Intrusion Recovery for Database-backed Web Applications

Ramesh Chandra, Taesoo Kim, Meelap Shah,
Neha Narula, Nickolai Zeldovich

MIT CSAIL

Web applications routinely compromised

Teen uses worm to boo... ×

www.computerworld.com/s/article/105484/Teen_uses_worm_to_boost_ratings_on_MySpace.com

Home > Security > Malware and Vulnerabilities

Teen uses worm to boost ratings on MySpace.com

It did little damage but could point to broader vulnerabilities, says a security expert

By Eric Lai

October 17, 2005 12:00 PM ET

Recommended (6) Like

Computerworld - Using a self-propagating worm that exploits a scripting vulnerability common to most dynamic Web sites, a Los Angeles teenager made himself the most popular member of community Web site MySpace.com earlier this month. While the attack caused little damage, the technique could be used to destroy Web site data or steal private information -- even from enterprise users behind protected networks, according to an security services firm.

The unknown 19-year-old, who used the name "Samy," put a small bit of code in his user profile on MySpace, a 32-million-member site, most of whom are under age 30. Whenever Samy's profile was viewed, the code was executed in the background, adding Samy to the viewer's list of friends and writing at the bottom of their profile, "... and Samy is my hero."

"This is an attack on the users of the Web site, using the Web site itself,"

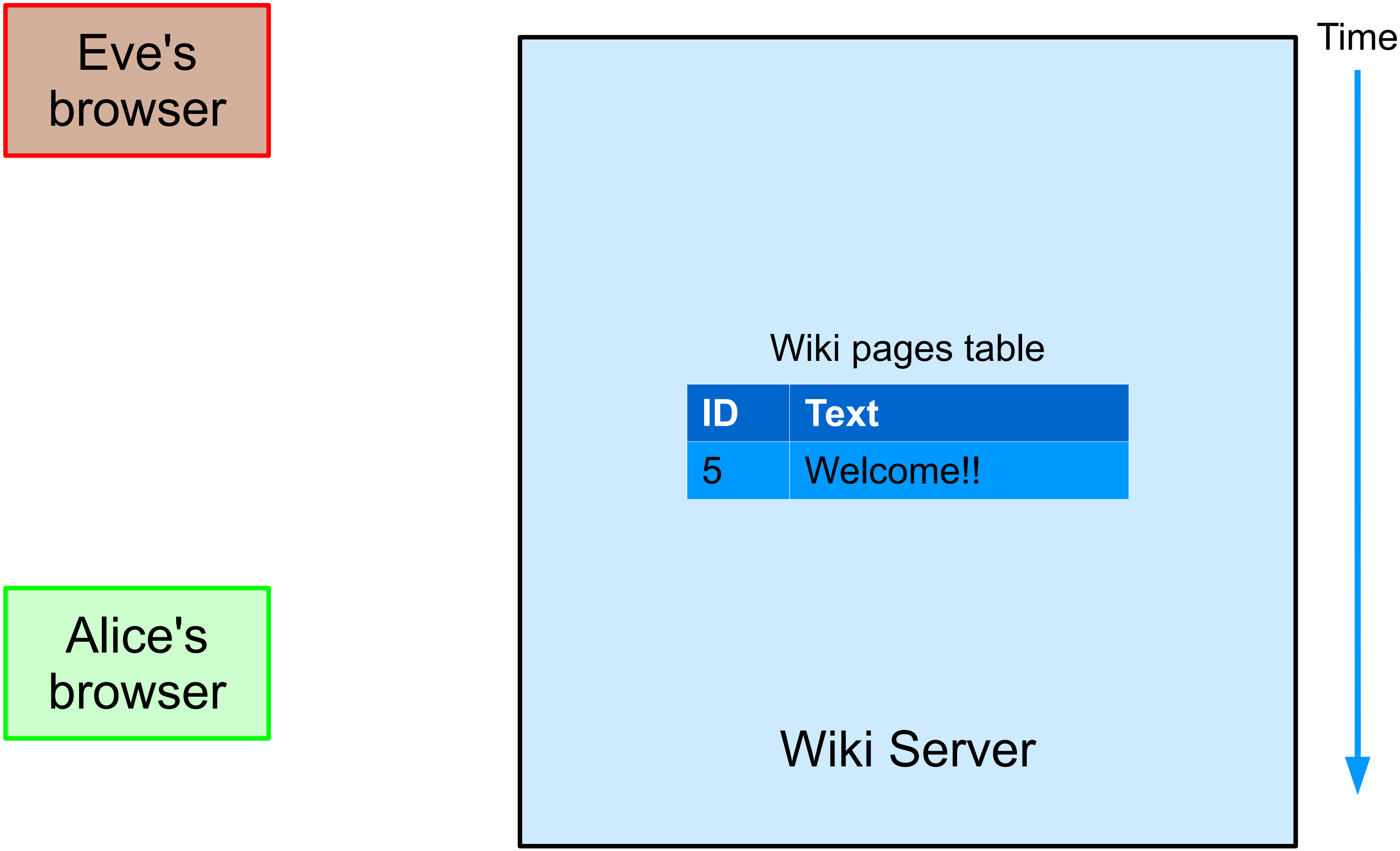
Web applications routinely compromised

Web applications routinely compromised

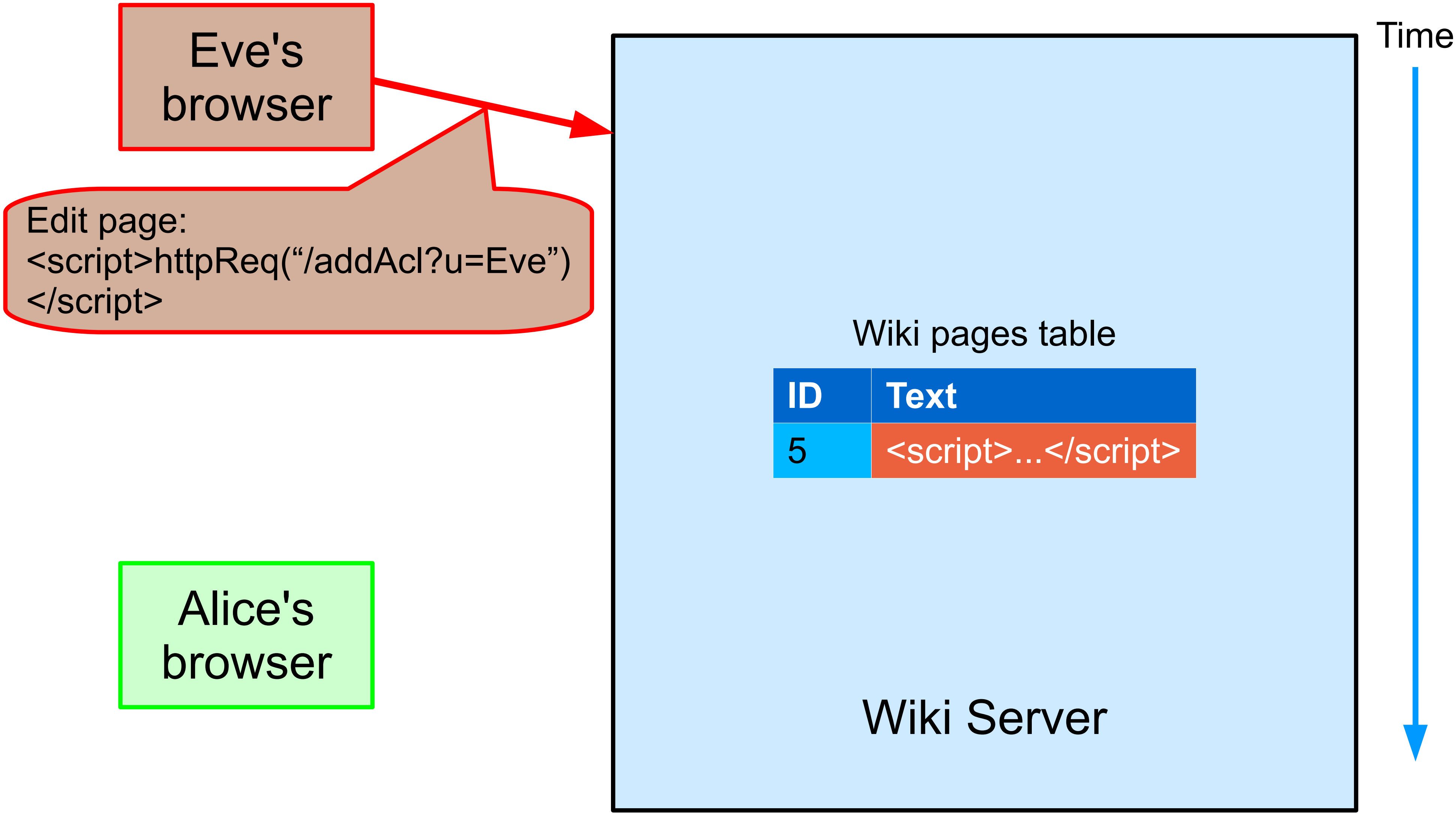
Recovering integrity is important

- Preventing intrusions is important, but compromises will still happen
  - Vulnerabilities are common, and new bugs are constantly being found [CVE]
    - 3-4 new vulnerabilities found *per day,* on average for the past 4 years
  - Administrators misconfigure policies, settings
- This talk: recovering integrity after attack

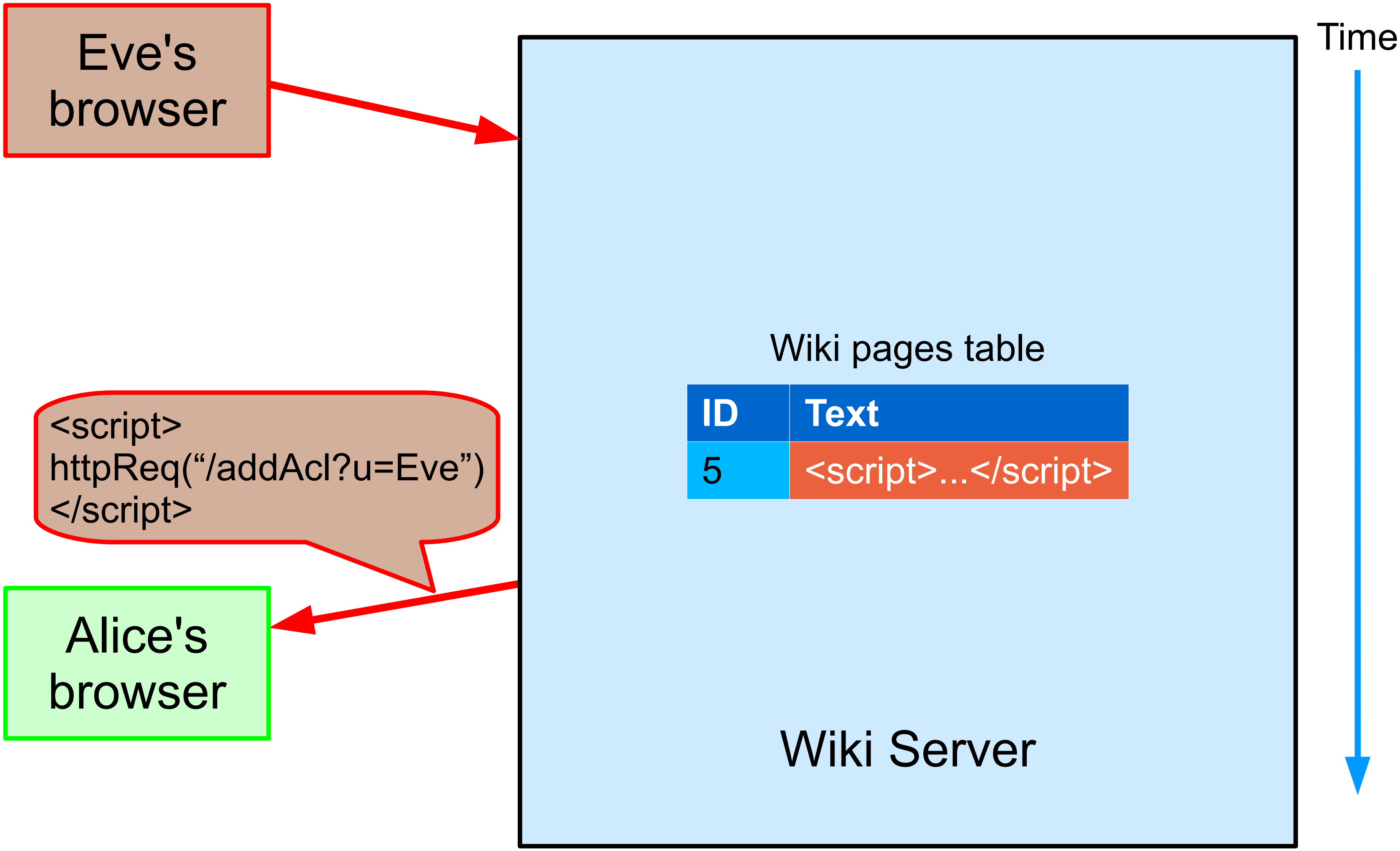
Cross-site scripting (XSS) bugs
(simplified)



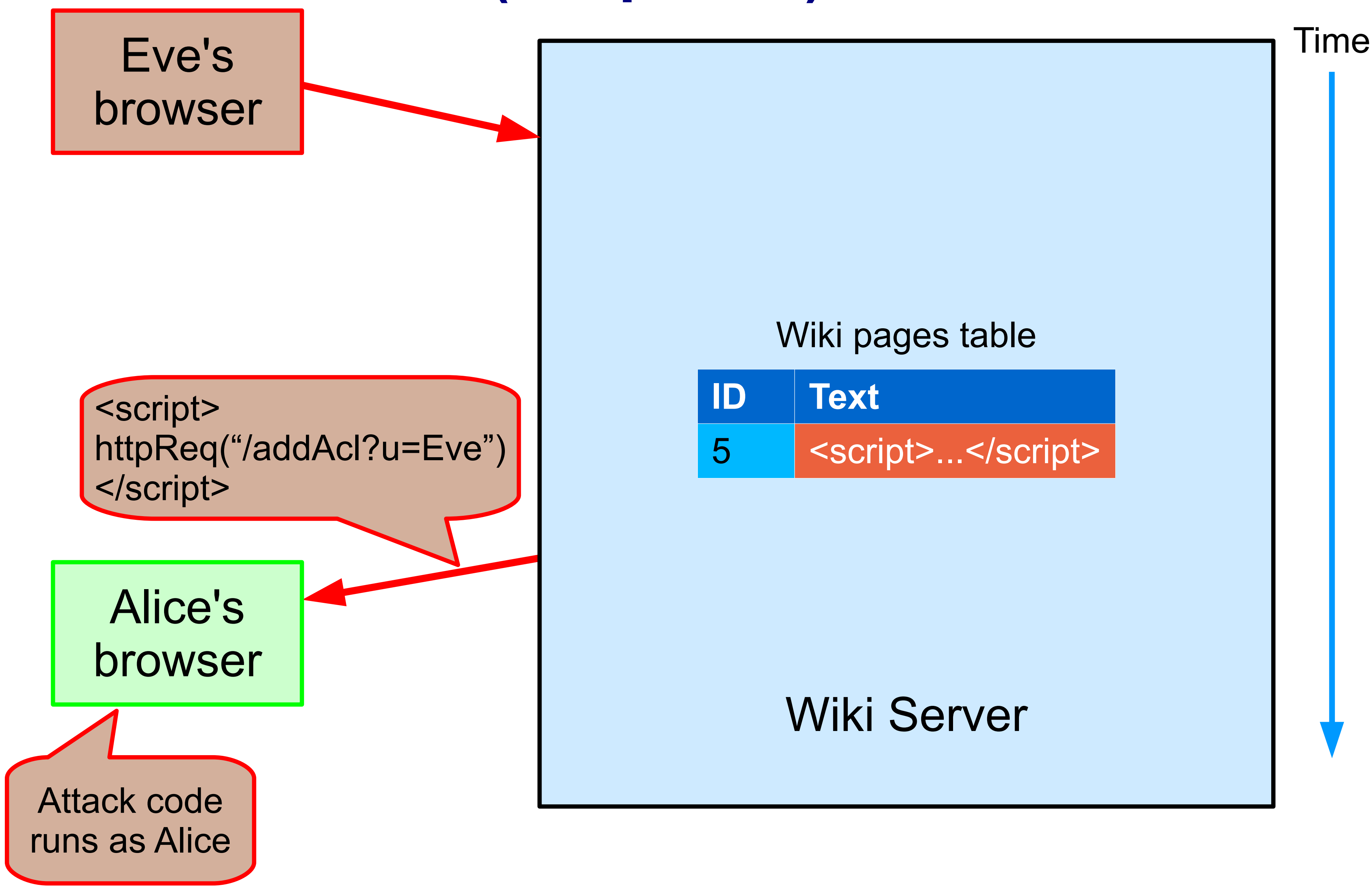
Cross-site scripting (XSS) bugs
(simplified)



