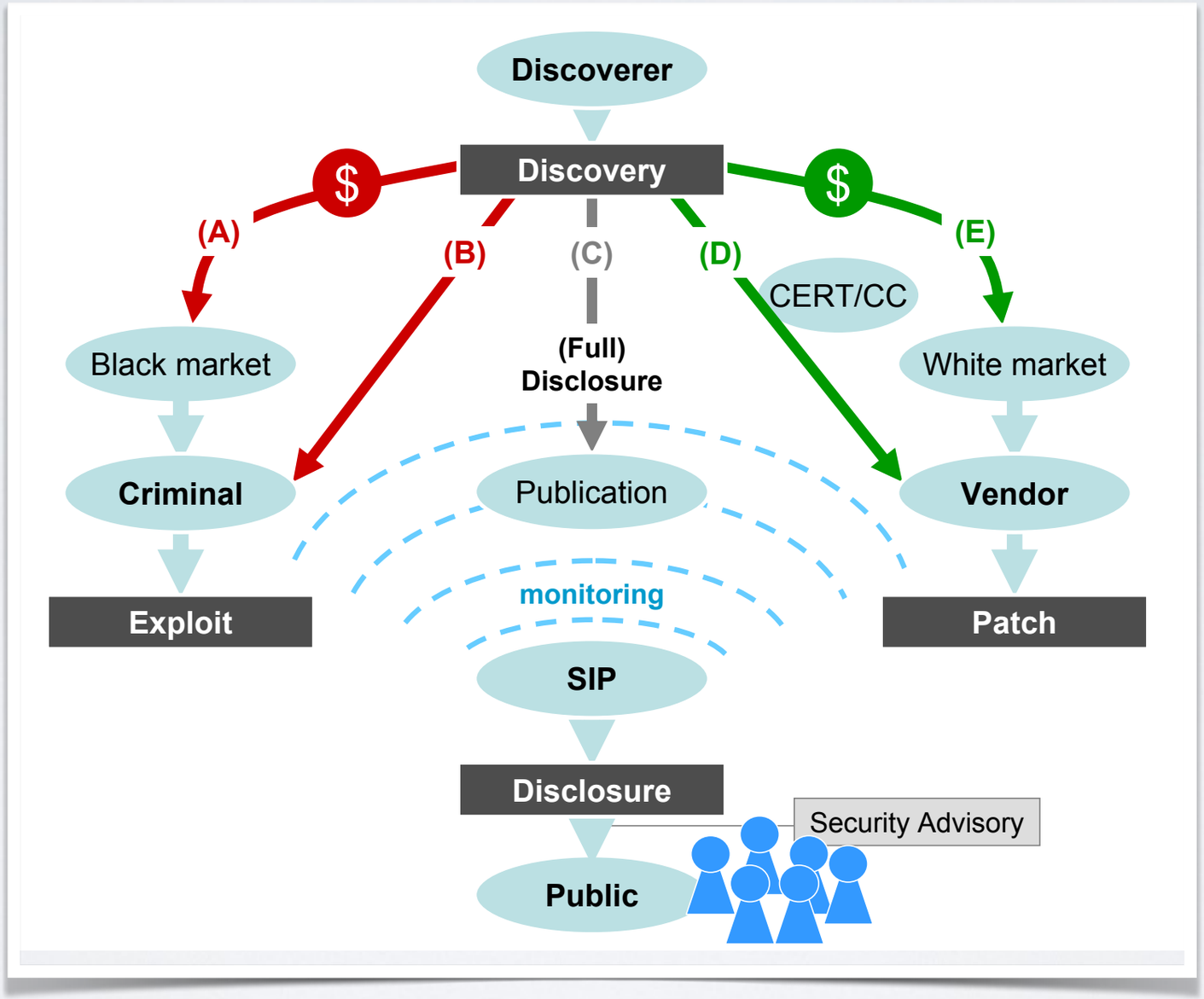
VULNERABILITY LIFE CYCLE



- **CREATION**
- **DISCOVERY**
  - Malicious users (hackers)
  - Benign users (final users, security firms, researchers, …)
- **EXPLOITATION**
- **DISCLOSURE**
  - Keep it secret ?
  - Publicly disclose ?
  - Sell ?!?!?!?
- **PATCH**

