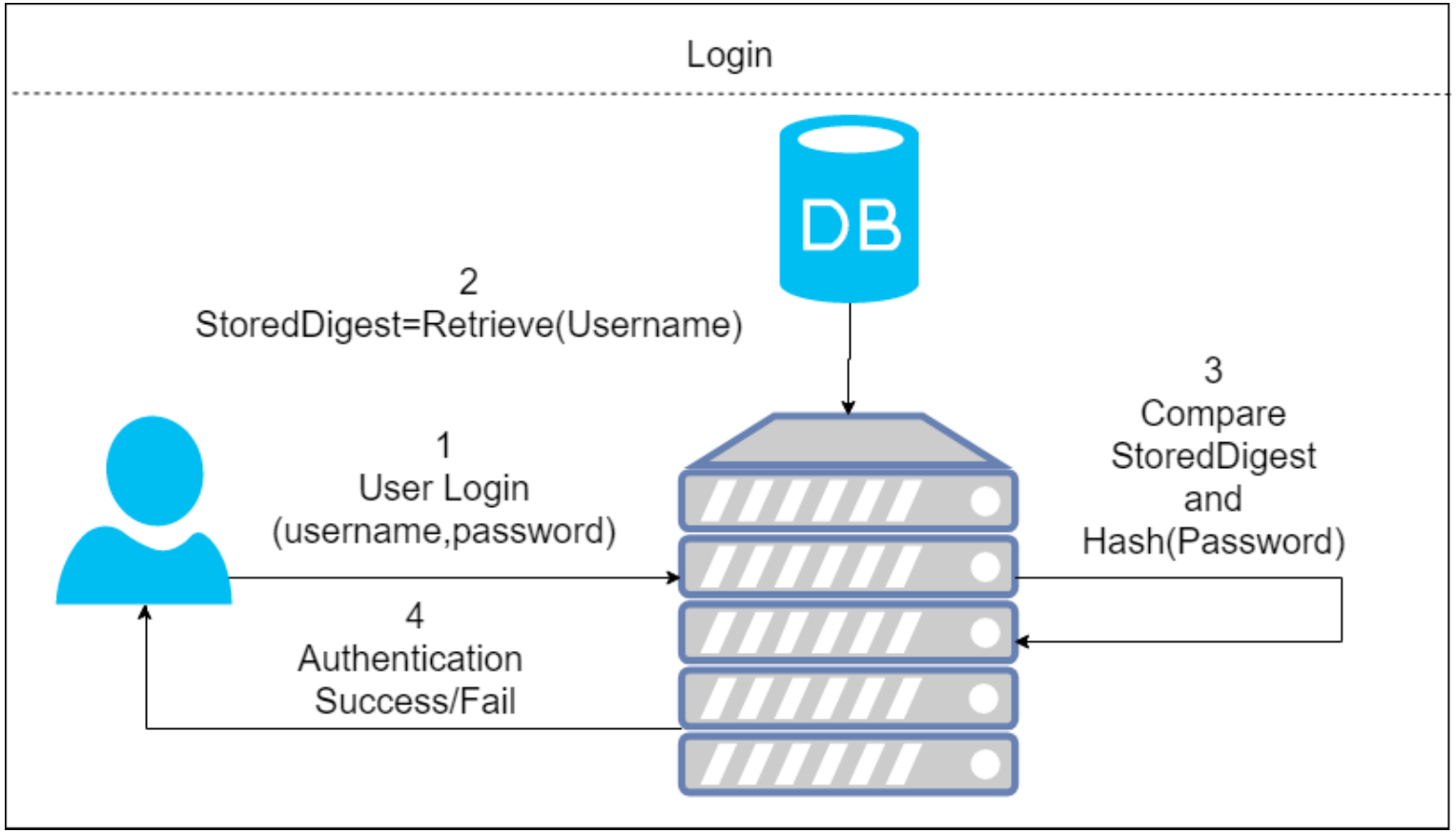
Practical Password Hardening based on TLS

Constantinos Diomedous and Elias Athanasopoulos

University of Cyprus

How authentication works today?



How web services protect passwords?