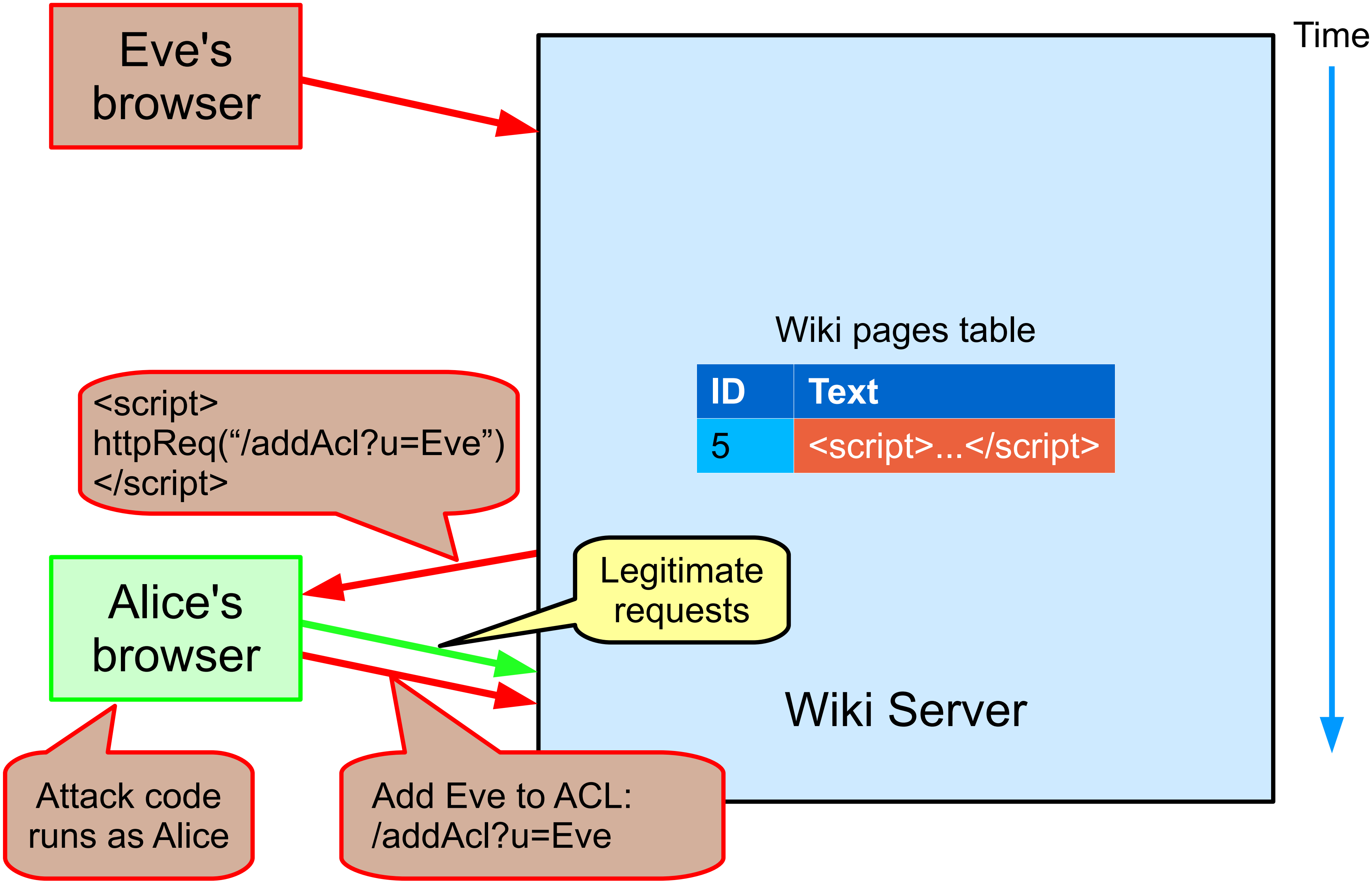
Cross-site scripting (XSS) bugs
(simplified)



Cross-site scripting (XSS) bugs
(simplified)



Cross-site scripting (XSS) bugs
(simplified)



Recovering web application integrity is hard

- Web apps store data in shared data store
  - Multiple users data is commingled
  - Users access each other's data
- Makes recovering from attack complicated:
  - Attack propagates across users
  - Attack can arbitrarily corrupt user data
    - e.g., financial information
  - Attack can install backdoors
    - e.g., modify ACLs, install Google apps scripts

Limited recovery tools

- Backup-and-restore tools
  - Attack may be detected days or weeks later
  - Restoring from backup discards *all* users' changes
- Manual recovery
  - Admin spends days or weeks tracking attack's effects
  - Admin could miss a subtle backdoor or corruption

Contributions

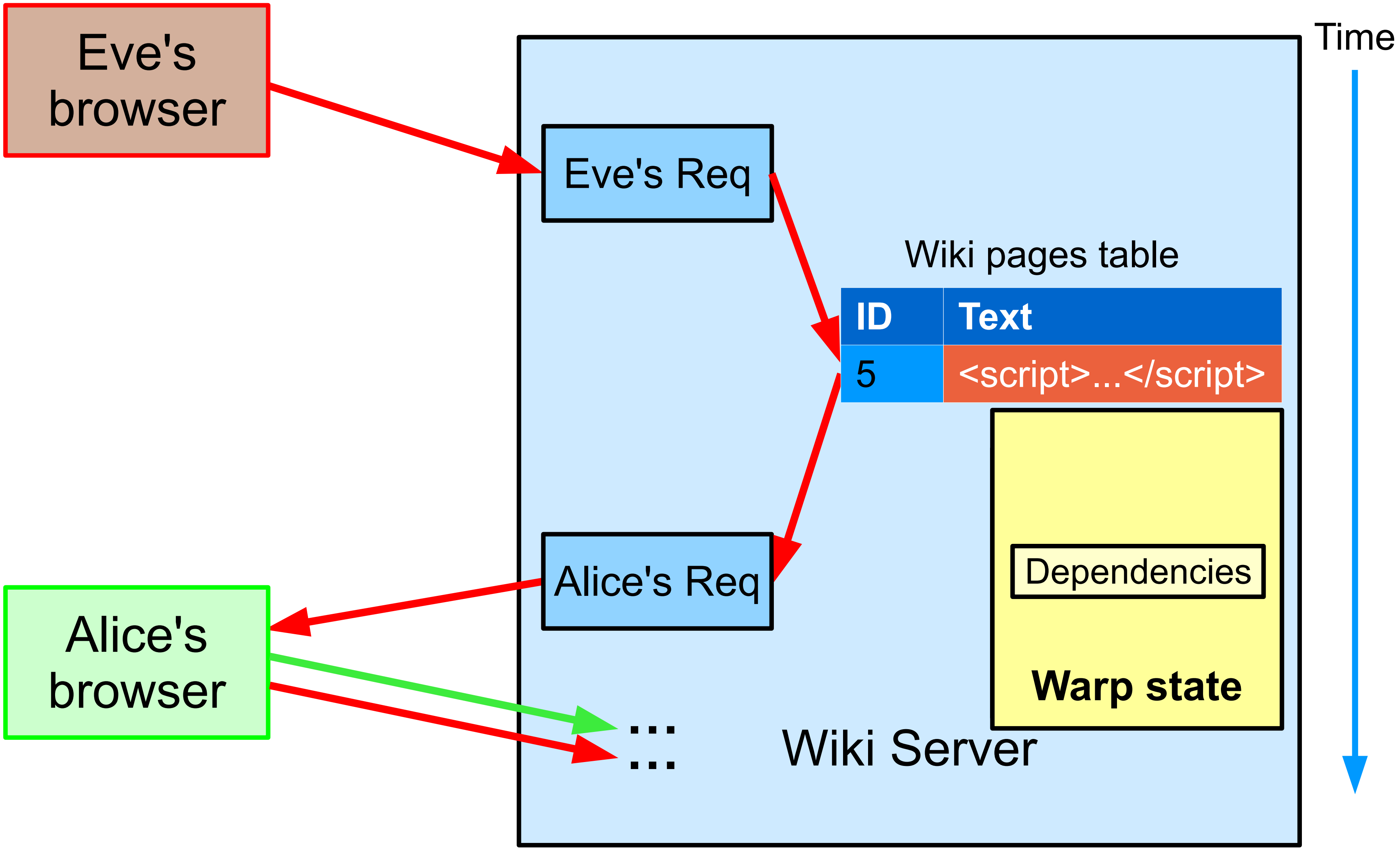
- Warp: web application intrusion recovery
  - Undoes effects of attack but keeps legitimate changes
  - Works for real applications: MediaWiki, Drupal, Gallery2

- Key ideas:
  - *Retroactive patching* eliminates need to pinpoint attack
  - *Time-travel DB* precisely tracks causal effects
  - *DOM-level replay* preserves users' intended changes

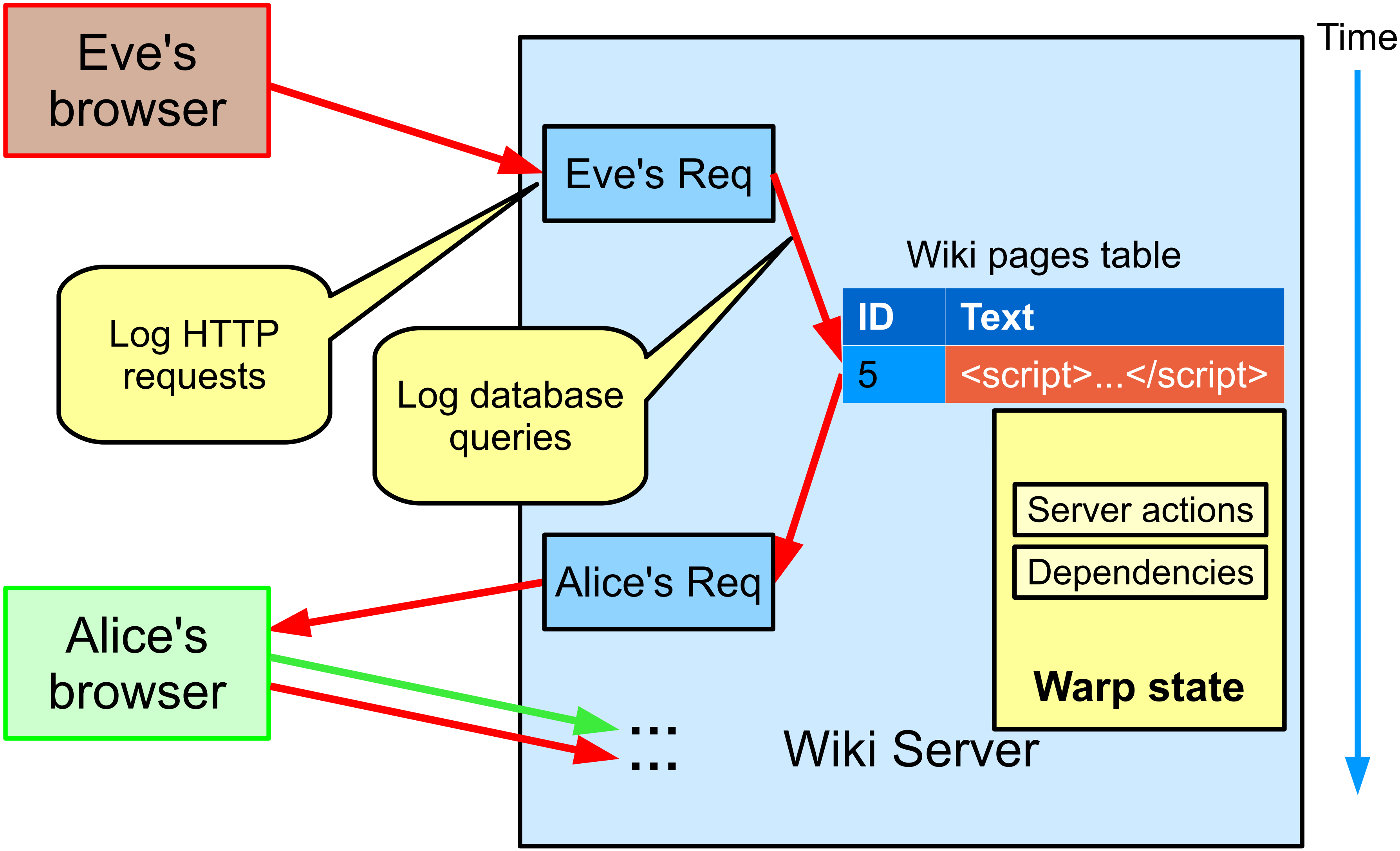
High-level approach: rollback and re-execute

- Normal execution
  - Record *actions* in system to a log
  - Record causal dependencies between actions
  - Record checkpoints system state
- Repair
  - Identify attack action
  - Rollback *affected* system state to before attack
  - Replay all affected actions except attack action