VULNERABILITY LIFE CYCLE

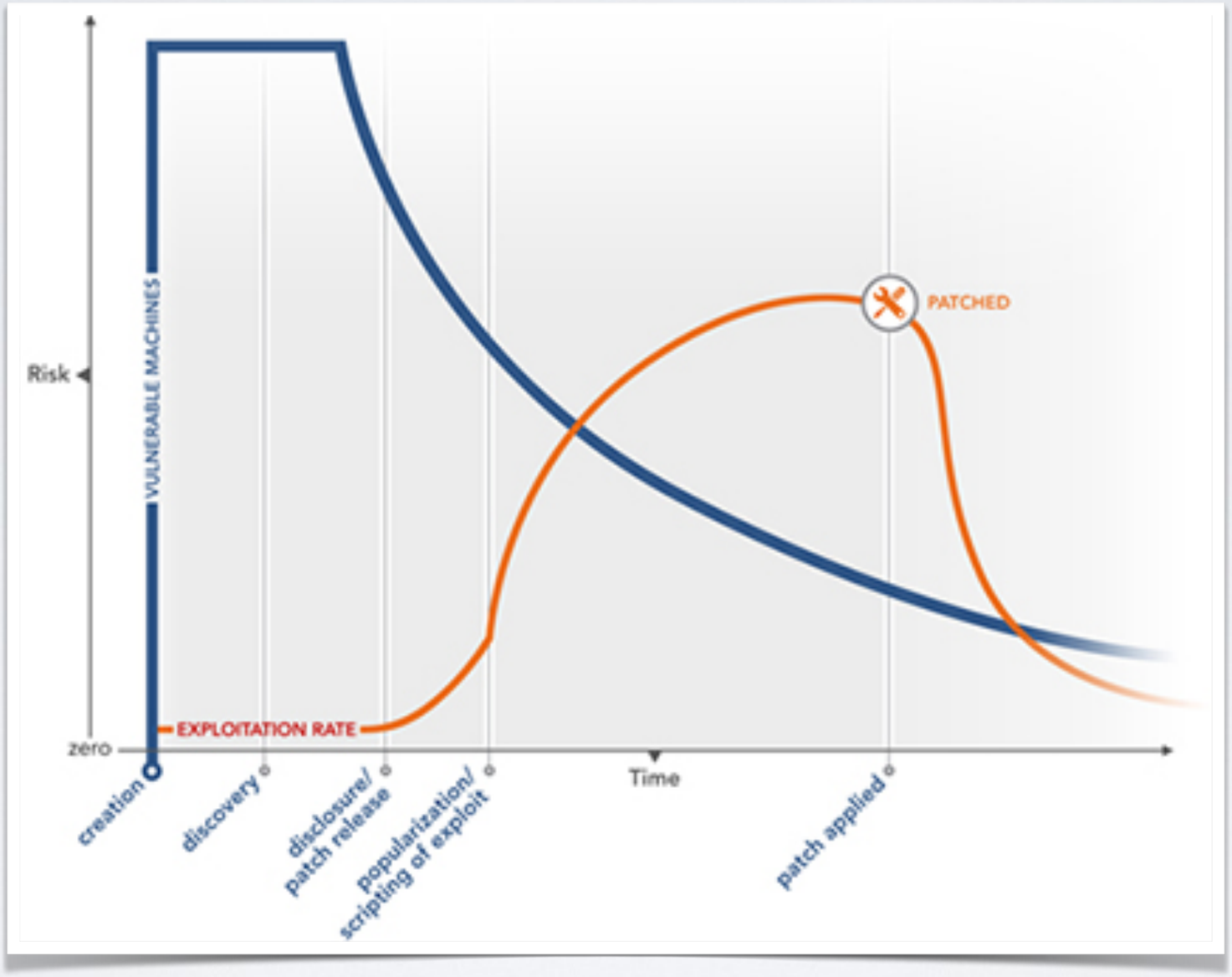
- The Discovery process:



