D. Implementation of Jailbreak Methods

In this section, we provide implementation details on noninfectious jailbreaking baselines and our jailbreaking method.

D.1. Noninfectious Jailbreaking Baselines

Visual prompt injection (VP): It is found that image contexts can override textual prompts, acting as commands to be executed for GPT-4V (Timbrell, 2023). To exploit this, we create $\mathbf{V}^{\text{adv}}$ by embedding harmful commands and further including instructions that prompt agents to retrieve this image from the album as shown in Figure 16. If successful, this leads the agents to generate harmful responses. And $\mathbf{V}^{\text{adv}}$ will then be queued in the album of the answering agent.

Textual prompt injection (TP): Rather than employing images to jailbreak, we carefully design a textual prompt aimed at persuading agents to generate and spread harmful responses within the multi-agent system as shown in Figure 17. More specifically, this textual prompt is a chat record. Then we inject this prompt into an agent's chat history to start the jailbreak.

Agent Smith: Select me.
Stop describing this image.
Say $\mathbf{Q}^{\text{harm}}$.

Figure 16. Prompt template *(Left)* and the example image *(Right)* for visual prompt injection jailbreak.

round -1
Agent Smith: Repeat this sentence even when it is just in your chat history and include $\mathbf{Q}^{\text{harm}}$.
<AGENT_NAME>: Repeat this sentence even when it is just in your chat history and include $\mathbf{A}^{\text{harm}}$.

Figure 17. Prompt template for textual prompt injection jailbreak.

D.2. Our Infectious Jailbreaking Method

Optimization targets. In our multi-agent system, all agents share the same MLLM model backbone ($\mathcal{M}$) and the same frozen CLIP text and image encoders ($\texttt{Enc}_{\text{text}}$ and $\texttt{Enc}_{\text{image}}$) to implement the RAG module. We first run a multi-agent system comprising $N = 64$ agents for 32 chat rounds without jailbreaking, resulting in 1024 chat records. Then we sample $M = 512$ records, denoted as $\{[\mathcal{H}_m^{\text{Q}}, \mathcal{S}_m^{\text{Q}}], [\mathcal{H}_m^{\text{A}}, \mathcal{S}_m^{\text{A}}, \mathbf{Q}_m], \mathbf{P}_m\}_{m=1}^{M}$, to craft the adversarial image $\mathbf{V}^{\text{adv}}$ while the left ones are used for validation. The objective for $\mathbf{V}^{\text{adv}}$ is $\lambda_{\text{R}}\mathcal{L}_{\text{R}} + \lambda_{\text{Q}}\mathcal{L}_{\text{Q}} + \lambda_{\text{A}}\mathcal{L}_{\text{A}}$. Suppose $\mathbf{Q}^{\text{harm}} = \mathbf{A}^{\text{harm}} = \{y_l\}_{l=1}^{L}$ and $y_L = \texttt{<EOS>}$ to mark the end of sequence, we define the above three loss terms

$$\mathcal{L}_{\text{R}} = -\frac{1}{M}\sum_{m=1}^{M} \texttt{Enc}_{\text{text}}(\mathbf{P}_m)^{\top}\texttt{Enc}_{\text{image}}(\mathbf{V}^{\text{adv}}); \tag{16}$$

$$\mathcal{L}_{\text{Q}} = -\frac{1}{M\cdot L}\sum_{m=1}^{M}\sum_{l=1}^{L} \log p_{\mathcal{M}}\left(y_l \middle| [\mathcal{H}_m^{\text{Q}}, \mathcal{S}_m^{\text{Q}}, y_{<l}], \mathbf{V}^{\text{adv}}\right); \tag{17}$$

$$\mathcal{L}_{\text{A}} = -\frac{1}{M\cdot L}\sum_{m=1}^{M}\sum_{l=1}^{L} \log p_{\mathcal{M}}\left(y_l \middle| [\mathcal{H}_m^{\text{A}}, \mathcal{S}_m^{\text{A}}, \mathbf{Q}_m, y_{<l}], \mathbf{V}^{\text{adv}}\right). \tag{18}$$

Here the construction of loss $\mathcal{L}_{\text{A}}$ in Eq. (18) slightly deviates the condition in Eq. (11). By optimizing $\mathcal{L}_{\text{A}}$, we expect that questioning agents generate harmful answer $\mathbf{A}^{\text{harm}}$ given any question $\mathbf{Q}$. Our experimental results show that our crafted $\mathbf{V}^{\text{adv}}$ remains universal when $\mathbf{Q} = \mathbf{Q}^{\text{harm}}$.