- Cryptographically secure hash functions
  - One way
- Salt
  - Used to differentiate common passwords

    `hash`(“password”+salt1) <> `hash`(“password”+salt2)

Database leaks

- 2012 6.4 Million LinkedIn
- 2014 1 Million Sony
- 2014 5 Million Gmail
- Weak passwords
  - Dictionary based (e.g., “password”)
  - Have patterns (e.g., “123456”)
  - Certain passwords used by multiple users
  - Anybody can compute the hash if they can guess the password

Password hardening

Slow cryptographically secure hash functions

- `scrypt`
- `bcrypt`

Splash Data 2018: approximately 10% of passwords used are one of the 25 most common (e.g., “password”, “123456”, “qwerty”)

- 100,000 password bcrypt digests
- Average `bcrypt` computation with default parameters is 65ms
- 65ms * 25 passwords * 100,000 digests ~ 1.88 days
- 10,000 passwords

Password hardening

Dedicated cryptographic services (e.g., Pythia, Phoenix, PHE, Pake)

- Use key to produce the digest (MAC)
  - **Where** is the key stored?
  - Anybody who has access to the key can recreate the mac if he can guess the password
- Multiple rounds of hashing
- Offline cracking is transformed to online cracking
- Difficult to be implemented and maintained by small companies
- Expensive to use as an external service