Source: S. Frei, D. Schatzmann, B. Plattner and B. Trammell, Modelling the Security Ecosystem - The Dynamics of (In)Security,

VULNERABILITY LIFE CYCLE

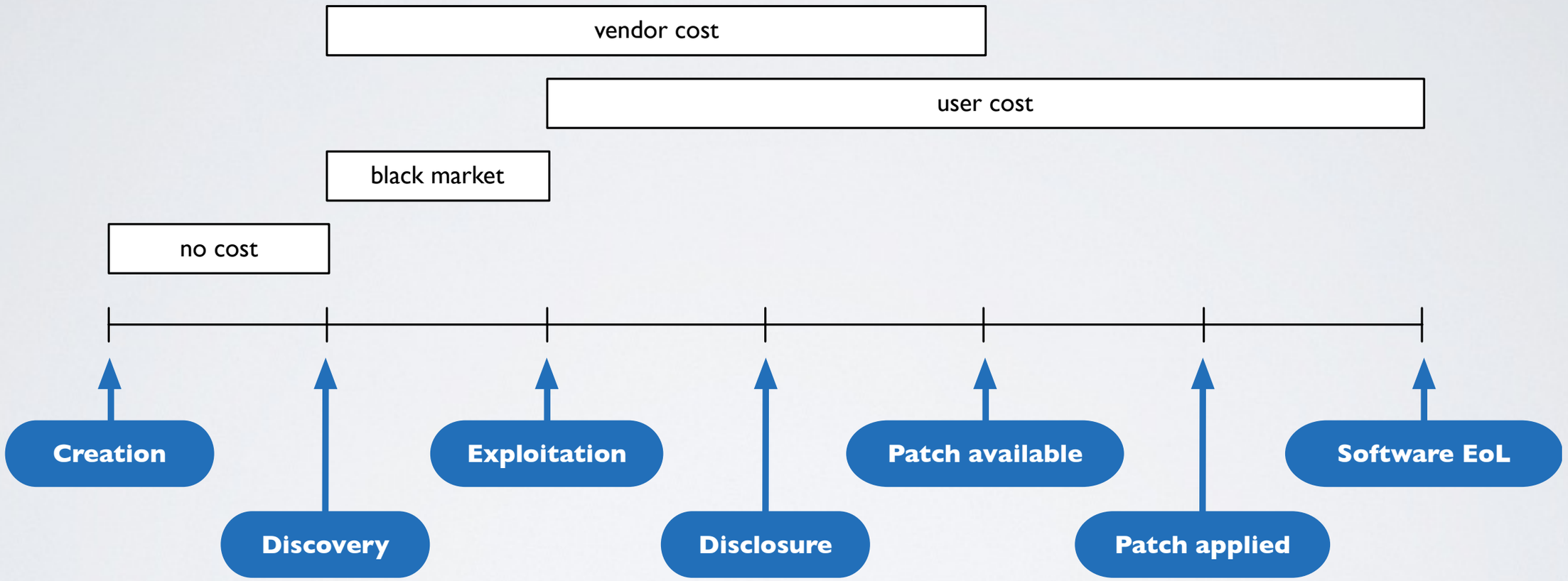
- *Patching*



Source: Defense through the vulnerability lifecycle, https://www.alertlogic.com/defense-through-the-vulnerability-lifecycle/

VULNERABILITY LIFE CYCLE

- Who pays ?