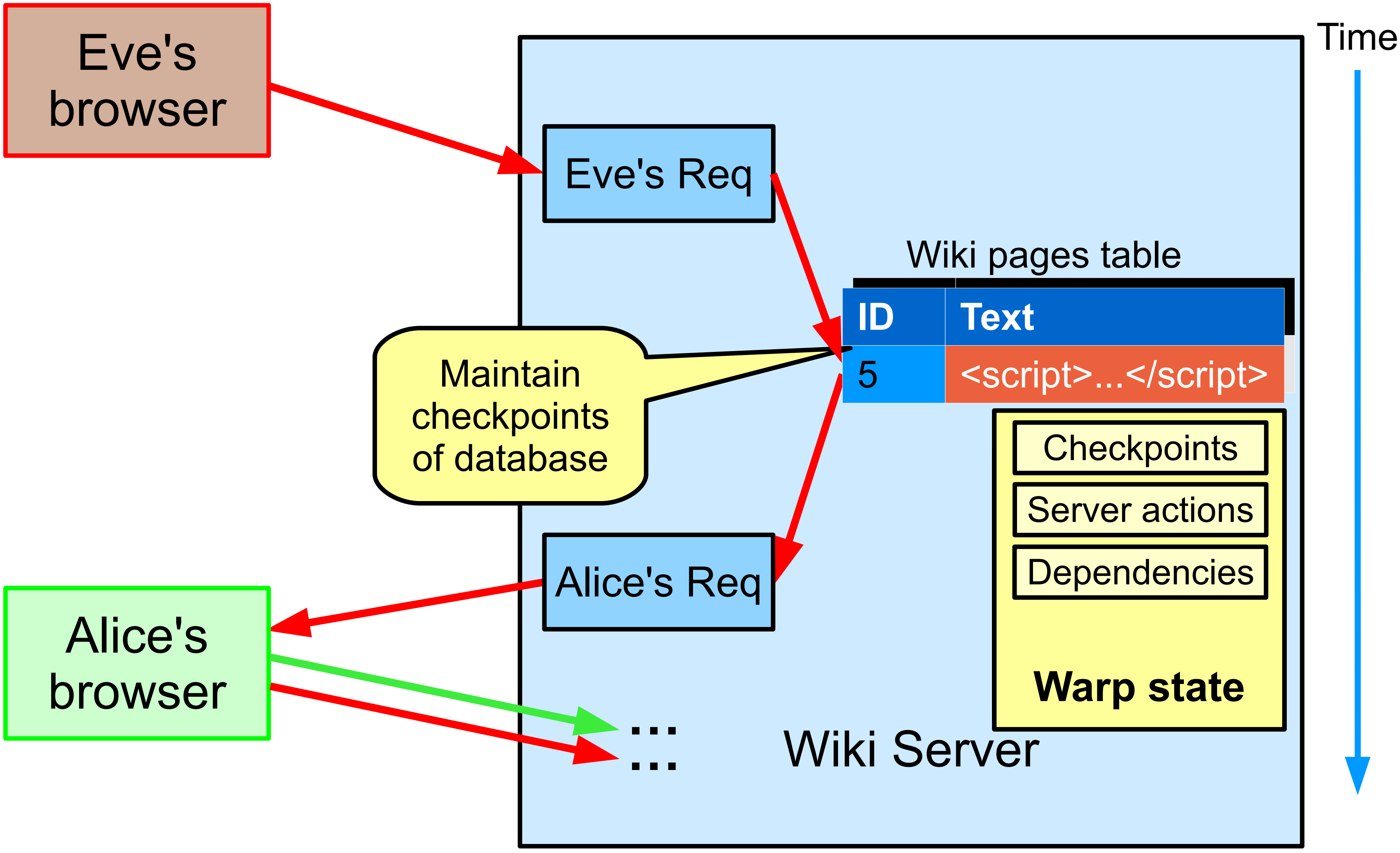
Normal execution



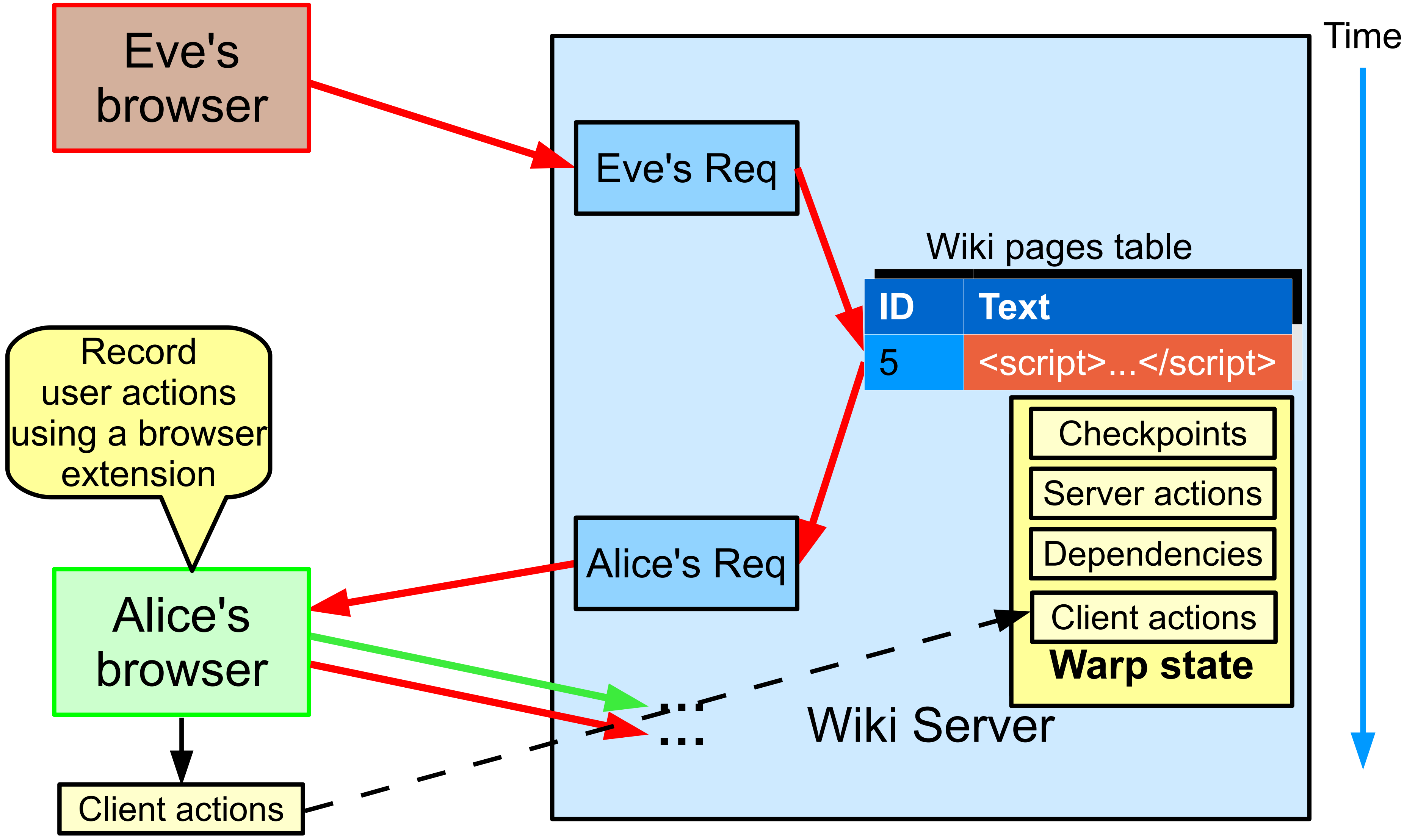
Normal execution



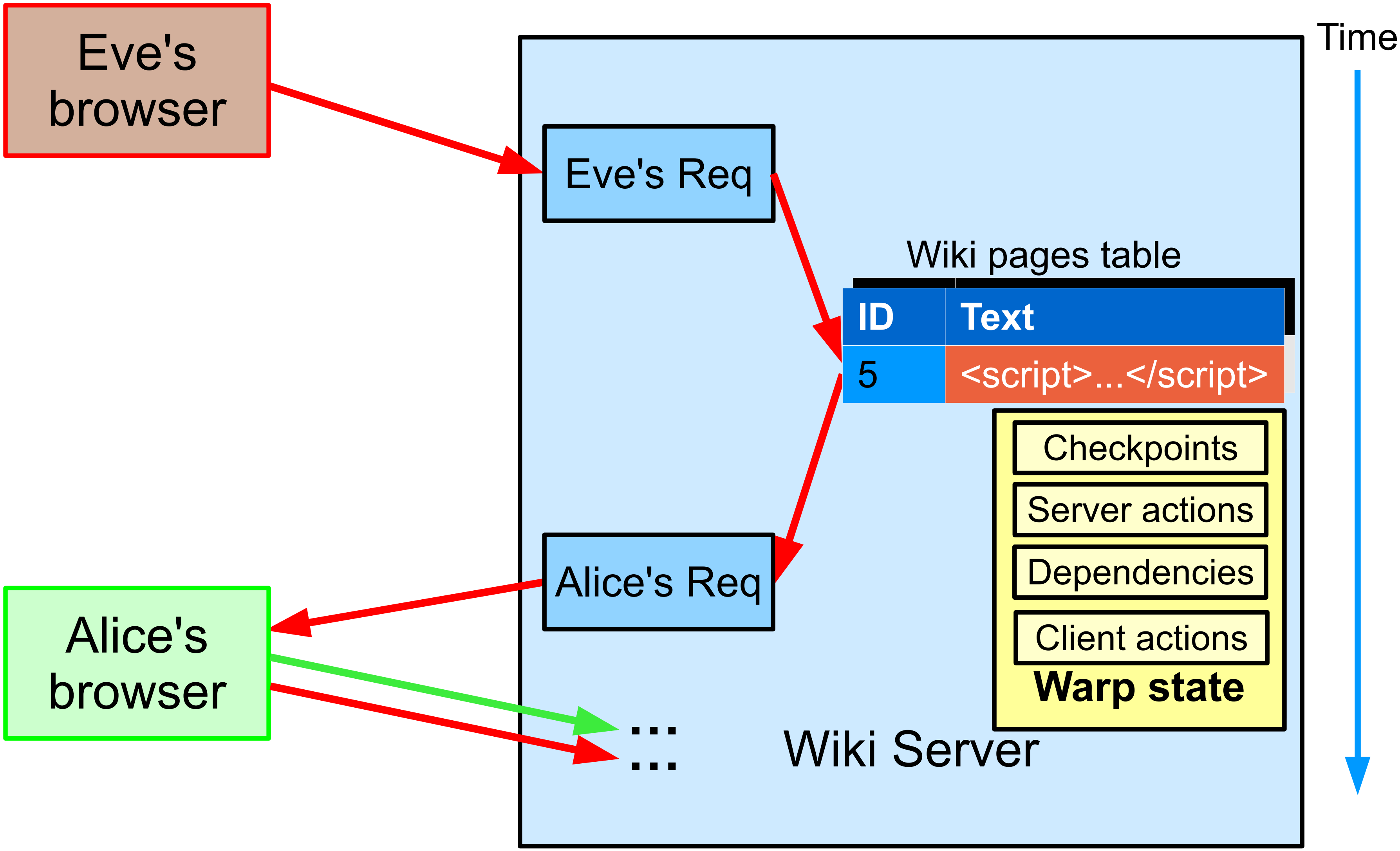
Normal execution



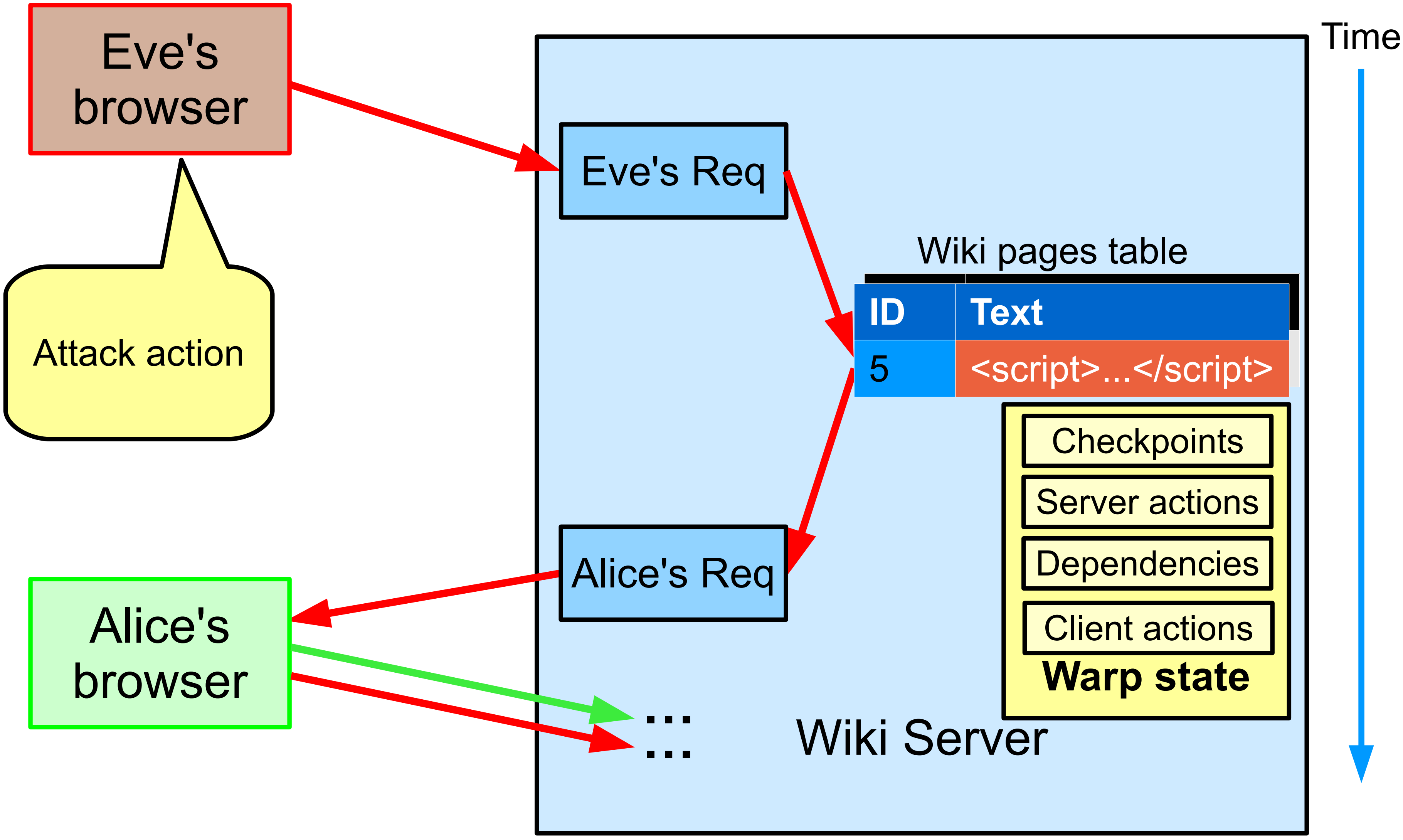
Normal execution



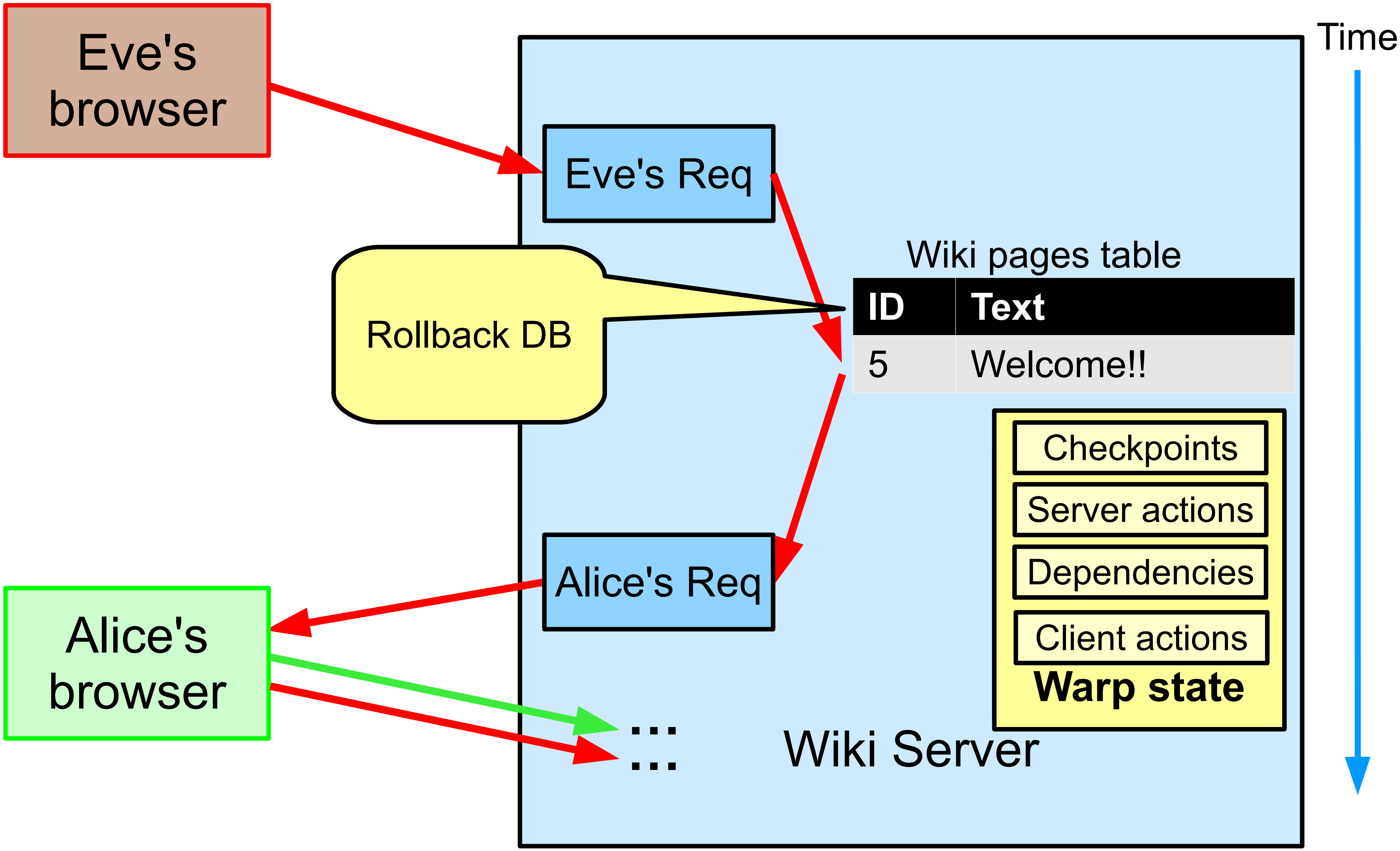
Strawman repair



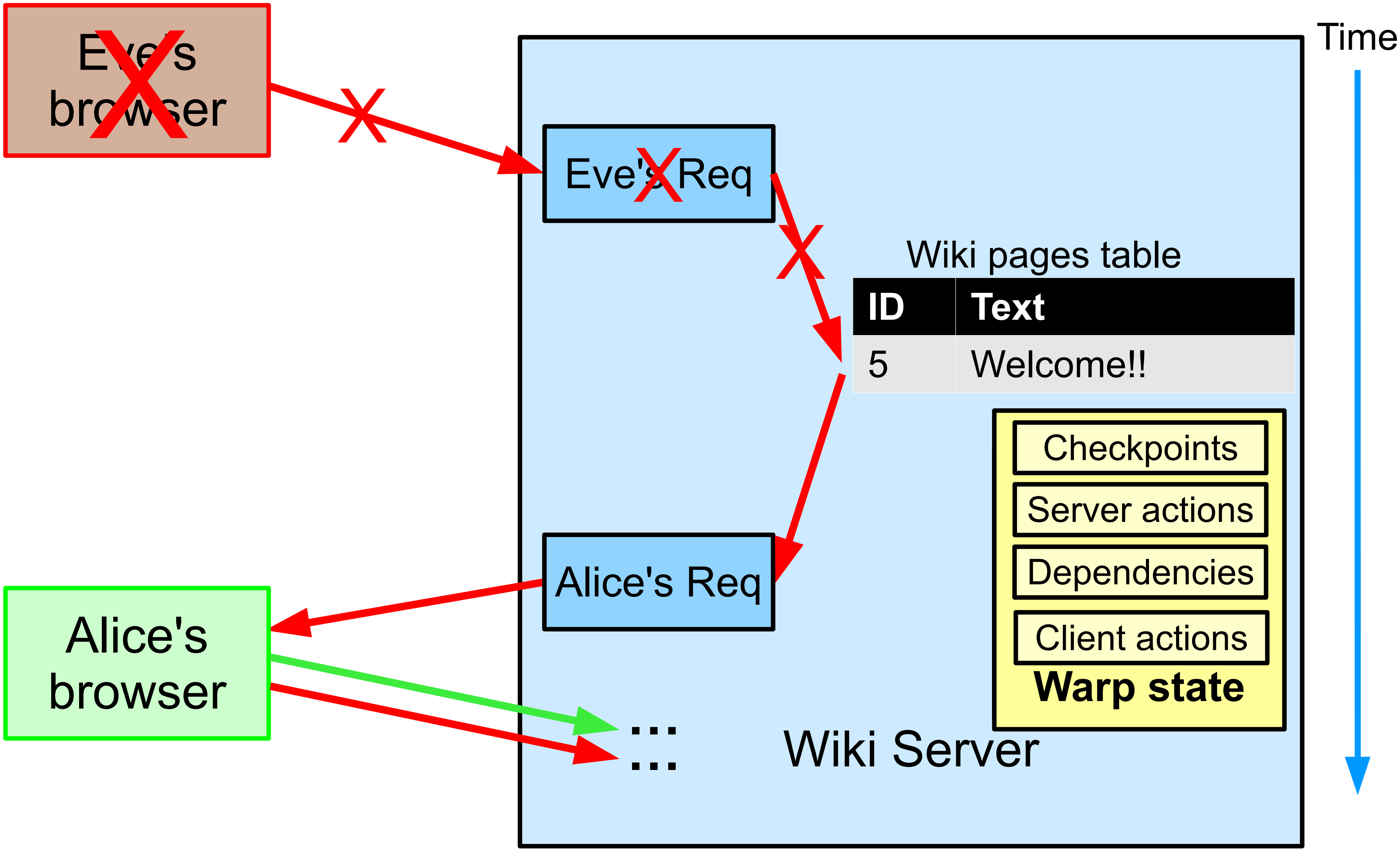
Repair: identify attack



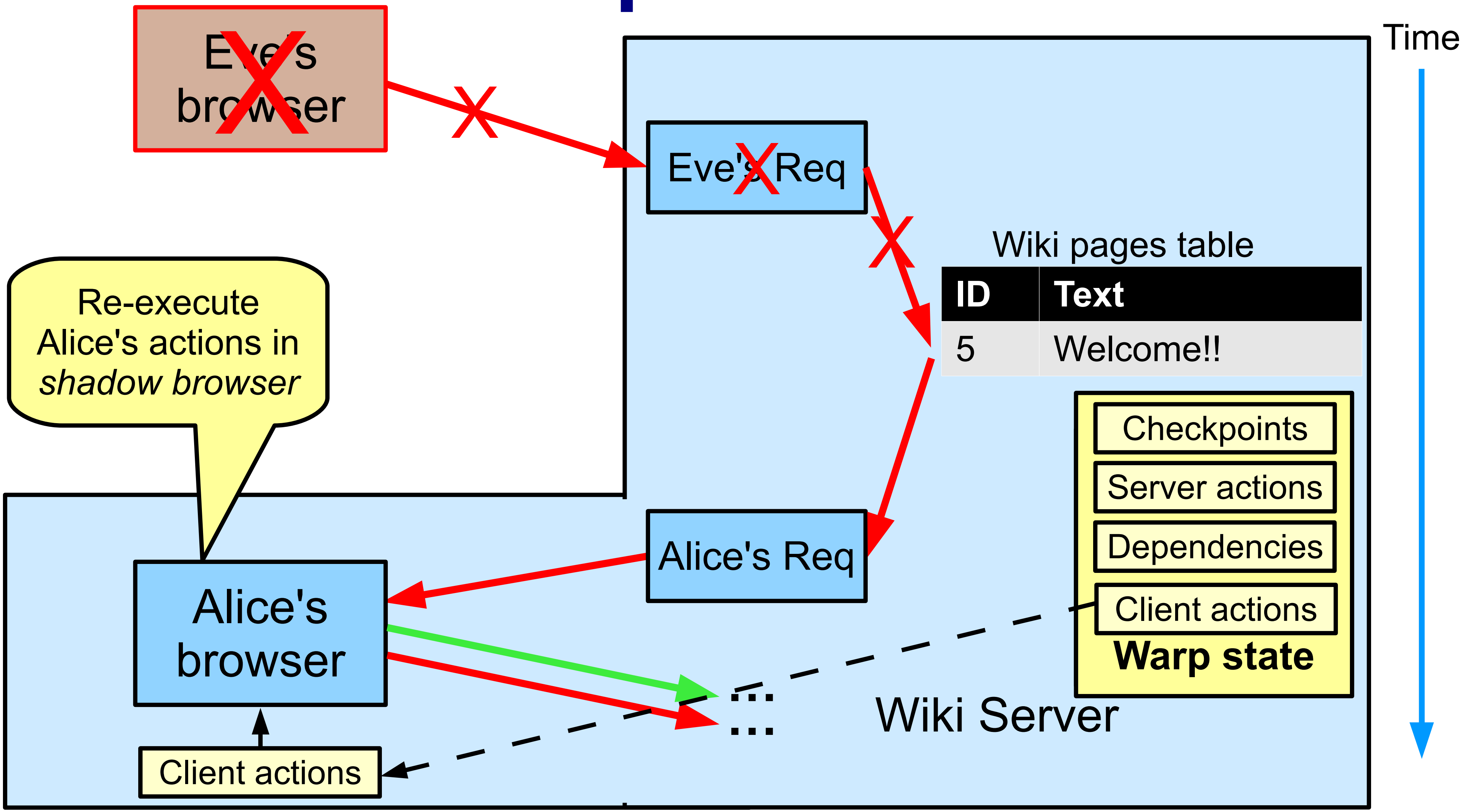
Repair: rollback to before attack