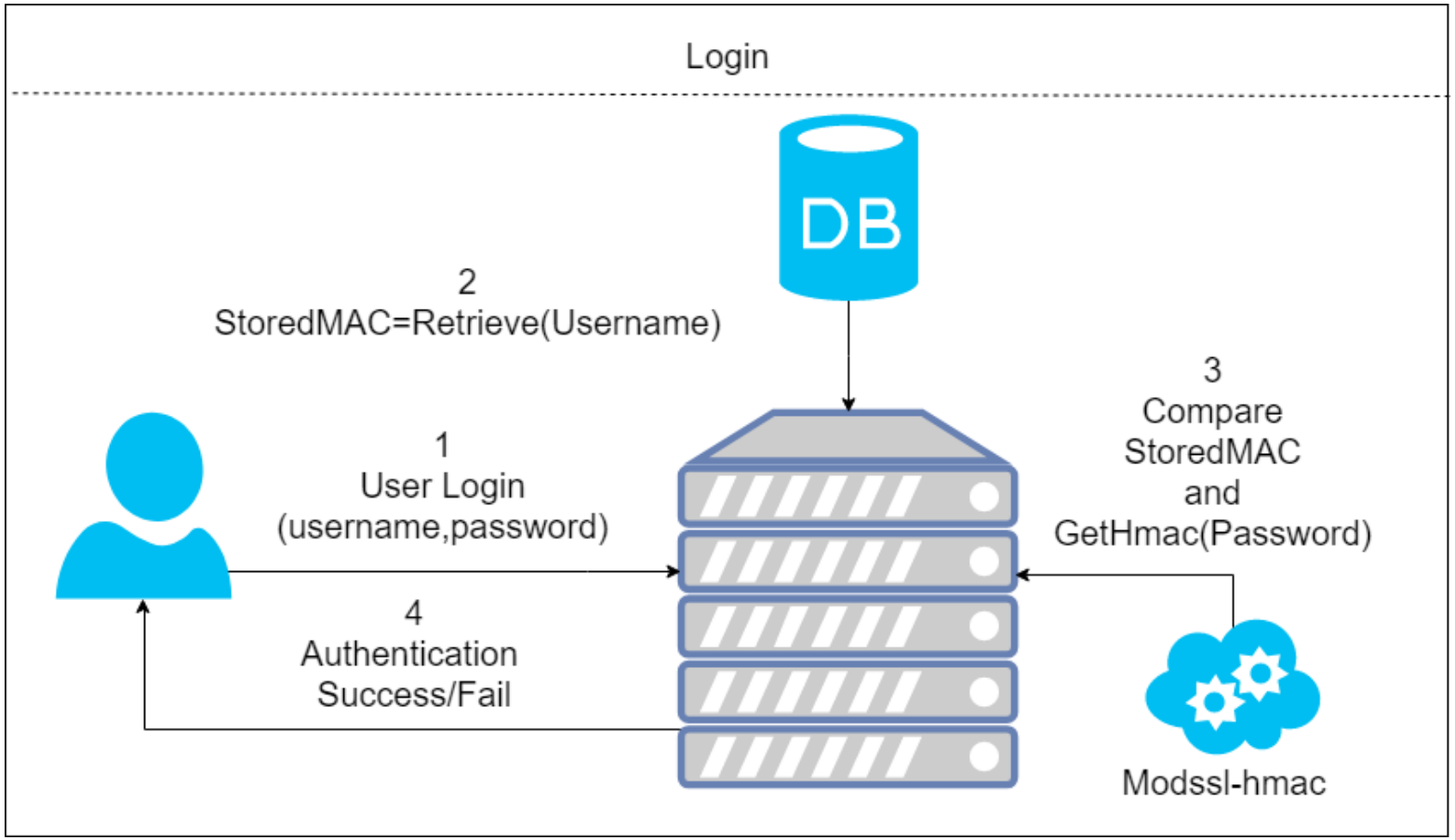
Our solution: `modssl-hmac`

- Local cryptographic service
- Leverage existing cryptographic elements
- MAC with TLS private key
- Password cracking now needs to leak TLS private key

Threat model

- Attacker leaked hashes with their salts
- Easy passwords exist on database
- Attacker has the computational power to crack easy passwords
- Attacker has no permanent access to web server
- Web server has TLS enabled

Authentication model with `modssl-hmac`



Modssl-hmac Requirements

- Transparent operation
- Easy deployment
- Web applications do not have direct access to TLS private key

Apache

- Web server
- Modular
- Each module
  - Process requests
  - Handles requests
  - Filters requests

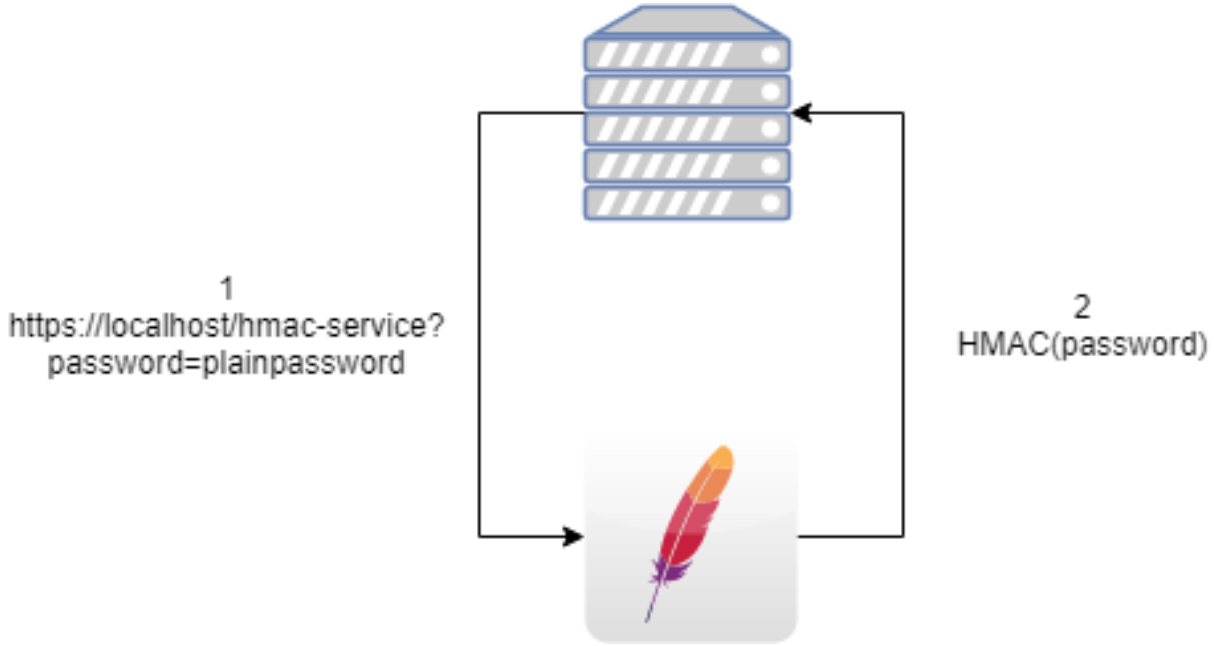
Modssl

- TLS support for apache
  - Initialize secure communication
  - Decode inbound content (filtering)
  - Encode outgoing content (filtering)