RESPONSIBLE DISCLOSURE

- Forget about malicious users
- What process should a responsible user follow to disclose the vulnerability ?
  - No consensus
  - CERT allows a 45-days grace period
  - OIS allows a 30-days grace period
  - Security firms follow their internal guidelines

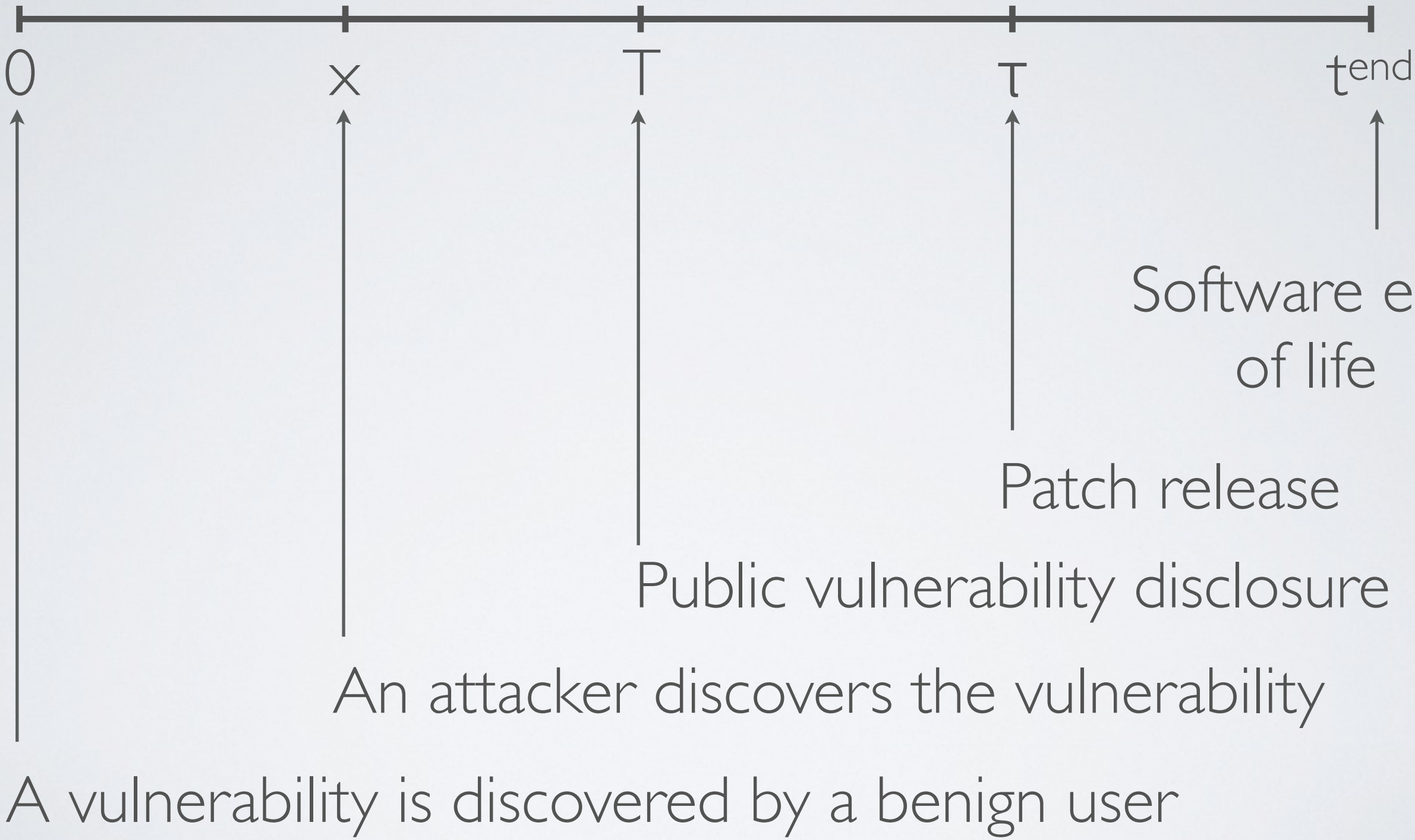
DISCLOSURE POLICY EFFECTS

- **Full Vendor Disclosure**
  - Promotes *secrecy*
  - Gives full control of the process to the vendor
- **Immediate Public Disclosure**
  - Promotes *transparency*
  - Gives the vendor a strong incentive to fix the problem
  - Allows vulnerable users to take intermediate measures
  - Immediate exposure to risks
- **Hybrid Disclosure**
  - Promotes both *secrecy* and *transparency*