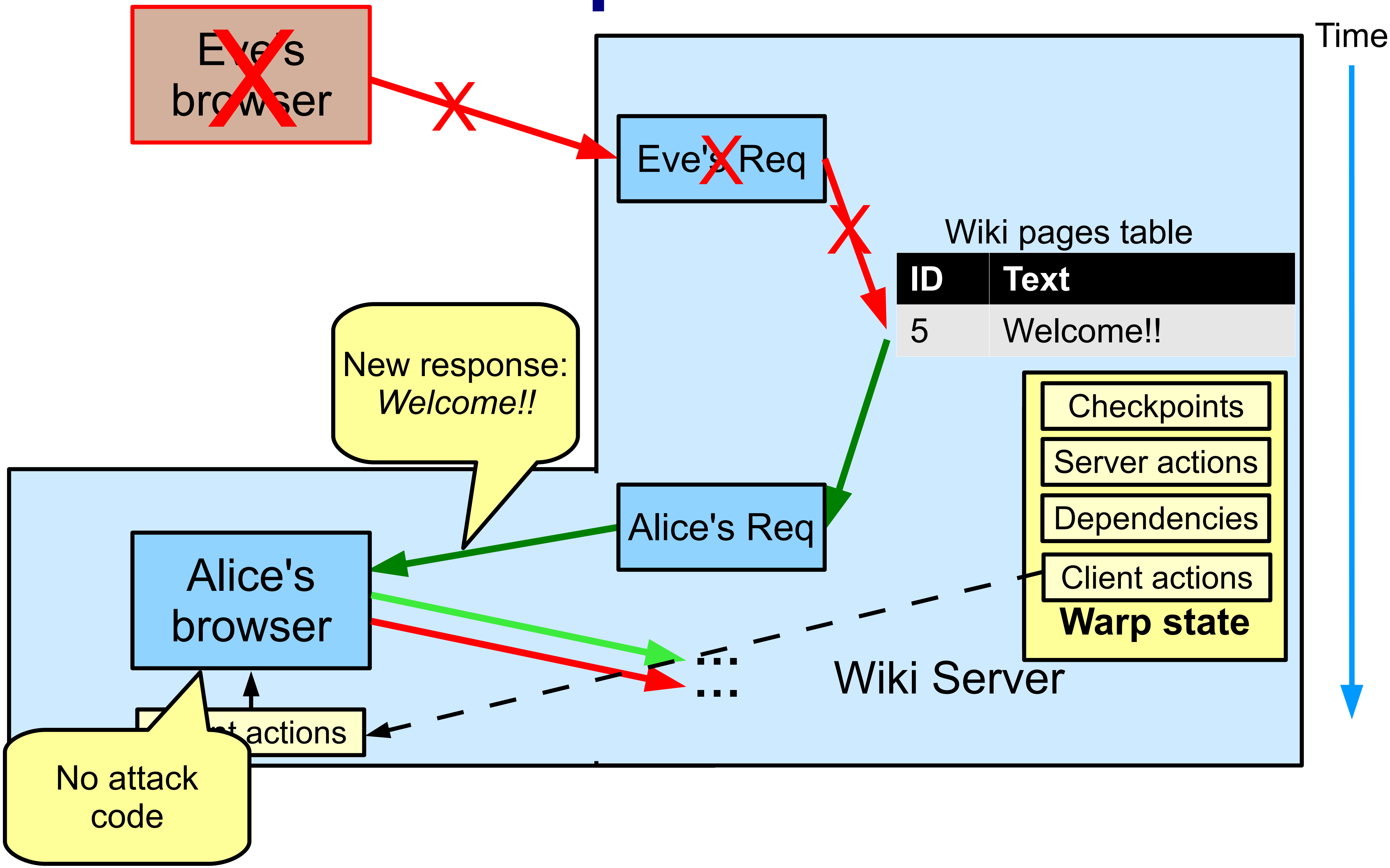
Repair: skip attack action



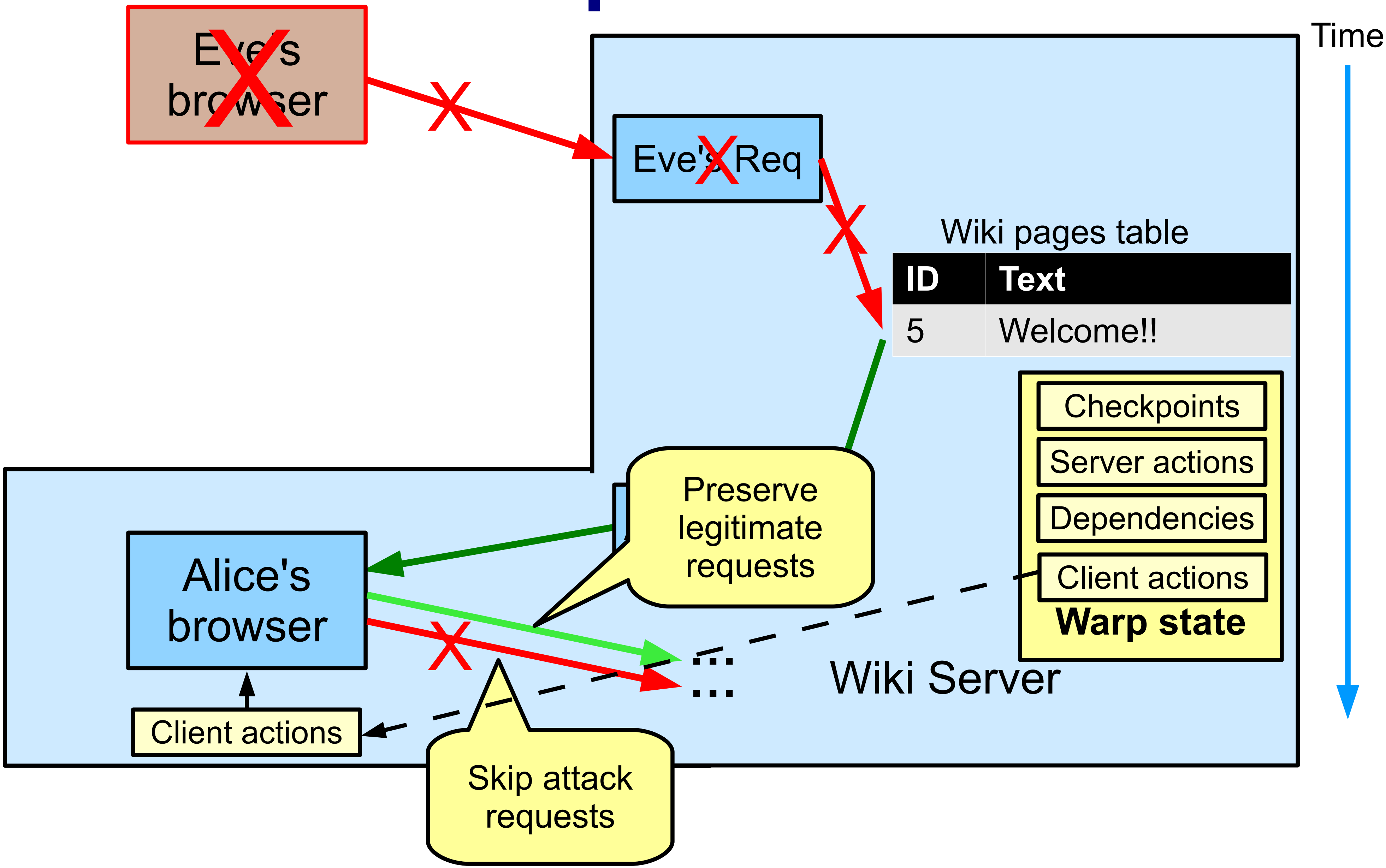
Repair: re-execute subsequent actions



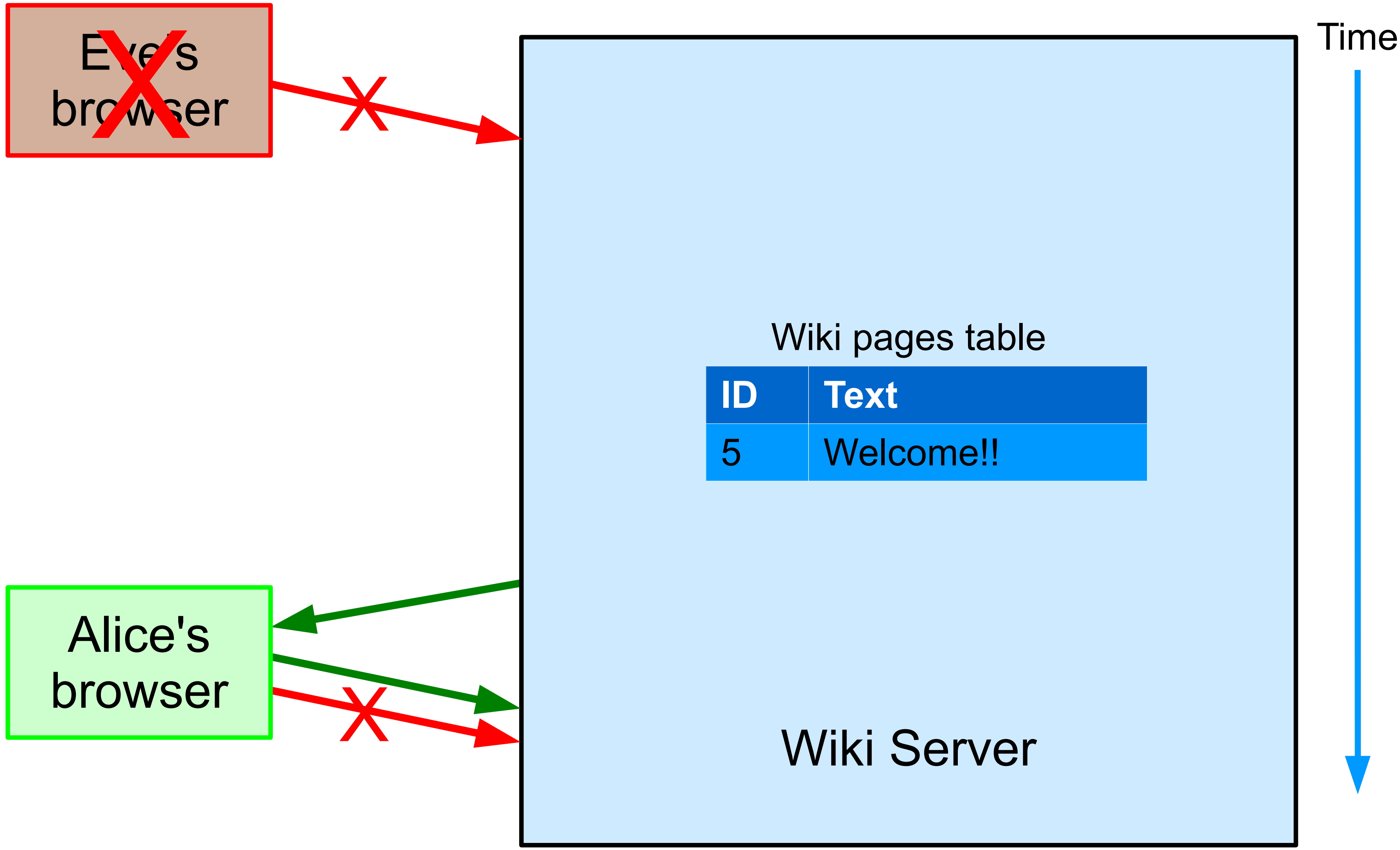
Repair: re-execute subsequent actions



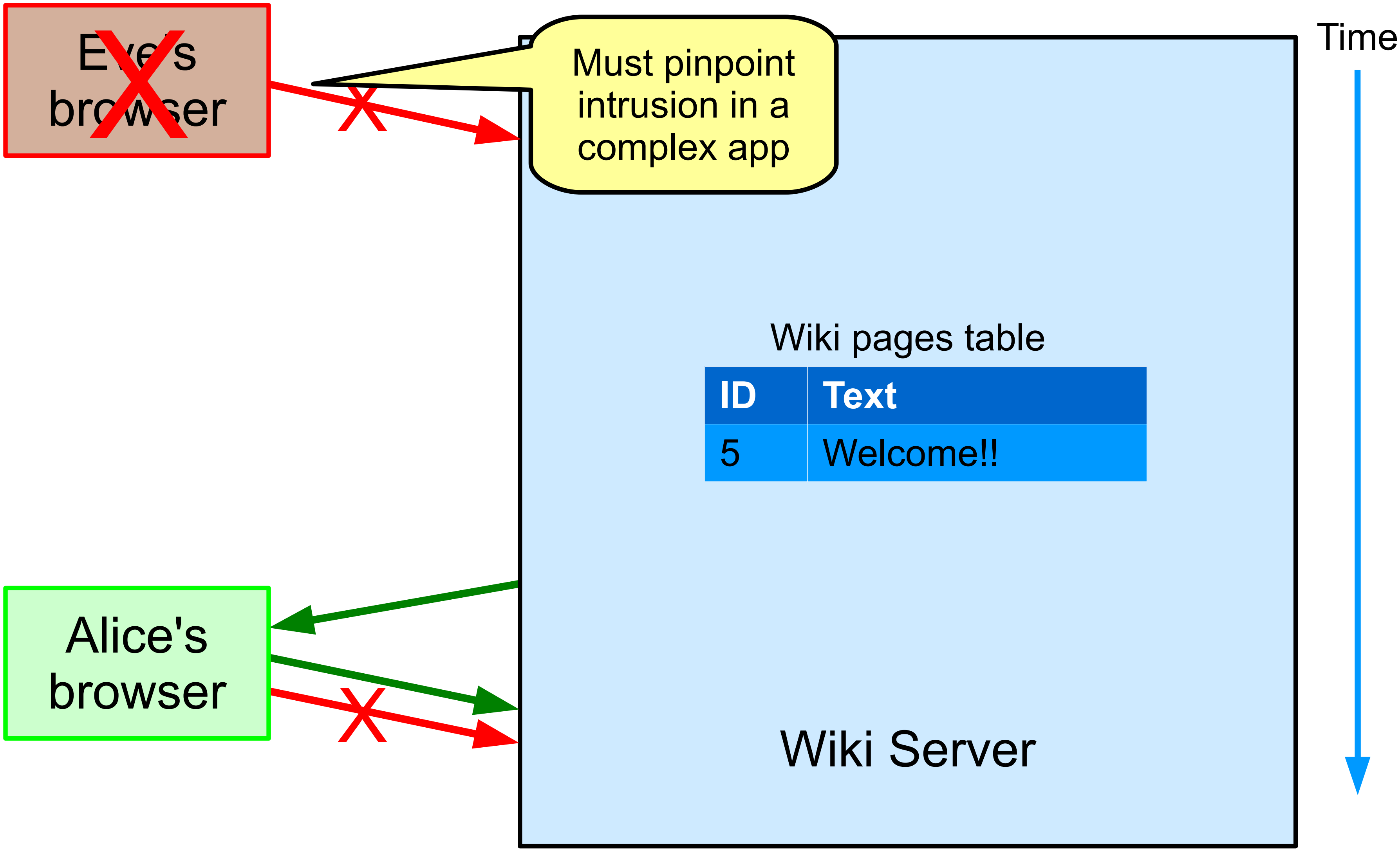
Repair: re-execute subsequent actions



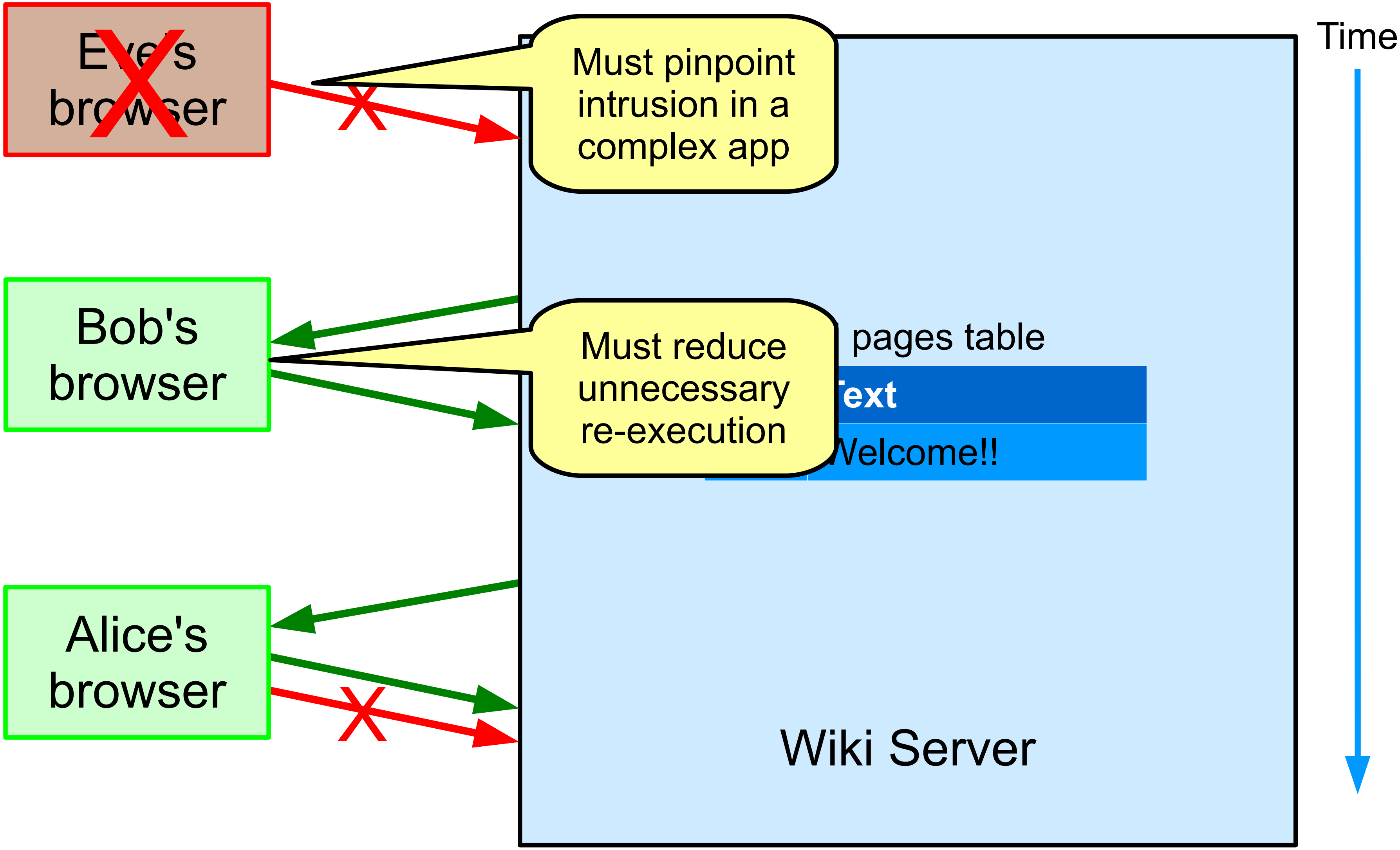
Challenges to intrusion recovery



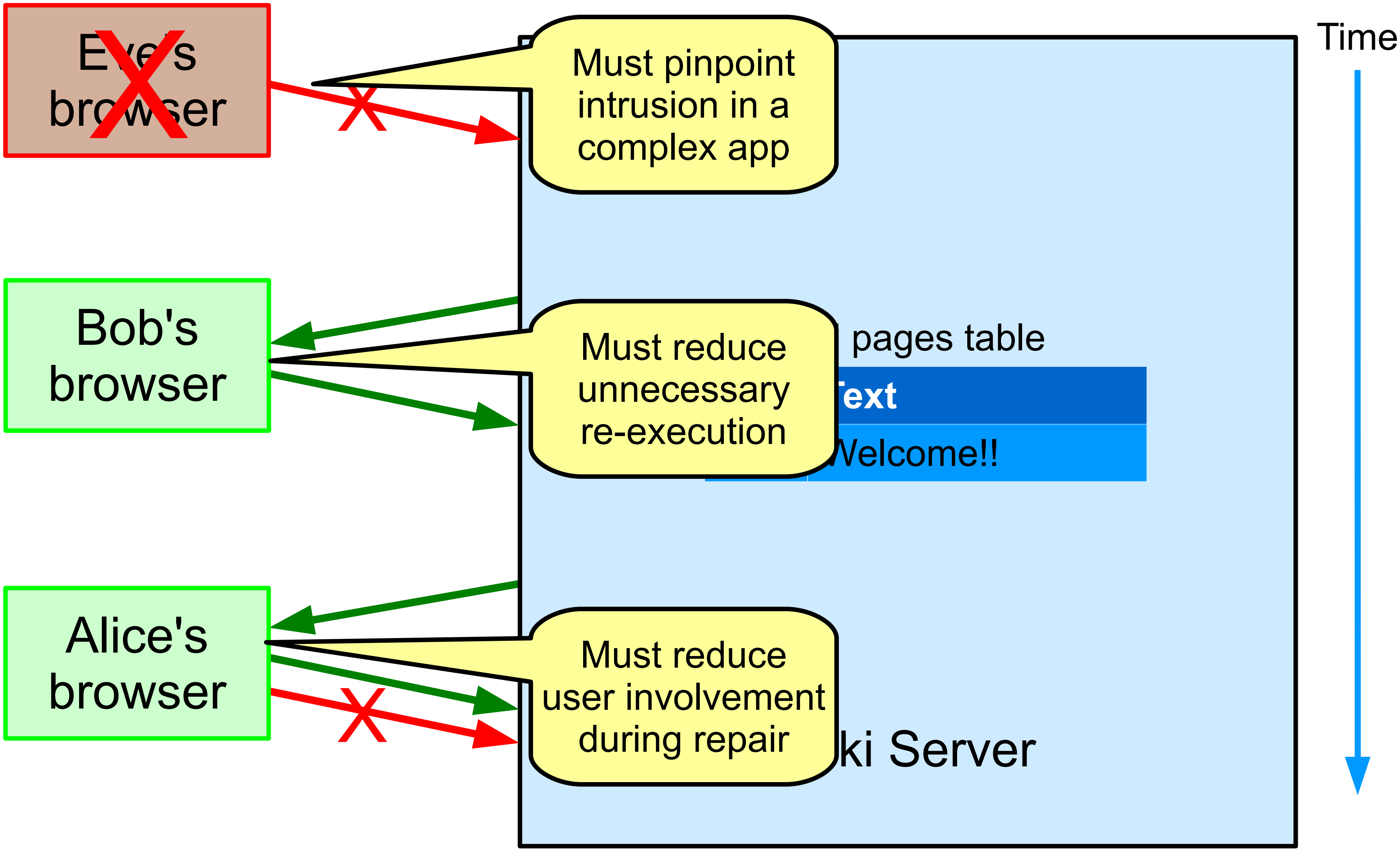
