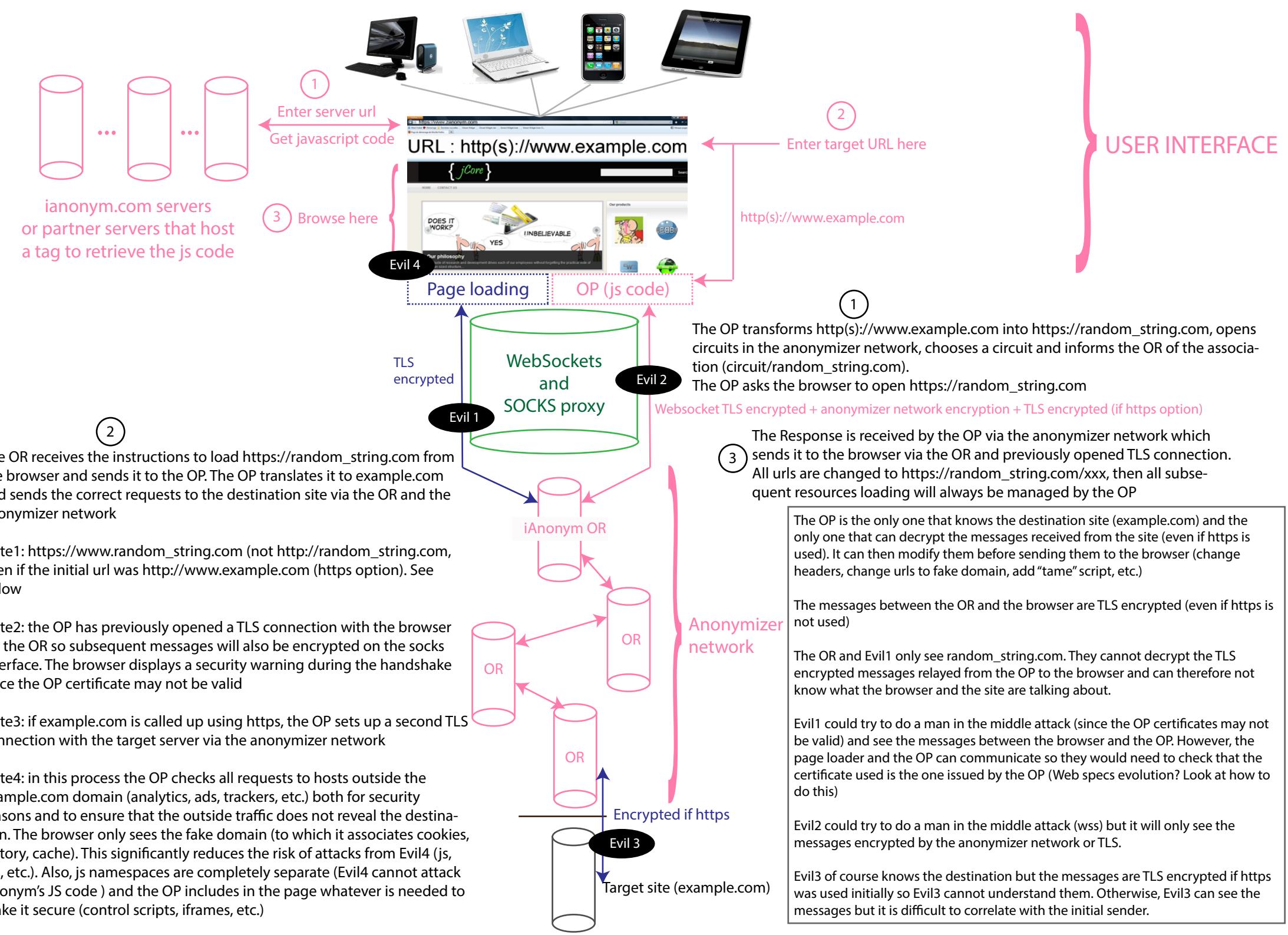

1

The OP transforms http(s)://www.example.com into https://random_string.com, opens circuits in the anonymizer network, chooses a circuit and informs the OR of the association (circuit/random_string.com).
The OP asks the browser to open https://random_string.com

2

The OR receives the instructions to load https://random_string.com from the browser and sends it to the OP. The OP translates it to example.com and sends the correct requests to the destination site via the OR and the anonymizer network

Note1: https://www.random_string.com (not http://random_string.com, even if the initial url was http://www.example.com (https option). See below

Note2: the OP has previously opened a TLS connection with the browser via the OR so subsequent messages will also be encrypted on the socks interface. The browser displays a security warning during the handshake since the OP certificate may not be valid

Note3: if example.com is called up using https, the OP sets up a second TLS connection with the target server via the anonymizer network

Note4: in this process the OP checks all requests to hosts outside the example.com domain (analytics, ads, trackers, etc.) both for security reasons and to ensure that the outside traffic does not reveal the destination. The browser only sees the fake domain (to which it associates cookies, history, cache). This significantly reduces the risk of attacks from Evil4 (js, css, etc.). Also, js namespaces are completely separate (Evil4 cannot attack Anonym's JS code) and the OP includes in the page whatever is needed to make it secure (control scripts, iframes, etc.)

3

The Response is received by the OP via the anonymizer network which sends it to the browser via the OR and previously opened TLS connection. All urls are changed to https://random_string.com/xxx, then all subsequent resources loading will always be managed by the OP

The OP is the only one that knows the destination site (example.com) and the only one that can decrypt the messages received from the site (even if https is used). It can then modify them before sending them to the browser (change headers, change urls to fake domain, add "tame" script, etc.)

The messages between the OR and the browser are TLS encrypted (even if https is not used)

The OR and Evil1 only see random_string.com. They cannot decrypt the TLS encrypted messages relayed from the OP to the browser and can therefore not know what the browser and the site are talking about.

Evil1 could try to do a man in the middle attack (since the OP certificates may not be valid) and see the messages between the browser and the OP. However, the page loader and the OP can communicate so they would need to check that the certificate used is the one issued by the OP (Web specs evolution? Look at how to do this)

Evil2 could try to do a man in the middle attack (wss) but it will only see the messages encrypted by the anonymizer network or TLS.

Evil3 of course knows the destination but the messages are TLS encrypted if https was used initially so Evil3 cannot understand them. Otherwise, Evil3 can see the messages but it is difficult to correlate with the initial sender.