ANALYTICAL MODEL

- An analytical model allows to identify critical choices and good behaviors
- Actors
  - Vendor: produces the software and is responsible for patching
  - Users: use the software provided by the vendor
  - Attackers: exploit vulnerabilities in software to attack users
  - Social planner: manages the disclosure of a vulnerability identified by a user

ANALYTICAL MODEL



ASSUMPTIONS

- Users and attackers follow fixed behaviors
  - Users incur losses when attackers exploit the vulnerability
  - Attackers exploit the vulnerability as soon as they are aware of it (either x or T, whichever is earlier)
- The vendor aims at minimizing its costs
  - It deterministically decides a time τ for patch release
- The social planner aims at minimizing the total societal cost
  - societal cost = cost incurred by vendor + cost incurred by users

ASSUMPTIONS

- Users incur a loss from the moment the vulnerability is known by an attacker to the patch release time
  - $l(\tau - x)$ or $l(\tau - T)$ cumulative user loss
  - $l(t)$ is increasing and strictly convex in t
  - $F(x)$ probability that an attacker discovers the vulnerability at time x.
- Therefore the pre-patch user loss is

$$L(\tau, T) = \begin{cases} \int_0^{\tau} l(\tau - x)\, dF(x), & \text{when } \tau < T \\ \int_0^{T} l(\tau - x)\, dF(x) + (1 - F(T)) l(\tau - T), & \\ & \text{when } \tau \geq T. \end{cases}$$

ASSUMPTIONS

- Vendors incur losses as well

$$V(\tau, q; T) = C(\tau, q) + \lambda \mathcal{L}(\tau, T, q)$$

ASSUMPTIONS

- Part ($\lambda<1$) of the user loss is internalized by the vendor
  - Loss of reputation
  - Contractual liability
  - Market competition
  - Customer support
  - Loss of future sales
  - ...

ASSUMPTIONS

- The vendor also incurs a patch development cost $C(\tau,q)$
  - $C_\tau(\tau,q)<0$
  - $C_{\tau\tau}(\tau,q)>0$

  Cost is decreasing and convex in τ. The later the patch is released, the less it costs.

  - $C_q(\tau,q)>0$
  - $C_{\tau q}(\tau,q)<0$

  Cost increases with quality.

ASSUMPTIONS

- The vendor will in any case develop a patch before the EOL

$$\min_{\tau}\{C(\tau, q) + \lambda \int_0^{\tau} l(\tau - x)\, dF(x)\} < \lambda \int_0^{t^{\text{end}}} l(t^{\text{end}} - x)\, dF(x)$$