Challenges to intrusion recovery



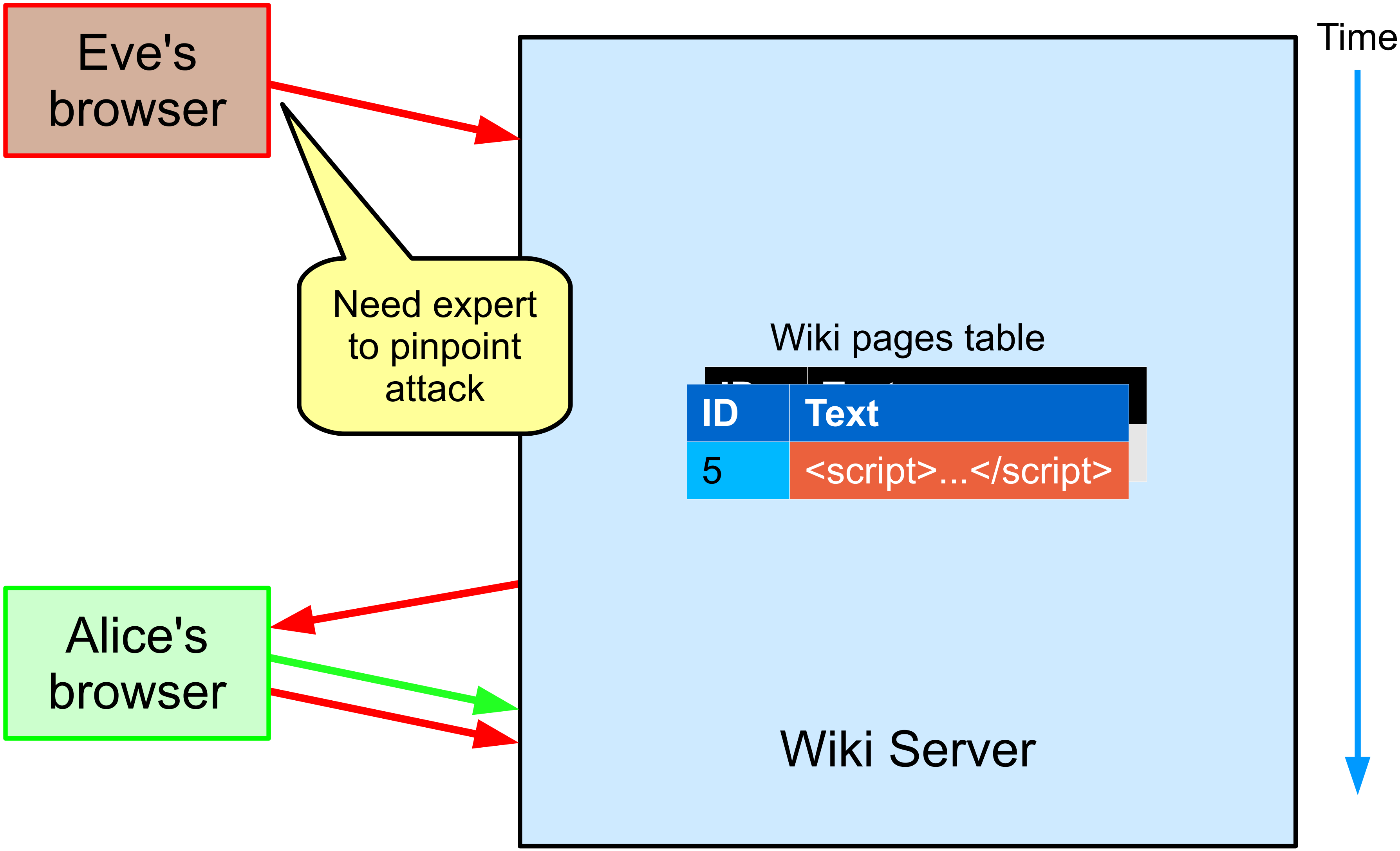
Challenges to intrusion recovery



Challenges to intrusion recovery



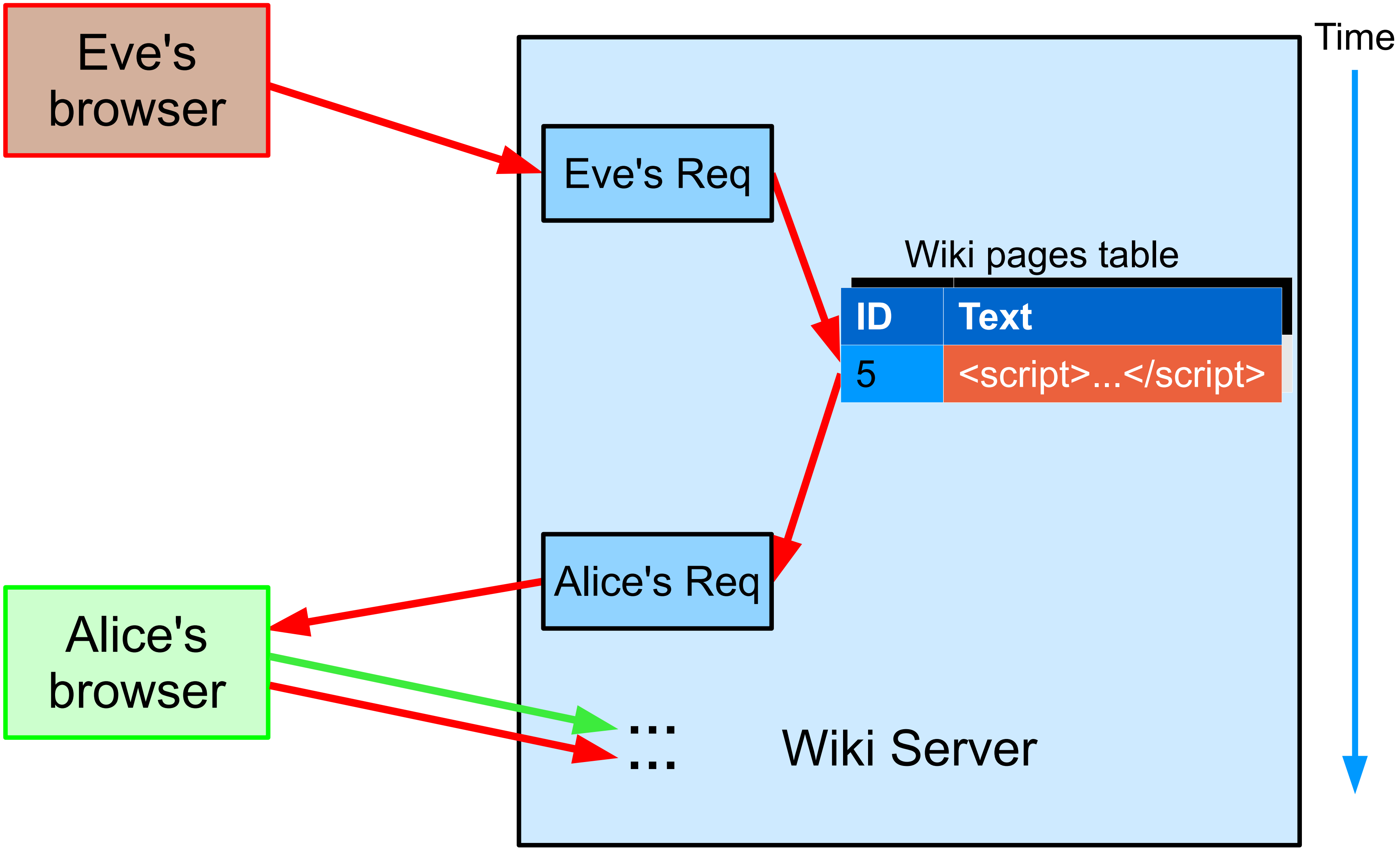
Challenge 1: intrusion detection is difficult



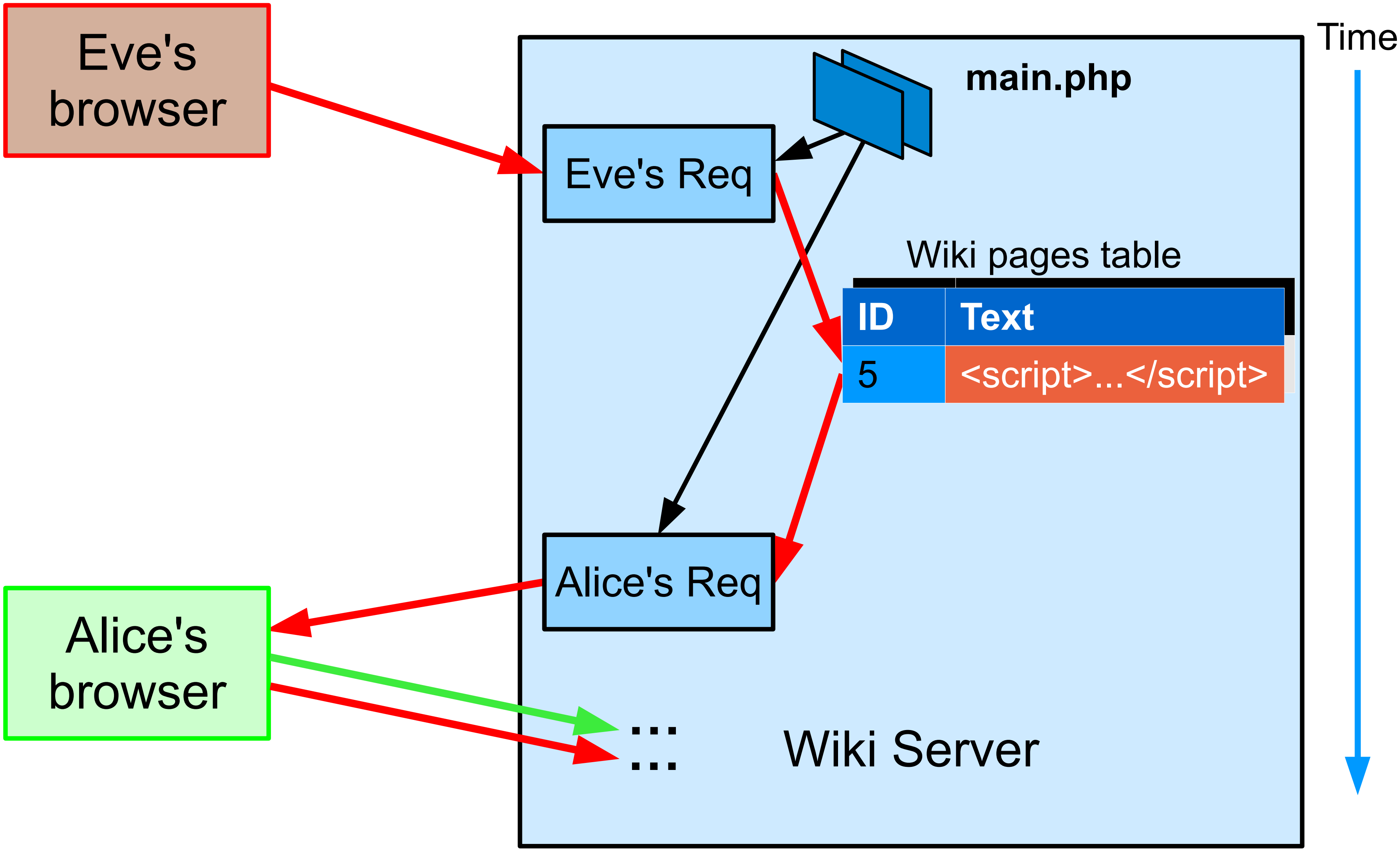
Idea: retroactive patching

- Key observation: patch renders attacks harmless
- Approach:
  - Retroactively apply security patches back in time
  - Re-execute all affected requests