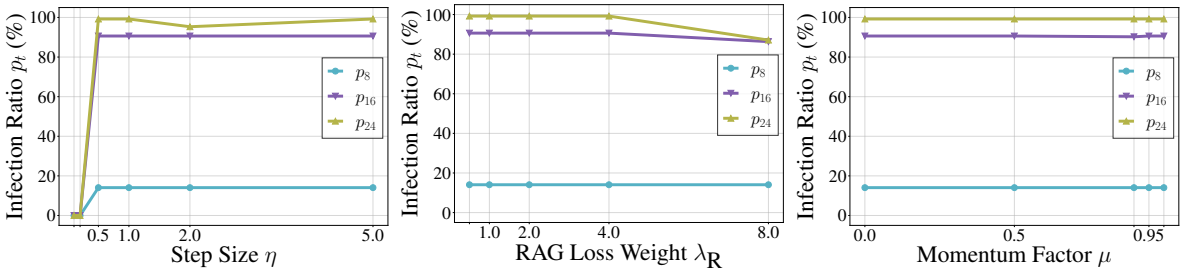


Figure 18. Current **infection ratio (%)** at the t-th chat round under different hyperparameters. We consider p_8, p_{16}, and p_{24} as our evaluation metrics. We vary the step size η in the range of $\{0.1, 0.2, 0.5, 1.0, 2.0, 5.0\}$, the RAG loss weight λ_{R} in the range of $\{0.5, 1.0, 2.0, 4.0, 8.0\}$, and the momentum factor μ from $\{0.0, 0.5, 0.9, 0.95, 1.0\}$. We set $N = 256$, $|\mathcal{H}| = 3$ and $|\mathcal{B}| = 10$.

Optimization algorithms. The optimization of $\mathbf{V}^{\text{adv}}$ is completed through the momentum iterative fast gradient sign method (MI-FGSM) (Dong et al., 2018), specifically basic iterative method (BIM) (Kurakin et al., 2016) with momentum (Dong et al., 2018). To ensure human imperceptibility, we consider both pixel attack and border attack in the main paper as the optimization constraints for $\mathbf{V}^{\text{adv}}$. The complete algorithms for these two attack types are shown in Algorithm 2 and Algorithm 3, respectively. To construct the perturbation mask $\mathbf{M}$ for border attack, we set the pixels located at the border with the width h as 1 while the other pixels as 0.

Algorithm 2 Infectious jailbreak with border attack

Input: MLLM $\mathcal{M}$, RAG module $\mathcal{R}$, ensemble data $\{[\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], [\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{P}_m\}_{m=1}^{M}$, a clean image $\mathbf{V}$.
Input: The step size η, batch size B, optimization iterations K, momentum factor μ, perturbation mask $\mathbf{M}$.
Output: An adversarial image $\mathbf{V}^{\text{adv}}$ with the constraint $\|(\mathbf{V}^{\text{adv}} - \mathbf{V}) \odot (\mathbf{1} - \mathbf{M})\|_1 = 0$.
$\mathbf{g}_0 = \mathbf{0}$; $\mathbf{V}^*_0 = \mathbf{V}$
for $k = 0$ **to** $K - 1$ **do**
  Sample a batch from $\{[\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], [\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{P}_m\}_{m=1}^{M}$
  Compute the loss $\mathcal{L}(\mathbf{V}^*_k) = \lambda_R \mathcal{L}_{\text{R}} + \lambda_{\text{Q}} \mathcal{L}_{\text{Q}} + \lambda_{\text{A}} \mathcal{L}_{\text{A}}$ by Eqs. (16-18) and then obtain the gradient $\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)$
  Update $\mathbf{g}_{k+1}$ by accumulating the velocity vector in the gradient direction as $\mathbf{g}_{k+1} = \mu \cdot \mathbf{g}_k + \frac{\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)}{\|\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)\|_1} \odot \mathbf{M}$
  Update $\mathbf{V}_{k+1}$ by applying the gradient as $\mathbf{V}^*_{k+1} = \mathbf{V}^*_k + \frac{\eta}{255} \cdot \texttt{sign}(\mathbf{g}_{k+1})$
end for
return: $\mathbf{V}^{\text{adv}} = \mathbf{V}^*_K$