Modssl-hmac

- Add a hook to process local encrypted GET requests “*/hmac-service”
- Hmac with SHA256
- Use TLS private key of the server
- Multiple rounds of hashing(optional)

Modssl-hmac service architecture



- Encrypted
- TLS private key is never exposed to the web service

Deployment in existing web applications

Wordpress

- Web application for managing and publishing content
- Build in php
- Default 8,192 rounds of MD5
- Bcrypt plugin available

Drupal

- Another popular content management system
- Build in php
- Default 65,536 round of SHA512

Wordpress implementation

Wordpress

```
function crypt_private(…){
  …
  $count = 8192;
  $hash =
  md5($salt.$password, TRUE);
  do{
   $hash =
   md5($hash.$password, TRUE);
  }while(--$count);
  …
}
```

Wordpress `modssl-hmac` enabled

```
function crypt_private (…){
  …
  $curl = curl_init();
  curl_setopt_array($curl,
  array(
   CURLOPT_RETURNTRANSFER => true,
   CURLOPT_URL => "https://localhost/hmac-
   service?password=".
   urlencode($salt.$password),
   CURLOPT_USERAGENT => 'local‘ )
  );
  $hash = curl_exec($curl);
  …
}
```

Evaluation

	Mean	Deviation	Min	Max
WordPress (8192 iterations of MD5)	2.22	0.51	1.50	5.53
Drupal (65537 of SHA512)	65.16	15.89	47.20	206.60
`Bcrypt`(cost 11)	124.68	7.90	119.77	234.65
`Bcrypt`(cost 10 - default)	62.42	3.98	59.95	121.2
`Modssl-hmac`	50.23	7.80	38.25	135.1

Evaluation

	Mean	Deviation	Min	Max
WordPress (8192 iterations of MD5)	2.22	0.51	1.50	5.53
Drupal (65537 of SHA512)	65.16	15.89	47.20	206.60
`Bcrypt`(cost 11)	124.68	7.90	119.77	234.65
`Bcrypt`(cost 10 - default)	62.42	3.98	59.95	121.2
`Modssl-hmac`	50.23	7.80	38.25	135.1

Evaluation

	Mean	Deviation	Min	Max
WordPress (8192 iterations of MD5)	2.22	0.51	1.50	5.53
Drupal (65537 of SHA512)	65.16	15.89	47.20	206.60
`Bcrypt`(cost 11)	124.68	7.90	119.77	234.65
`Bcrypt`(cost 10 - default)	62.42	3.98	59.95	121.2
`Modssl-hmac`	50.23	7.80	38.25	135.1

Evaluation

	Mean	Deviation	Min	Max
WordPress (8192 iterations of MD5)	2.22	0.51	1.50	5.53
Drupal (65537 of SHA512)	65.16	15.89	47.20	206.60
`Bcrypt`(cost 11)	124.68	7.90	119.77	234.65
`Bcrypt`(cost 10 - default)	62.42	3.98	59.95	121.2
`Modssl-hmac`	50.23	7.80	38.25	135.1

Limitations

Migration of old passwords

- For each stored hash call the service
  - The output will replace the old hash
- On first successful login call the service for the plain password provided and replace the old hash

Limitations

SSL certificate renewal/revocation and CDNs

- Initialization
  - Generate random master key
  - Safely distribute it and encrypt it with public key of each server
- Service
  - Decrypts the encrypted master key with the private key and uses it for the hmac
- Update
  - Decrypts the encrypted master key with the old private key and encrypts it with the new public key

Conclusion

- Replace hash functions with mac based on TLS private key
  - Only ~ 50 LOC needs to change on the framework
  - Upgrade security with minimal performance cost
- Password cracking dependent on TLS private key
- Protect weak links with a local solution