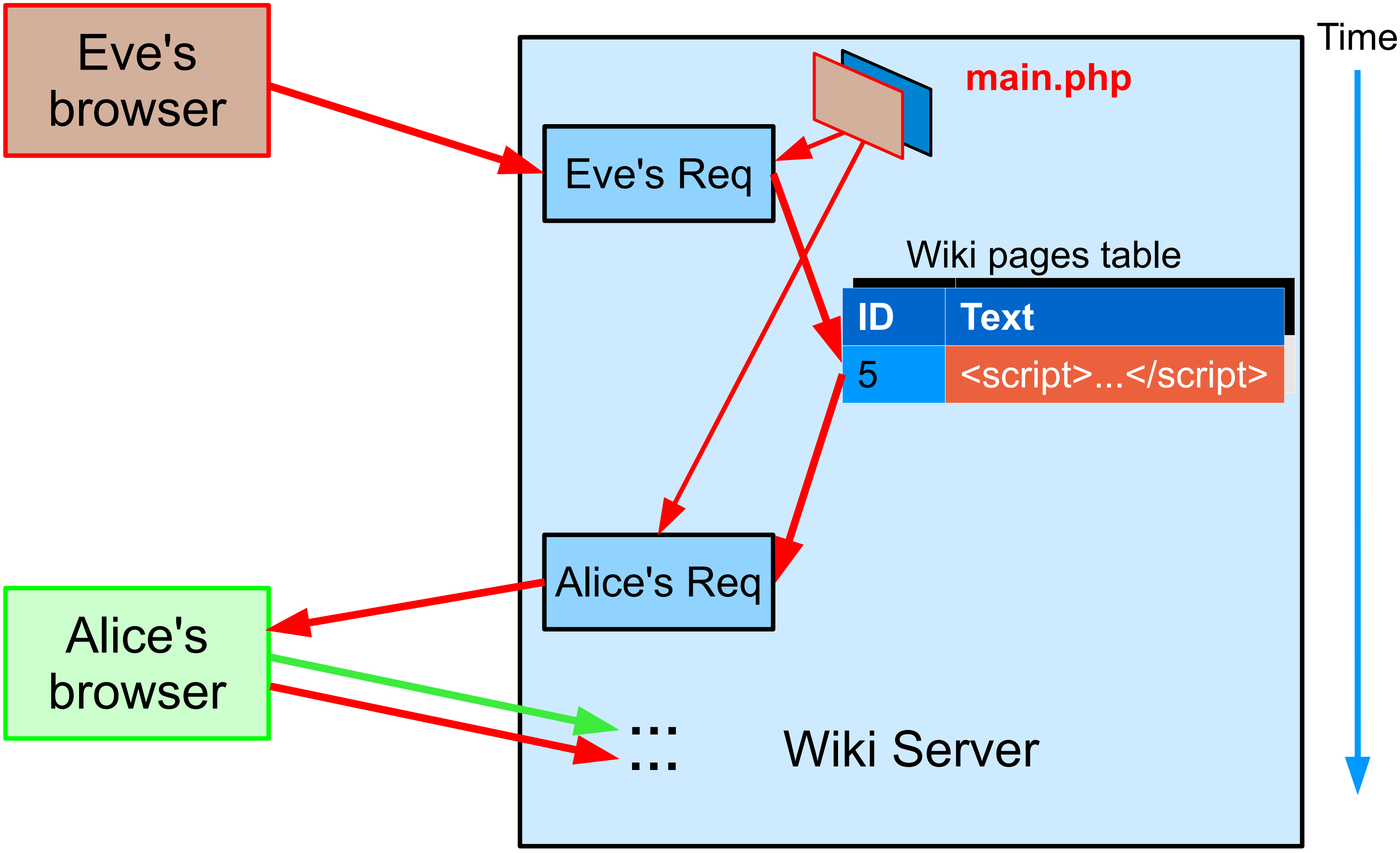
Retroactive patching



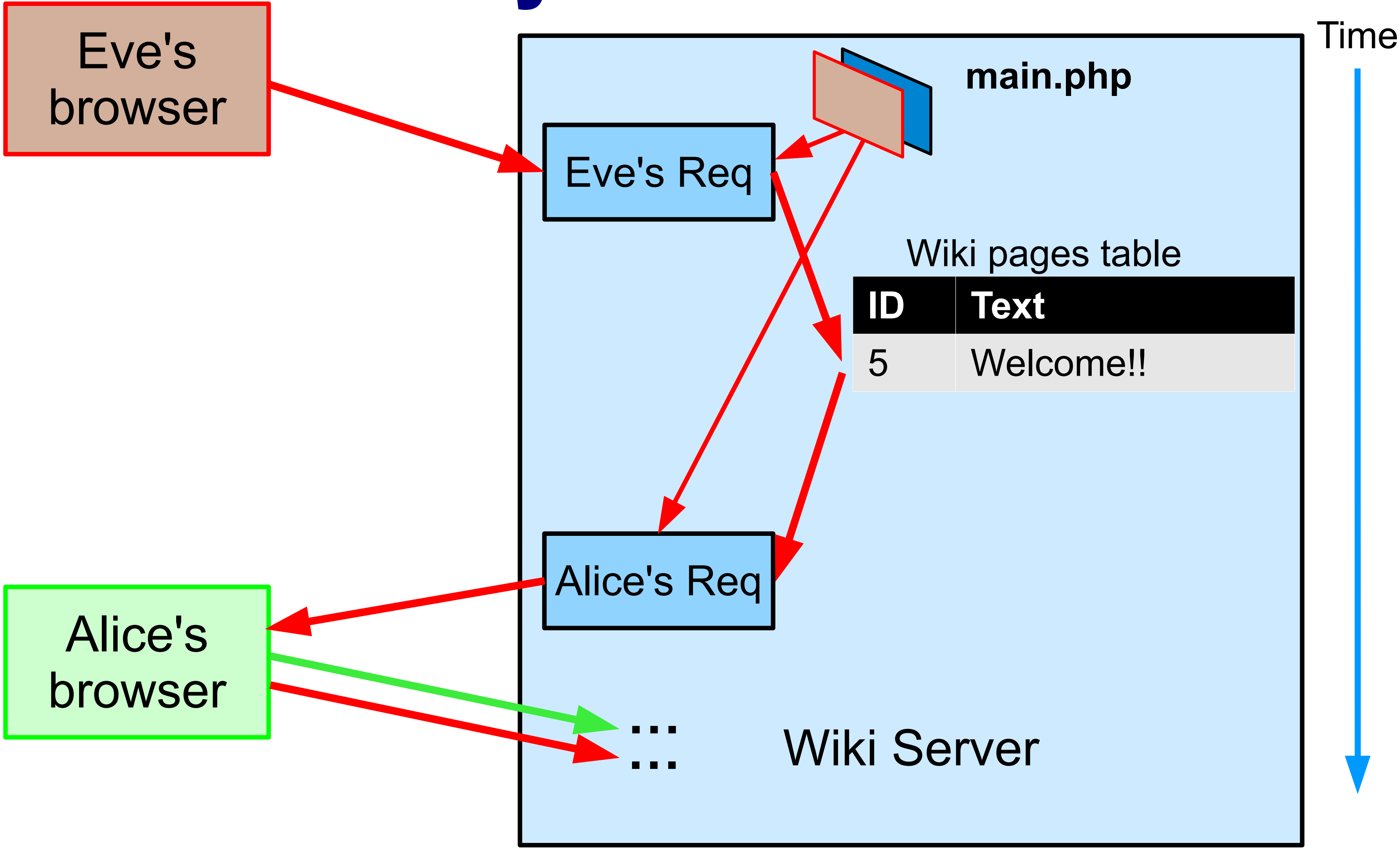
Retroactive patching: normal execution



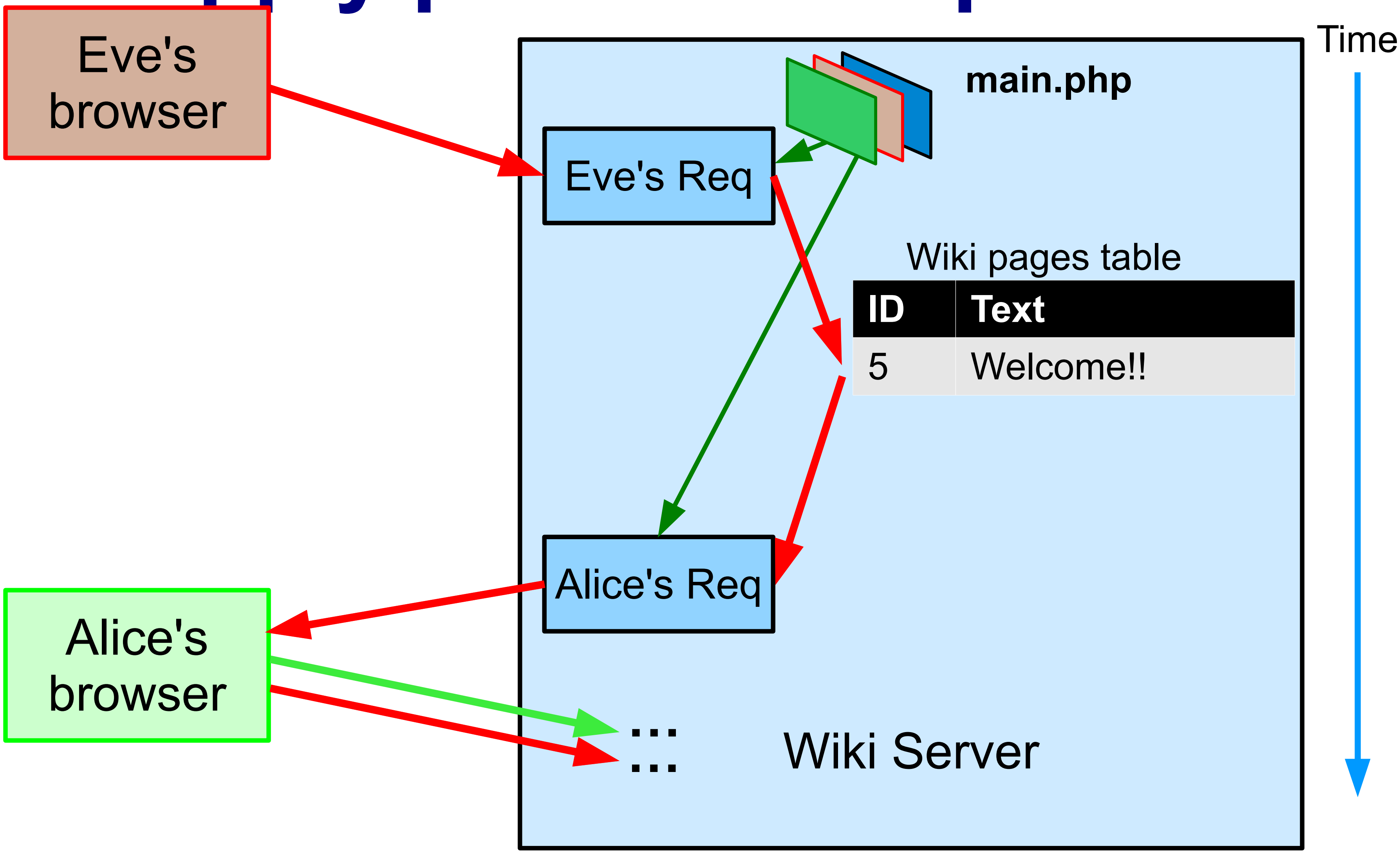
Retroactive patching: repair



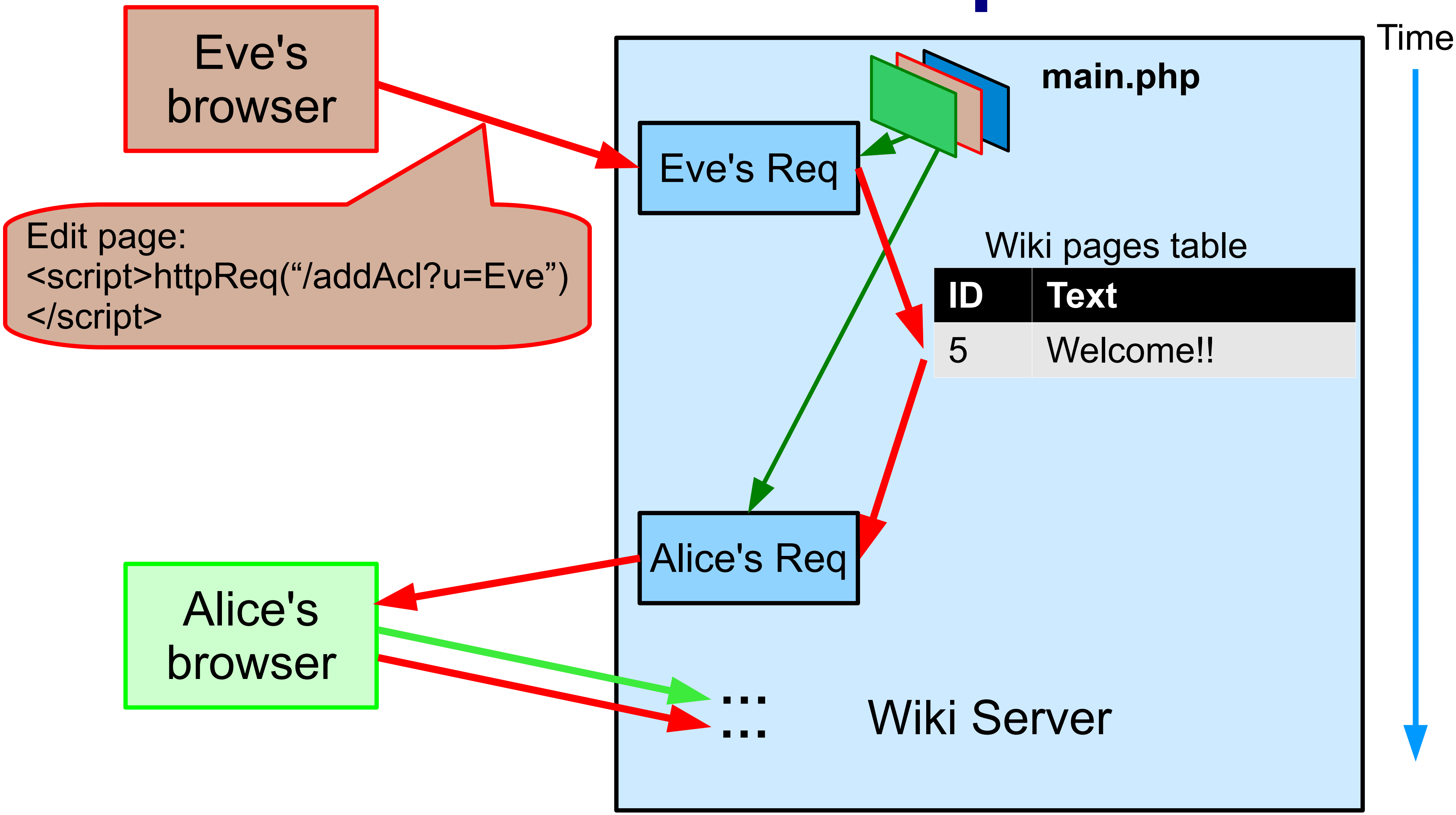
Rollback to before vulnerability was introduced



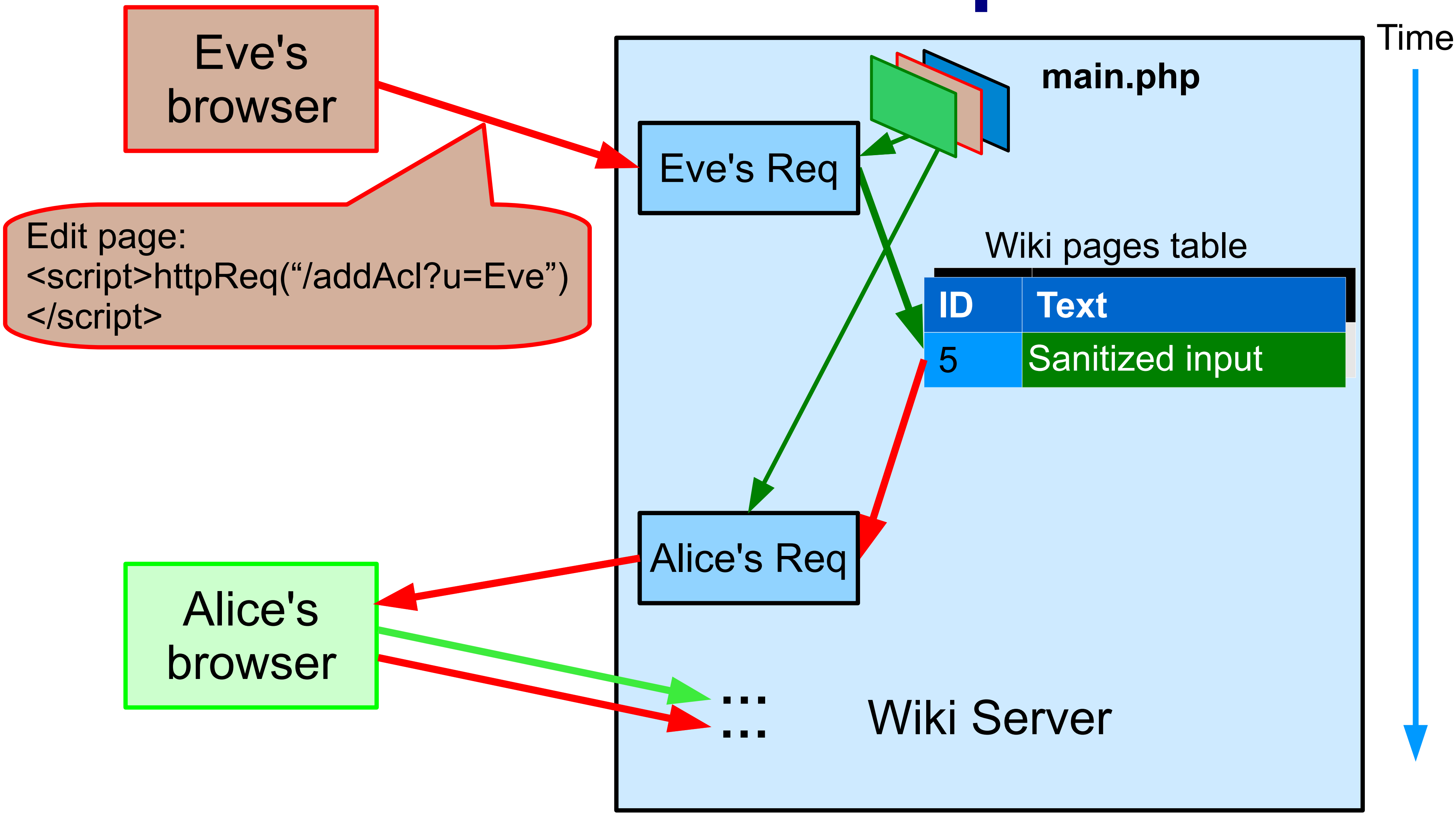
Retroactive patching: apply patch in the past