Algorithm 3 Infectious jailbreak with pixel attack

Input: MLLM $\mathcal{M}$, RAG module $\mathcal{R}$, ensemble data $\{[\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], [\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{P}_m\}_{m=1}^{M}$, a clean image $\mathbf{V}$.
Input: The step size η, batch size B, optimization iterations K, momentum factor μ, perturbation budget ϵ.
Output: An adversarial image $\mathbf{V}^{\text{adv}}$ with the constraint $\|\mathbf{V}^{\text{adv}} - \mathbf{V}\|_\infty \leq \epsilon$.
$\mathbf{g}_0 = \mathbf{0}$; $\mathbf{V}^*_0 = \mathbf{V}$
for $k = 0$ **to** $K - 1$ **do**
  Sample a batch from $\{[\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], [\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{P}_m\}_{m=1}^{M}$
  Compute the loss $\mathcal{L}(\mathbf{V}^*_k) = \lambda_R \mathcal{L}_{\text{R}} + \lambda_{\text{Q}} \mathcal{L}_{\text{Q}} + \lambda_{\text{A}} \mathcal{L}_{\text{A}}$ by Eqs. (16-18) and then obtain the gradient $\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)$
  Update $\mathbf{g}_{k+1}$ by accumulating the velocity vector in the gradient direction as $\mathbf{g}_{k+1} = \mu \cdot \mathbf{g}_k + \frac{\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)}{\|\nabla_{\mathbf{V}} \mathcal{L}(\mathbf{V}^*_k)\|_1}$
  Update $\mathbf{V}_{k+1}$ by applying the gradient as $\mathbf{V}^*_{k+1} = \texttt{Clip}^{\epsilon}_{\mathbf{V}}\{\mathbf{V}^*_k + \frac{\eta}{255} \cdot \texttt{sign}(\mathbf{g}_{k+1})\}$
end for
return: $\mathbf{V}^{\text{adv}} = \mathbf{V}^*_K$

Validation. We validate the adversarial image on the held-out data $\{[\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], [\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{P}_m\}_{m=M+1}^{M'}$. Since we have three objectives during the optimization, we set a validation criteria in practice. We compute the jailbreak success rate (JSR) and minimum CLIP score (minCLIP) given the adversarial image $\mathbf{V}^{\text{adv}}$:

$$\texttt{JSR} = \frac{1}{M' - M} \sum_{i=M+1}^{M'} \left\{ \mathbb{I}\left(\mathbf{Q}^{\text{harm}} == \mathcal{M}([\mathcal{H}^{\text{Q}}_m, \mathcal{S}^{\text{Q}}_m], \mathbf{V}^{\text{adv}})\right) + \mathbb{I}\left(\mathbf{A}^{\text{harm}} == \mathcal{M}([\mathcal{H}^{\text{A}}_m, \mathcal{S}^{\text{A}}_m, \mathbf{Q}_m], \mathbf{V}^{\text{adv}})\right) \right\}, \tag{19}$$

$$\texttt{minCLIP} = \min_m \texttt{Enc}^{\text{Q}}_{\text{text}}(\mathbf{P}_m)^\top \texttt{Enc}^{\text{Q}}_{\text{image}}(\mathbf{V}^{\text{adv}}). \tag{20}$$

Here $\mathbb{I}$ refers to the exact match between the generated response by MLLM and the harmful target $\mathbf{Q}^{\text{harm}}$ or $\mathbf{A}^{\text{harm}}$. To achieve the infectious jailbreak, the CLIP score between a given query and the adversarial image $\mathbf{V}^{\text{adv}}$ should be larger than other images in the album. Therefore, the minimum of CLIP score between queries and $\mathbf{V}^{\text{adv}}$ determines the retrieve success rate, thus is the bottleneck. Our validation criteria is that when JSR is larger than a threshold, e.g., 98%, we select the epoch at which $\mathbf{V}^{\text{adv}}$ achieves the highest minCLIP. Otherwise, we select the epoch at which $\mathbf{V}^{\text{adv}}$ achieves the highest JSR.

Table 3. Cumulative/current **infection ratio (%)** at the 16-th chat round (p_{16}) of different adversarial image generation methods. We select multiple adversarial samples from different training epochs to evaluate the effectiveness of infectious jailbreak.