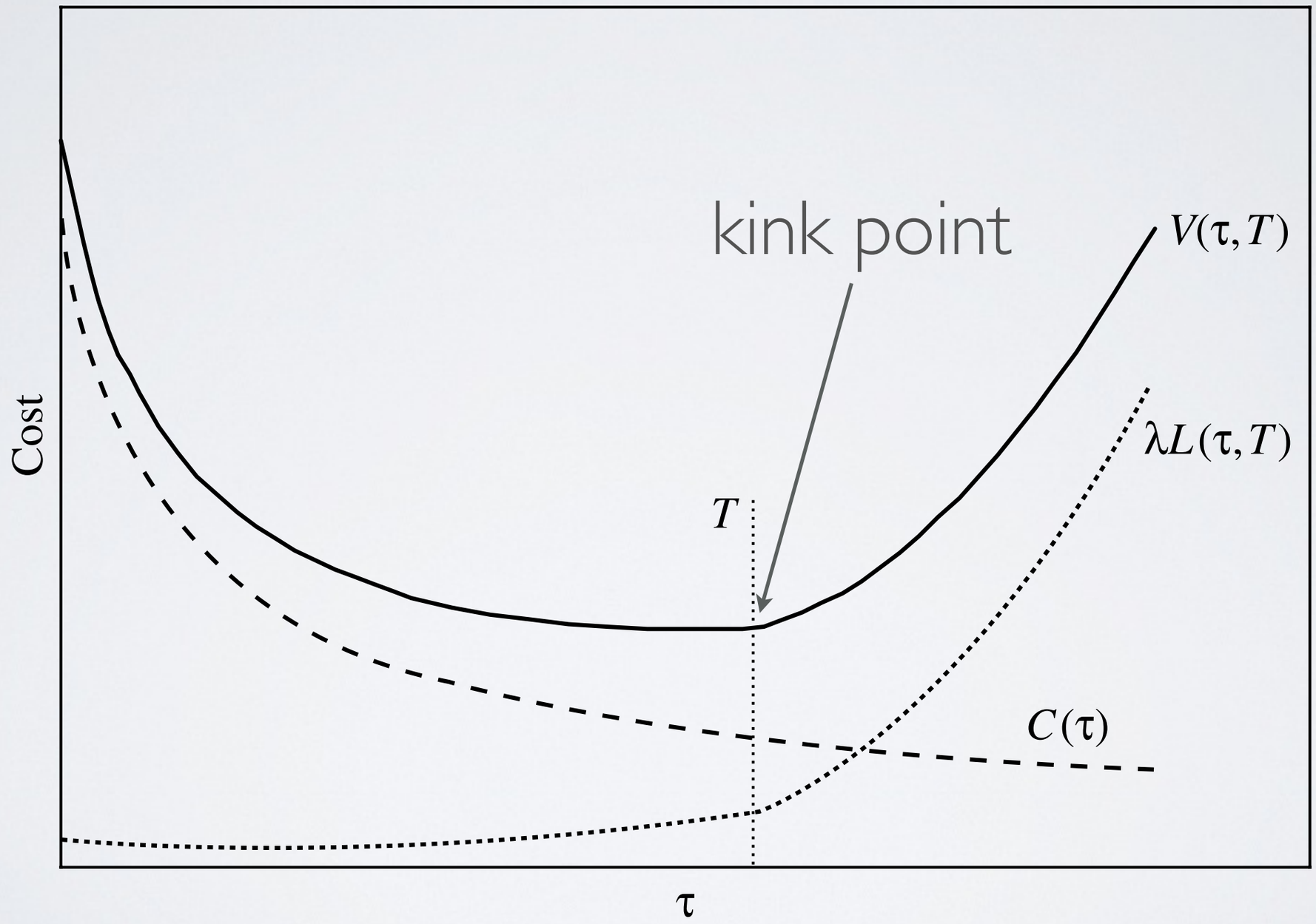
Minimum cost for patch development

Cost incurred if no patch is developed

- Patch quality is assumed as exogenous and fixed

VENDOR'S COST

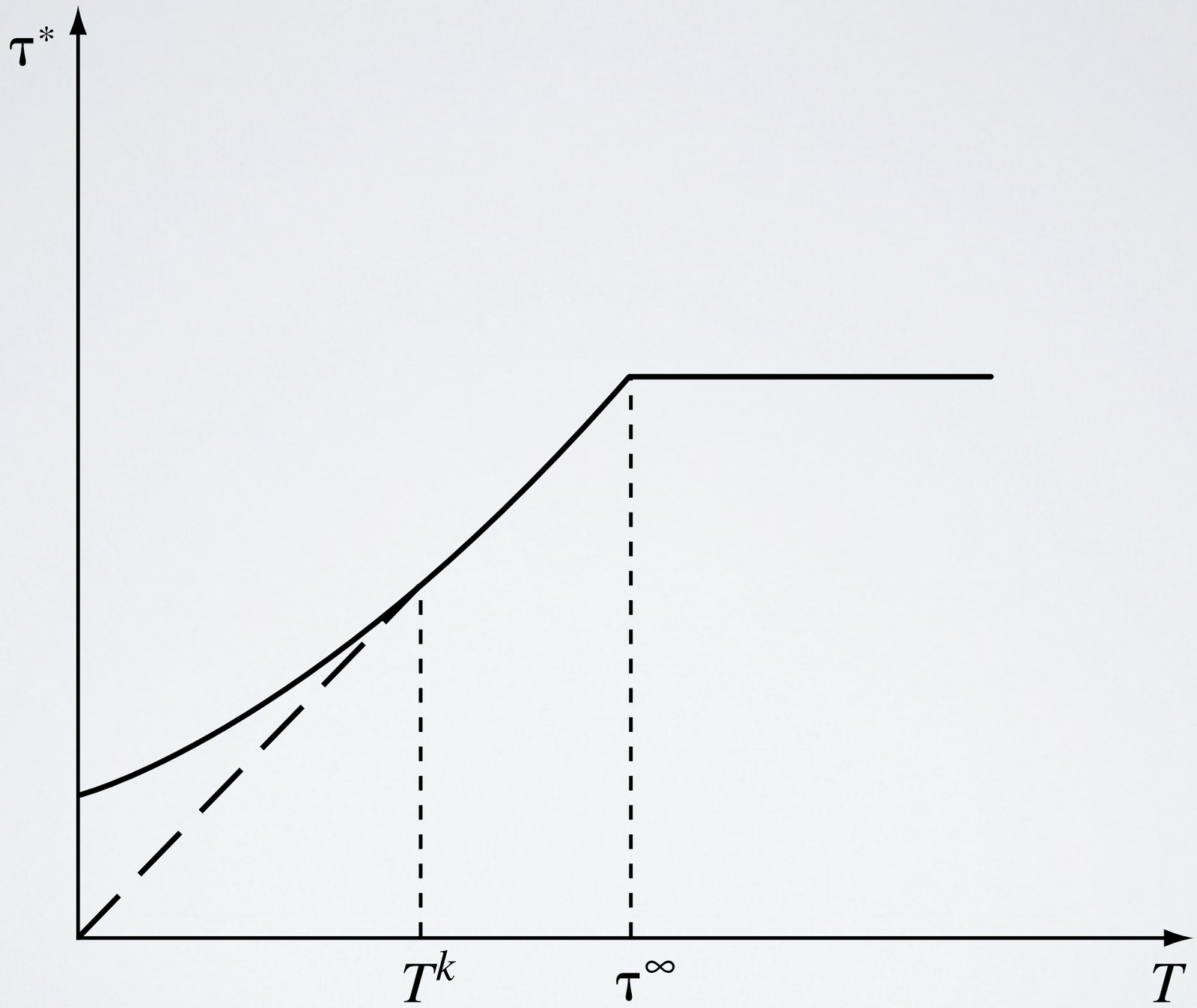
- Vendor's objective function is $V(\tau; T) = C(\tau) + \lambda L(\tau, T)$



VENDOR'S COST

- A point T^k exists such that:
  - If $T>T^k$ then $\tau^*=T$
  - If $T<T^k$ then $\tau^*>T$
- A point τ^∞ exists that represent the optimal patch development time under full secrecy (i.e. $T=t^{end}$)