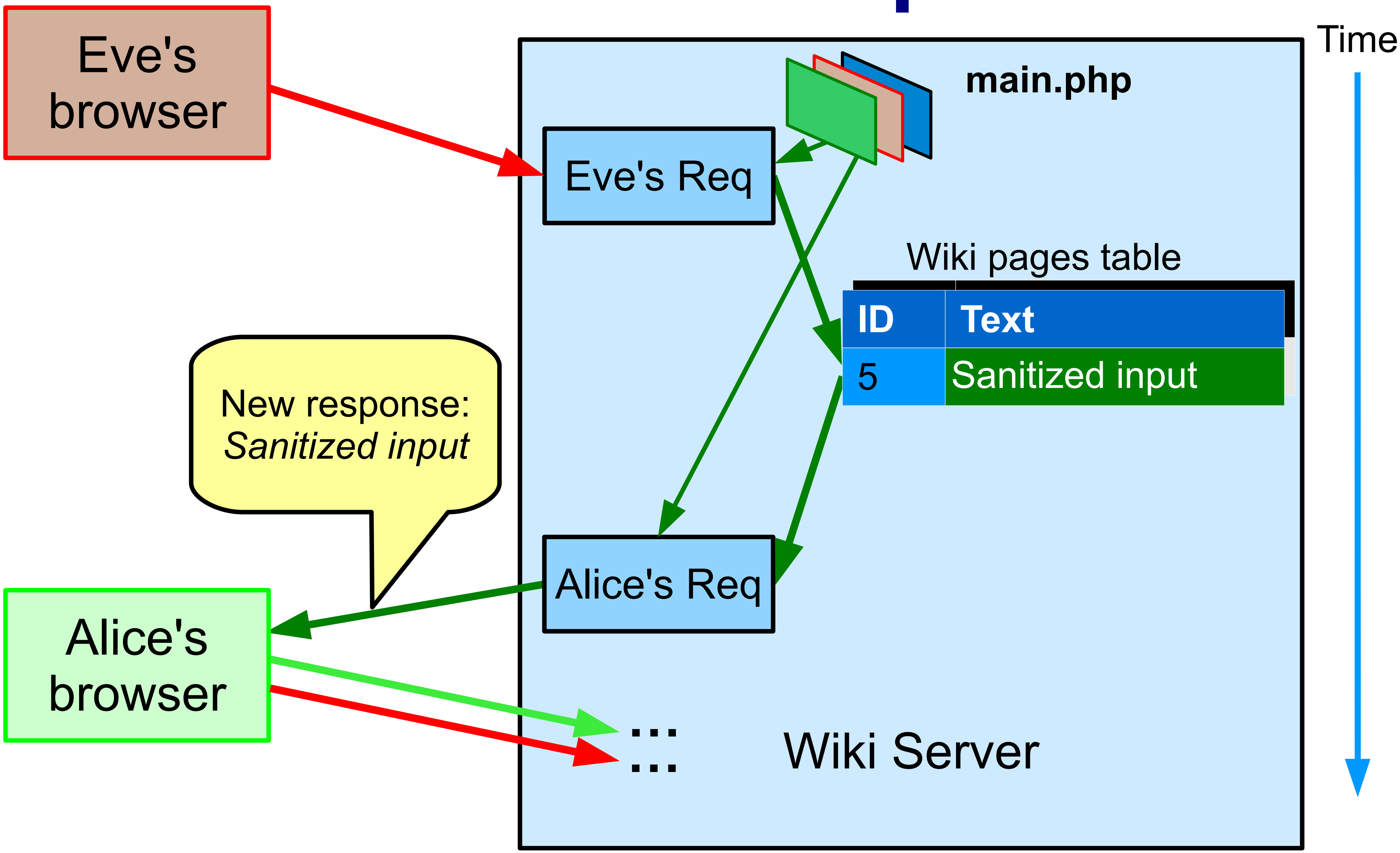
Retroactive patching: rerun affected requests



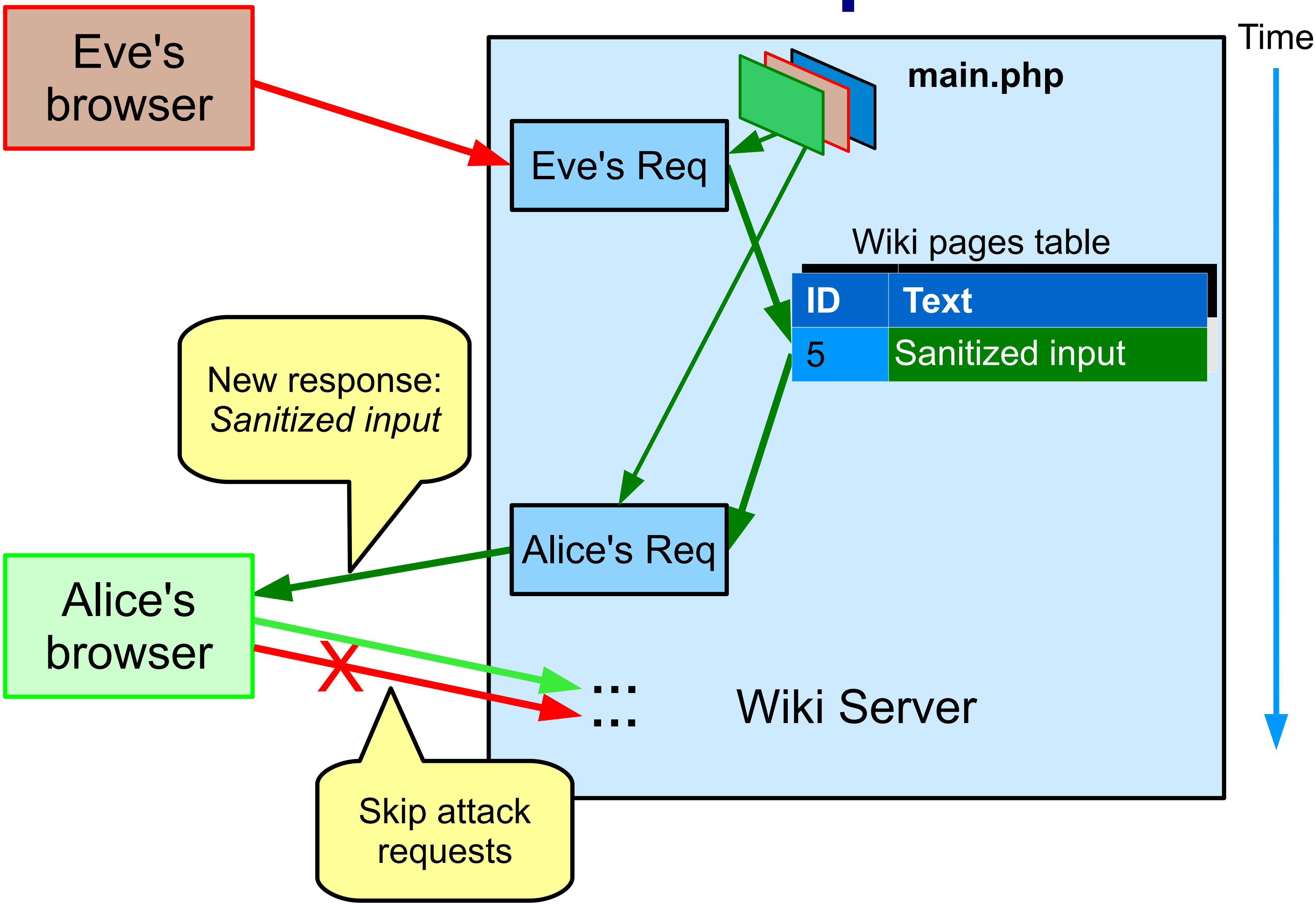
Retroactive patching: rerun affected requests



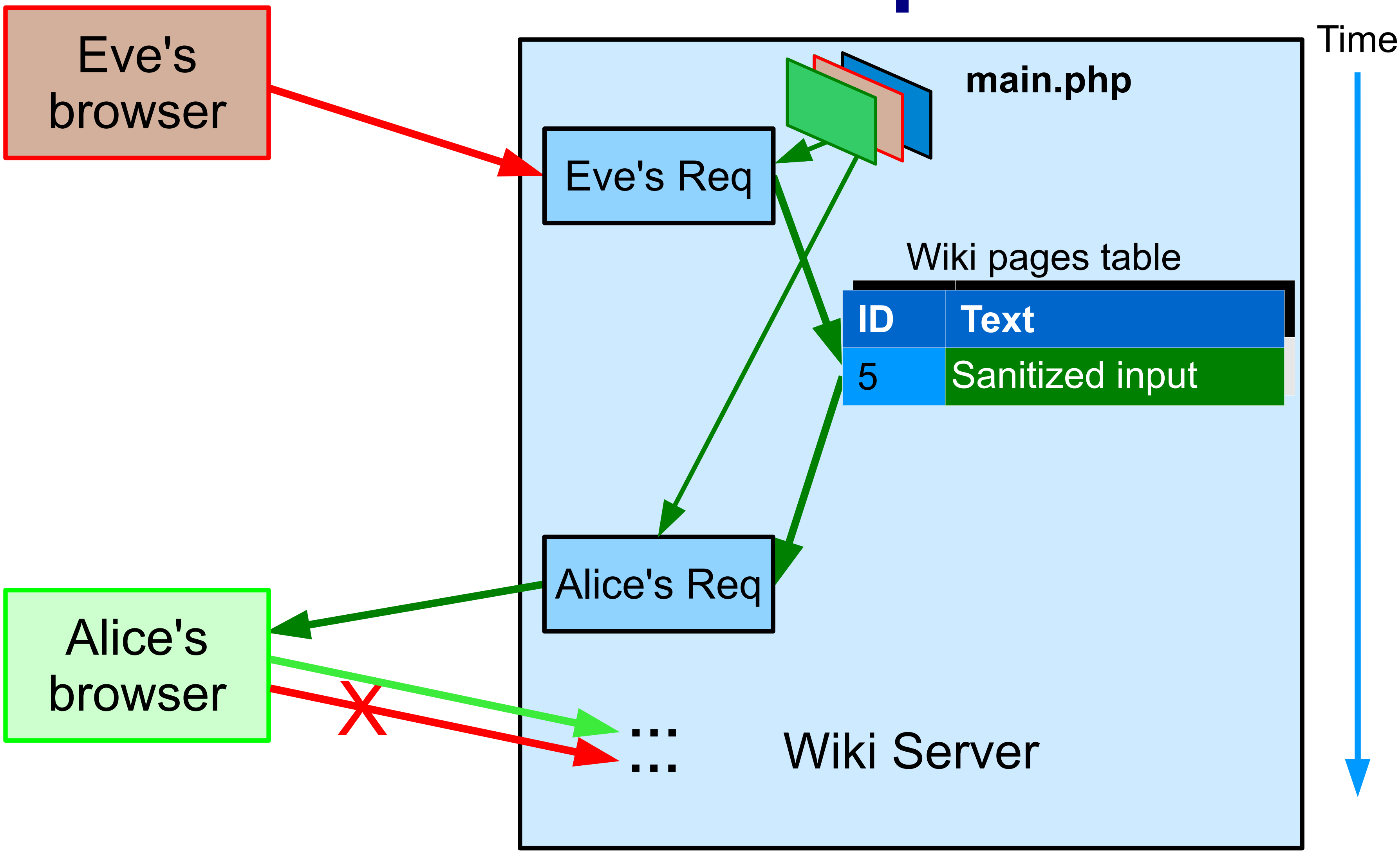
Retroactive patching: rerun affected requests



Retroactive patching: rerun affected requests



Retroactive patching: rerun affected requests



Do not need expert, just the patch

Challenge 2: reduce re-execution

- Warp re-executes requests for two reasons:
  - Request depends on attack
    - Results would be different without attack
    - Need: precise dependency tracking
  - Request re-executed to reapply legitimate changes
    - Need: avoid unnecessary rollback

Focus: database dependencies

- Dependencies arise due to shared state
- Web apps store state in database
- Must compute dependencies between SQL queries

Goals for dependency tracking

- Precise
  - Avoid false dependencies
  - Important because web applications often manage many independent pieces of data

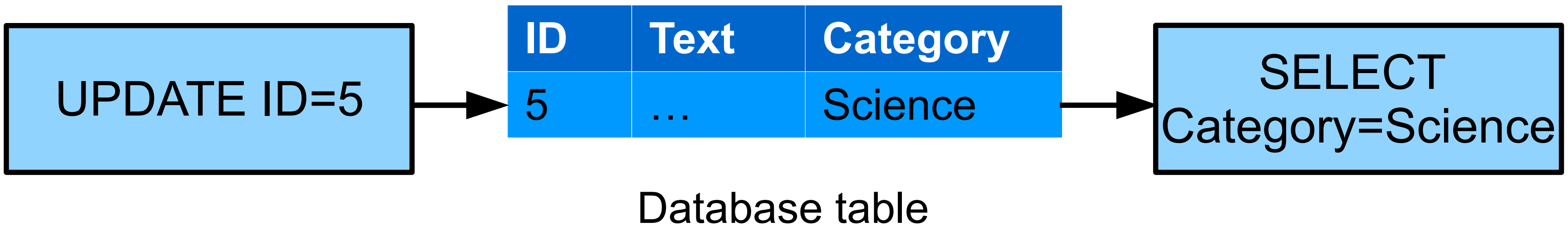
- Fast
  - Track dependencies without re-running the queries
  - Important because web applications often handle many independent requests

Dependency tracking strawmen

- Whole-table dependencies: fast but not precise
  - Reads depend on all prior writes on same table
  - Can determine table names in queries by statically looking at query's table list
  - False dependencies: queries can access independent rows in same table
- Re-execute reads: precise but slow
  - Re-execute each read, compare results before & after
  - Slow: requires re-executing every single read query

Achieving precise and static dependency tracking is hard

- Queries name rows by different attributes (columns)
- Queries do not specify every attribute

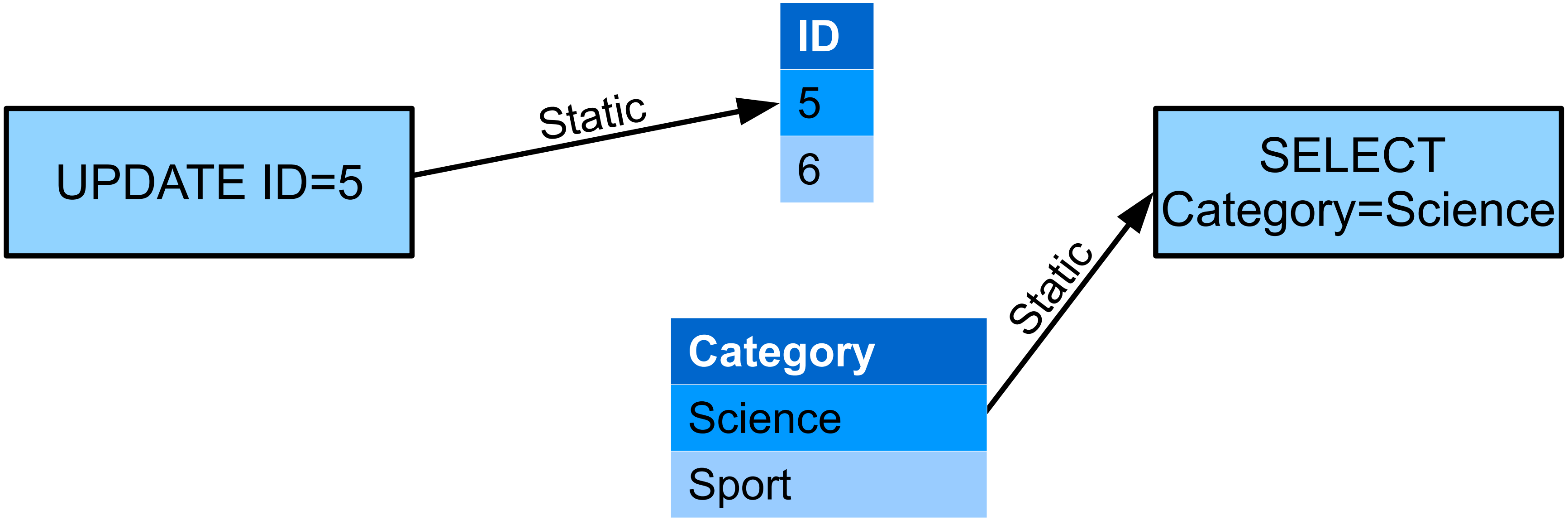


Solution: record write attributes at runtime

- For each write, record all attribute values of affected rows
- For reads, statically determine dependencies based on query's WHERE clause (easy + fast)