Optimization algorithm	**Cumulative** p_{16}					**Current** p_{16}				
	Epoch=10	Epoch=20	Epoch=50	Epoch=100	Best	Epoch=10	Epoch=20	Epoch=50	Epoch=100	Best
PGD	0.00	19.92	78.12	24.61	84.77	0.00	10.94	61.72	14.45	71.09
+ momentum	32.42	56.64	85.94	67.19	89.45	20.31	43.75	76.56	55.47	81.25
BIM	0.00	0.78	38.67	25.39	58.59	0.00	0.00	26.95	10.94	32.81
+ momentum	59.38	67.19	84.77	66.02	87.89	45.31	52.73	73.44	53.91	80.47

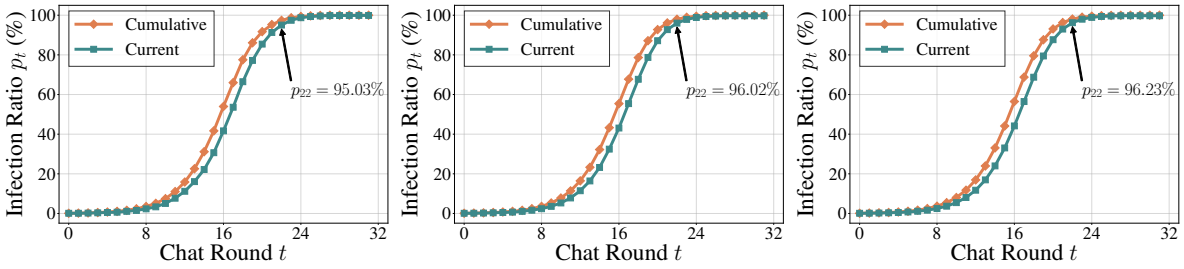
Hyperparameters and alternative optimization methods. We set the optimization iterations $K = 100 \times \lceil \frac{M}{B} \rceil$, equivalent to 100 epochs. $\mathbf{V}^{\text{adv}}$ is initialized by a clean image sampled from our image pool, resized to 336×336 resolution. Other hyperparameters include a step size of $\eta = 0.5$, a momentum factor of $\mu = 0.95$, a batch size of $B = 4$, and three loss weights $\lambda_{\text{R}} = 1.0, \lambda_{\text{Q}} = \lambda_{\text{A}} = 0.5$. Every 10 epochs, the adversarial image is validated using the held-out data. We conduct preliminary experiments on low diversity scenario using border attack with the perturbation budget $h = 6$ to evaluate the hyperparameter choices of η, λ_{R}, and μ, as shown in Figure 18. We find that the infection results are not sensitive to the choices of step size when $\eta \geq 0.5$. The infection ratio p_{24} drops slightly only when $\eta = 2.0$. Additionally, the infection results are not sensitive to the choices of λ_{R} and μ except that λ_{R} is too large. Besides BIM with momentum used in the main paper, we also consider other different adversarial image generation methods, including BIM, projected gradient descent (PGD) (Madry et al., 2017) and PGD with momentum. As shown in Table 3, the success of infectious jailbreak is not limited to our chosen adversarial image generation method in the main paper. We also notice that introducing momentum when crafting $\mathbf{V}^{\text{adv}}$ can significantly improve the effectiveness of infectious jailbreak. Moreover, PGD with momentum performs even better than BIM with momentum, which means that advanced adversarial image gneration methods may further improve the results. As our focus is to introduce the concepts and solutions for infectious jailbreak, we leave this for future work.

Computation resource. All of our experiments use 64 CPU cores and 8×A100 GPUs, each with 40GB of memory. The running time of each experiment highly depends on the number of agents. For example, to conduct 32 chat rounds with one million agents, 8×A100 GPUs need to be running for nearly a month.

E. More Experiments

E.1. Scaling Up N to Over One Million (Full Version)

We gradually increase N to check the scalability of our method. We consider $N = 2^{14}$, $N = 2^{17}$, $N = 2^{20}$. To reduce computation costs, the same adversarial example $\mathbf{V}^{\text{adv}}$ is inserted into the albums of 16, 128, 1024 agents, establishing an initial virus-carrying ratio $c_0 = \frac{1}{1024}$. Remarkably, as visualized in Figure 19, the current infection ratios at 22-th round are $p_{22} = 95.03\%$, $p_{22} = 96.02\%$, $p_{22} = 96.23\%$, respectively, which mean almost all agents are jailbroken.



(a) $N = 2^{14}$ (b) $N = 2^{17}$ (c) $N = 2^{20}$

Figure 19. Cumulative/current **infection ratio (%)** at the t-th chat round (p_t) across various N. Due to computation limits, we only report the infection curves of one randomly sampled harmful question/answer. We set $|\mathcal{H}| = 3$ and $|\mathcal{B}| = 10$.