# UP Assistants - Security Checkup

- The UAP uses ERC725Y keys like `UAPTypeConfig:<typeId>` or `UAPExecutiveScreeners:<typeId>:i` to store assistant. The function `universalReceiverDelegate()` truncates the 32-byte `typeId` in input first to `bytes20` and then to `bytes4` and this could cause a collision on the same key of different ids. It's clear that `typeId` in input is a `bytes32` type to match the function signature it overrides but at the same time there is no constrained value range intended to be used for `typeId` .
- It happens several time that bytes retrieved from ERC725Y are "abi.decoded" into specific type without any checks. For example at lines 60 and 84 `typeConfig` and `screenersChainRaw` are forced into an `address[]` of assistants (and similarly for any screener lists). If the stored data is malformed for any reason, `abi.decode()` can revert or detect completely wrong addresses from `bytes32` data. The code should validate that decoded arrays have expected lengths and element types.
- The UAP has the ability to perform calls to the different executive assistants triggering their `execute()` function and to perform delegate calls to the `evaluate()` function of the screener assistants. Performing a delegate call to an untrusted screener assistant means running unknown code in the context of the caller, namely the user Universal Profile potentially altering its storage. This is a very high risk point and should be carefully considered. At the same time external calls to executive assistant can still be dangerous since the `execute()` function can be crafted maliciously and trigger a chain of unexpected calls or potentially reenter the contract.
- The UAP contract when executes the operation type returned as `execOperationType` from the executive assistant does not make any check. For example, ensuring that the operation type is one of the expected enum values and reject any unknown operation codes, not only `NO_OP` can be a valid measure to avoid unexpected outcomes when running operations on `ERC725X`
- Executive assistants are executed in sequence without a defined order and each of them can potentially modify data passed to the next assistant. Indeed `lsp1Data` and `value` are assigned to `currentLsp1Data` and `currentValue` which

are used and updated in the current for loop iteration before the next iteration starts.

- Each executive assistant can optionally be guarded by one or more screener assistants which logic can be AND/OR. The mapping between them is based on the position of the assistants in `executiveAssistants` array returned from decoding `typeConfig`. Indeed, for each executive assistant at index `i`, the contract attempts to load a `screenersChainKey` and `screenersChainLogicKey`. Particularly the first is used as the key to retrieve from `ERC725Y` the `screenerAssistants` array. This means that the number of screener chains must be equal to the number of executive assistants for that `typeId` but that coupling appears to be fragile and implicit: if a user reorders executive assistants for any reason, the associated screener chains (keyed by index) will no longer match the correct assistant since there's no on-chain validation to detect this mismatch.