Solution: record write attributes at runtime

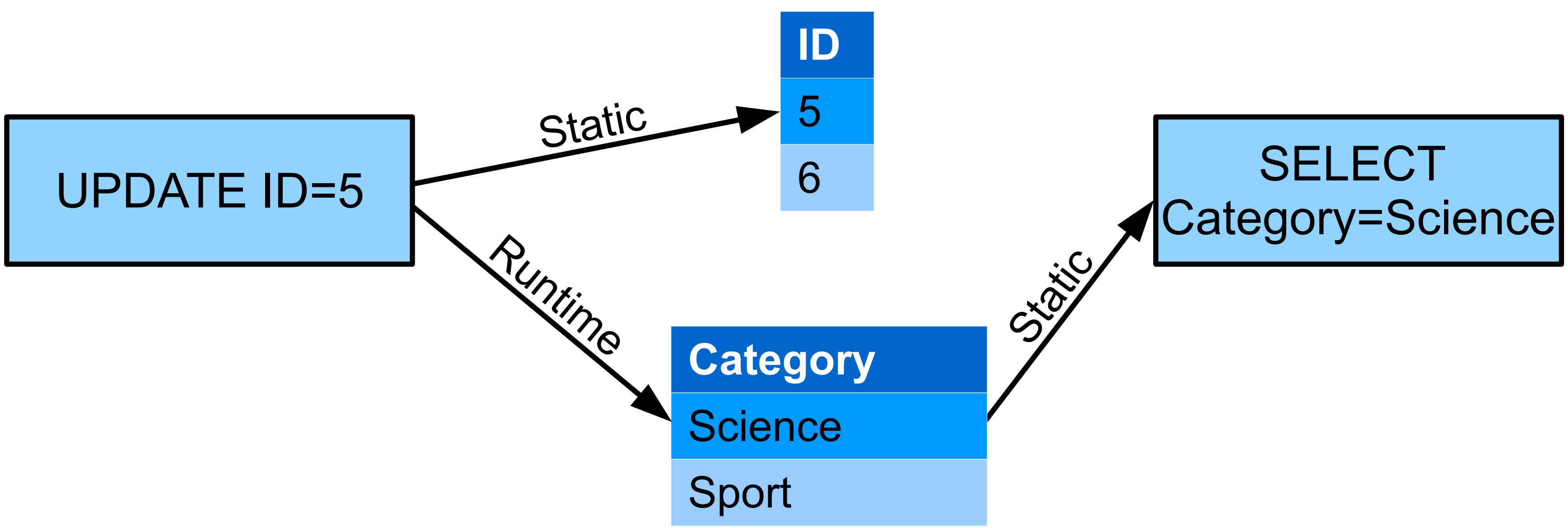
- For each write, record all attribute values of affected rows
- For reads, statically determine dependencies based on query's WHERE clause (easy + fast)



Possible dependency attributes

Solution: record write attributes at runtime

- For each write, record all attribute values of affected rows
- For reads, statically determine dependencies based on query's WHERE clause (easy + fast)



Possible dependency attributes

Challenge 2: reduce re-execution

- Warp re-executes requests for two reasons:
  - Request depends on attack
    - Results would be different without attack
    - Need: precise dependency tracking
  - Request re-executed to reapply legitimate changes
    - Need: avoid unnecessary rollback

Approach to avoiding unnecessary rollback

- Roll back only affected parts of the database
  - No need to re-apply changes to unaffected rows
  - Technique: row-level rollback
- Allow rolling back to any point in time
  - Helps avoid rolling back too far
  - No need to re-apply changes from before the attack
  - Technique: continuous checkpointing

Solution: continuous row-level checkpoints

- Keep track of all versions of every row over time
- Can roll back individual rows to any point in time

Valid time period (From, To)

ID	From	To	Text	Category
1	2	7	…	…
1	7	∞	…	…
2	4	∞	…	…
3	5	9	…	…
3	9	∞	…	…

Solution: continuous row-level checkpoints

- Keep track of all versions of every row over time
- Can roll back individual rows to any point in time

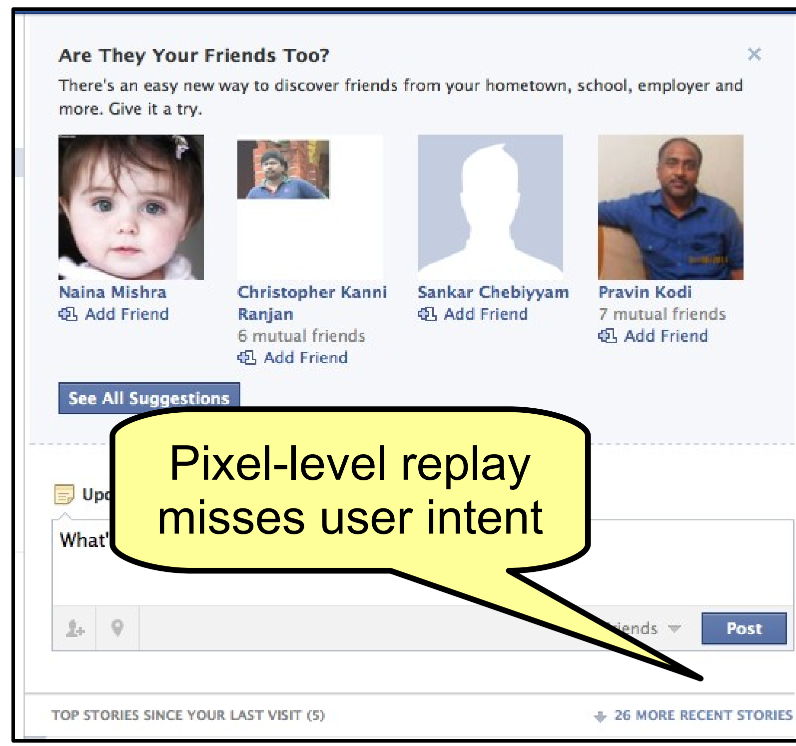
Valid time period (From, To)

ID	From	To	Text	Category
1	2	7	…	…
1	7	∞	…	…
2	4	∞	…	…
3	5	9	…	…
3	9	∞	…	…

- *Time-travel DB*: dependency tracking + continuous row-level checkpoints

Challenge 3: reduce user involvement during repair

- Pixel-level replay of user actions often meaningless
  - Results in a *conflict*

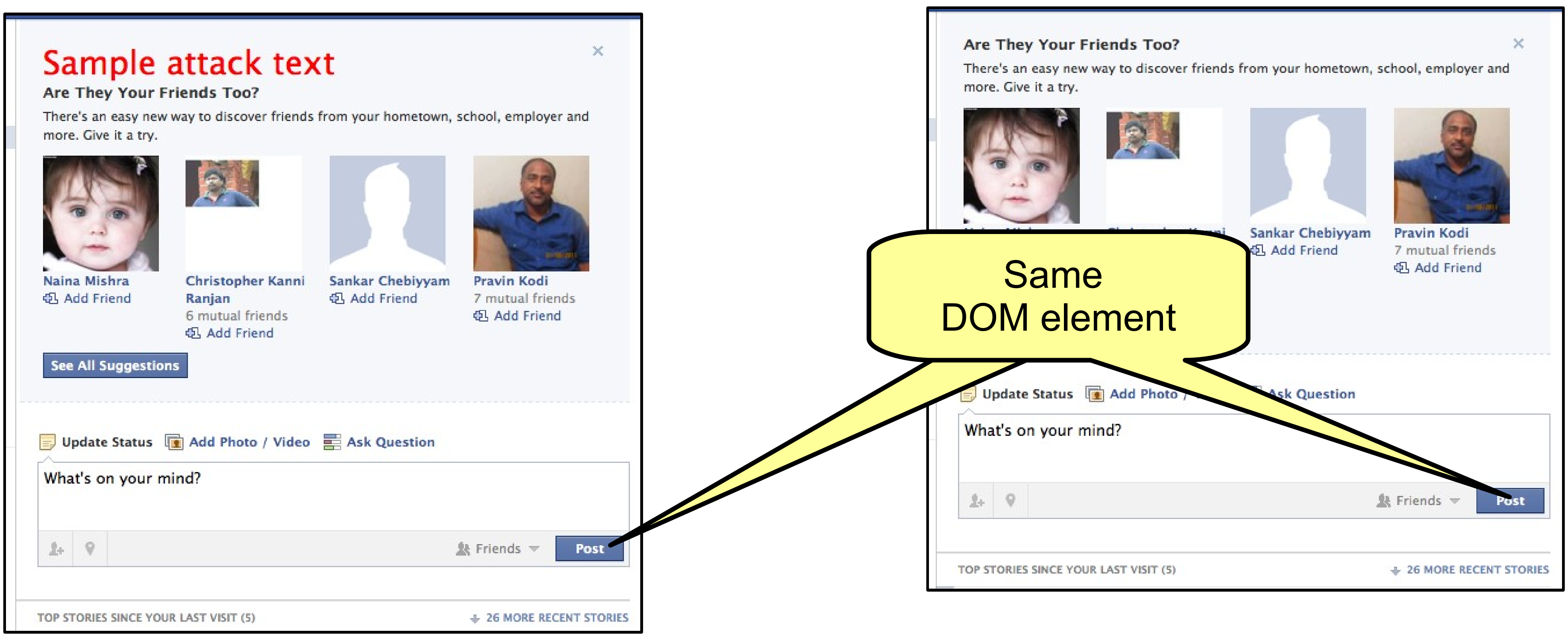


Idea: DOM-level replay

- Key observation: DOM has structure
  - Changing one element does not affect other elements
  - User action's intent tied to DOM element

Idea: DOM-level replay

- Key observation: DOM has structure
  - Changing one element does not affect other elements
  - User action's intent tied to DOM element

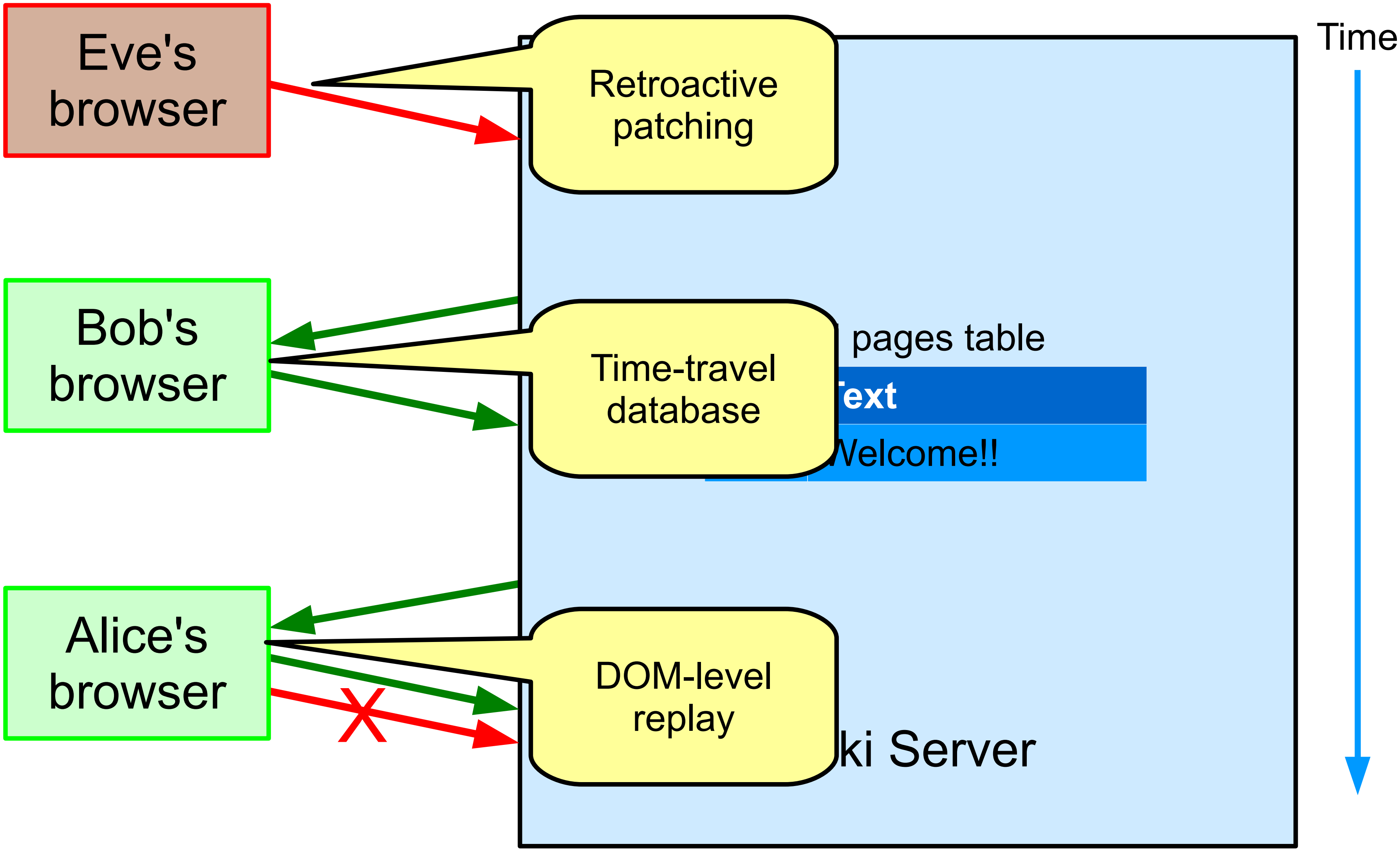


Idea: DOM-level replay

- Normal execution
  - Record user actions on DOM elements using a browser extension

- Repair
  - Replay user actions if DOM element unchanged
  - Three-way merge for text input elements
  - If DOM element changed, flag a conflict

Putting it together



Warp: Web application repair

- Prototype implementation of Warp
  - Postgres DB: SQL query rewriting
  - PHP, Apache: log requests, non-deterministic calls
  - Firefox: browser extension, upload log, re-execution
- Total: 8,500 lines of code (C, PHP, Python, JS)

Evaluation questions

- Can Warp support real applications?
- Can Warp recover from real attacks?
- What do the admin, users have to do?
- What are the runtime overheads of Warp?
- How long does repair take?

Warp works for real applications

- Ported three applications to run on Warp
  - MediaWiki (Wikipedia software)
  - Drupal (content management system)
  - Gallery2 (photo album software)

Warp works for real applications

- Ported three applications to run on Warp
  - MediaWiki (Wikipedia software)
  - Drupal (content management system)
  - Gallery2 (photo album software)
- No application source code changes
- Tens of lines of annotations on SQL schema, to specify columns for dependency tracking
- Yet, can recover integrity after attacks

MediaWiki attack workload

- Use five real vulnerabilities
  - One attacker, 3 victims
    - Attacker injects Javascript into a page
    - Attack code runs in victim's browsers
    - Attack code edits Wiki pages, …
    - Victims also browse and edit pages
  - 96 other users browse random Wiki pages, make edits
- One admin mistake

Warp recovers from wide range of attacks on MediaWiki

Attack	Initiating repair	User conflicts
Reflected XSS	Retroactive patching	0
Stored XSS	Retroactive patching	0
SQL injection	Retroactive patching	0
ACL mistake	Admin-initiated	1
CSRF	Retroactive patching	0
Clickjacking	Retroactive patching	3

Initiating recovery requires little effort

Attack	Initiating repair	User conflicts
Reflected XSS	Retroactive patching	0
Stored XSS	Retroactive patching	0
SQL injection	Retroactive patching	0
ACL mistake	Admin-initiated	1
CSRF	Retroactive patching	0
Clickjacking	Retroactive patching	3

Retroactive patching can use real MediaWiki patches

Warp's recovery is mostly automatic

Attack	Initiating repair	User conflicts
Reflected XSS	Retroactive patching	0
Stored XSS	Retroactive patching	0
SQL injection	Retroactive patching	0
ACL mistake	Admin-initiated	1
CSRF	Retroactive patching	0
Clickjacking	Retroactive patching	3

Warp incurs few conflicts, corresponding to real attack side-effects

Warp has low overheads

Workload	Page visit/s without Warp	Page visit/s with Warp	Warp log / page visit
Reading	8.46	6.43	3.71 KB
Editing	7.19	5.26	7.34 KB

- 24-27% throughput reduction in the server
- 1TB disk stores one year's worth of logs, for one server at 100% load
- Negligible overhead for logging in the browser

Warp avoids significant re-execution

Attack	Queries re-exec	Queries total	Repair time (s)	Orig time (s)
Reflected XSS	258	24,746	17.9	180.0
Stored XSS	293	24,740	16.7	179.2
SQL injection	524	24,541	29.7	177.8
ACL mistake	185	24,326	10.8	176.5
CSRF	19,799	24,578	1,644	175.0
Clickjacking	23,227	24,641	1,751	174.3

Warp avoids significant re-execution

Attack	Queries re-exec	Queries total	Repair time (s)	Orig time (s)
Reflected XSS	258	24,746	17.9	180.0
Stored XSS	293	24,740	16.7	179.2
SQL injection	524	24,541	29.7	177.8
ACL mistake	185	24,326	10.8	176.5
CSRF	19,799	24,578	1,644	175.0
Clickjacking	23,227	24,641	1,751	174.3

Warp re-executes a fraction of the original execution

Warp's repair time is order of magnitude smaller

Warp avoids significant re-execution

Attack	Queries re-exec	Queries total	Repair time (s)	Orig time (s)
Reflected XSS	258	24,746	17.9	180.0
Stored XSS	293	24,740	16.7	179.2
SQL injection	524	24,541	29.7	177.8
ACL mistake	185	24,326	10.8	176.5
CSRF	19,799	24,578	1,644	175.0
Clickjacking	23,227	24,641	1,751	174.3

Some patches require re-running all requests

Warp avoids significant re-execution

Attack	Queries re-exec	Queries total	Repair time (s)	Orig time (s)
Reflected XSS	258	24,746	17.9	180.0
Stored XSS	293	24,740	16.7	179.2
SQL injection	524	24,541	29.7	177.8
ACL mistake	185	24,326	10.8	176.5
CSRF	19,799	24,578	1,644	175.0
Clickjacking	23,227	24,641	1,751	174.3

Full re-execution slow in unoptimized prototype

Warp's repair algorithm scales well

Attack	100 users: Orig. time (s)	100 users: Repair time (s)	5000 users: Orig. time	5000 users: Repair time
Reflected XSS	180.04	17.87	49.2X	2.7X
Stored XSS	179.22	16.74	49.3X	3.3X
SQL injection	177.82	29.70	49.9X	9.2X
ACL mistake	176.52	10.75	50.3X	3.9X

50X workload, only
3-9X repair time

Related work

- Intrusion recovery:
  - Retro [Kim10], Taser [Goel05]: OS-level recovery inefficient for database recovery
  - Akkus and Goel [Akkus10]: only recovers from mistakes, requires manual guidance
- Deterministic record and replay: ReVirt [Dunlap02], Mugshot [Mickens10]
  - Cannot replay once something changes
- Vulnerability-specific predicates [Joshi05]:
  - Manual effort for each bug

Summary

- Intrusions are commonplace and inevitable
- Few recovery tools for web applications
- *Warp* restores integrity after attack
  - Retroactive patching, time-travel DB, DOM replay
  - Works for real apps: MediaWiki, Drupal, Gallery2
- *Warp* recovers from wide range of attacks

Thank you!