VENDOR'S COST

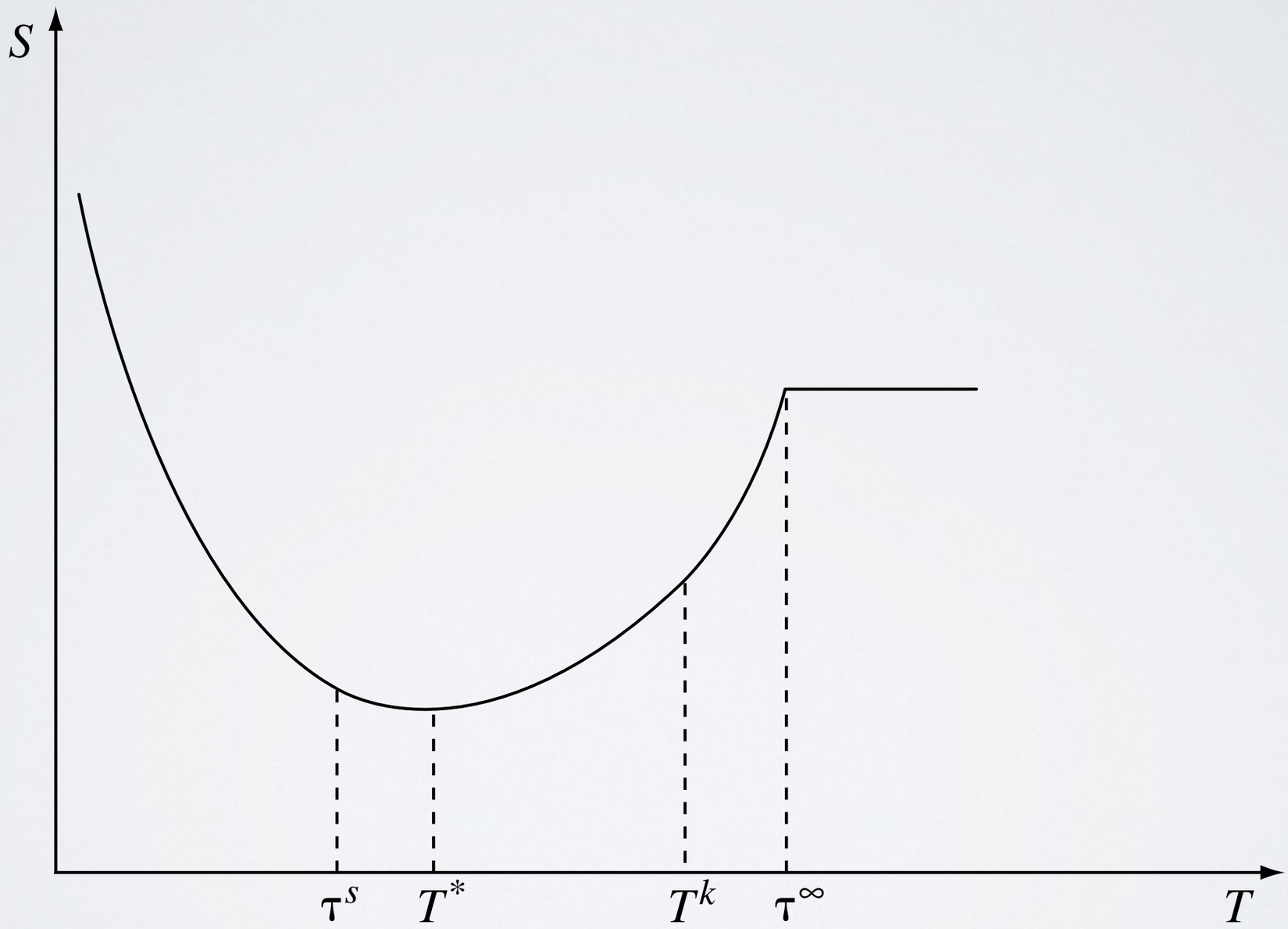


VENDOR'S COST

- T^k is the smallest protected period such that the vendor releases the patch within the protected period.
- If a delay in releasing the patch does not lead to any loss (i.e. $l'(0)=0$) then $T^k=\tau^\infty$
  - the larger $l'(0)$ is the larger will be the distance between T^k and τ^∞
- The lower λ is, the slower is the vendor in implementing the patch
  - This is true only as long as $T<T^k$
  - When $T>T^k$, the vendor optimal patch implementation time is no more dependent on λ

VENDOR'S COST

- The global societal cost is $S(T) = C(\tau(T)) + L(\tau(T), T)$



BIBLIOGRAPHY

[1] A. Arora, R. Telang and H. Xu. Optimal Policy for Software Vulnerability Disclosure. Journal of Management Science, Vo. 54, No. 4, 2008.